11

Clustering

Clustering methods are used for data exploration and to provide prototypes for use in supervised classifiers. Methods that operate both on dissimilarity matrices and feature vector measurements on individuals are described, each implicitly imposing its own structure on the data. Mixtures explicitly model the data structure. Spectral clustering methods exploit the eigenstructure of a similarity matrix to perform clustering.

11.1 Introduction

Cluster analysis is the grouping of individuals in a population in order to discover structure in the data. In some sense, we would like the individuals within a group to be close or similar to one another, but dissimilar from individuals in other groups.

Clustering is fundamentally a collection of methods of data exploration. One often uses a method to see if natural groupings are present in the data. If groupings do emerge, these may be named and their properties summarised. For example, if the clusters are compact, then it may be sufficient for some purposes to reduce the information on the original dataset to information about a small number of groups, in some cases representing a group of individuals by a single pattern. The results of a cluster analysis may produce identifiable structure that can be used to generate hypotheses (to be tested on a separate dataset) to account for the observed data.

It is difficult to give a universal definition of the term 'cluster'. All of the methods described in this chapter can produce a *partition* of the dataset – a division of the dataset into mutually nonoverlapping groups. However, different methods will often yield different groupings since each implicitly imposes a structure on the data. Also, the techniques will produce groupings even when there is no 'natural' grouping in the data. The term 'dissection' is used when the data consist of a single homogeneous population that one wishes to partition. Clustering techniques may be used to obtain dissections, but the user must be aware that a structure is being imposed on the data that may not be present. This does not matter in some applications.

Statistical Pattern Recognition, Third Edition. Andrew R. Webb and Keith D. Copsey.
© 2011 John Wiley & Sons, Ltd. Published 2011 by John Wiley & Sons, Ltd.

Before attempting a classification, it is important to understand the problem you are wishing to address. Different classifications, with consequently different interpretations, can be imposed on a sample and the choice of variables is very important. For example, there are different ways in which books may be grouped on your bookshelf – by subject matter or by size – and different classifications will result from the use of different variables. Each classification may be important in different circumstances, depending on the problem under consideration. Once you understand your problem and data, you must choose your method carefully. An inappropriate match of method to data can give results that are misleading.

There is a vast literature on clustering. Some of the more useful texts are given in Section 11.11. There is a wide range of application areas, sometimes with conflicting terminology. This has led to methods being rediscovered in different fields of study. Much of the early work was in the fields of biology and zoology, but clustering methods have also been applied in the fields of psychology, archaeology, linguistics and signal processing.

The topics discussed in this chapter are:

1. *Hierarchical methods*: methods that derive a clustering from a given dissimilarity matrix.
2. *Quick partitions*: methods for obtaining a partition as an initialisation to more elaborate approaches.
3. *Mixture models*: models that express the probability density function as a sum of component densities.
4. *Sum-of-squares methods*: methods that minimise a sum-of-squares error criterion, including k-means, fuzzy k-means, vector quantisation and stochastic vector quantisation.
5. *Spectral clustering*: methods that use the eigenvectors of a graph Laplacian to embed the data into a space that captures the underlying structure.
6. *Cluster validity*: addressing the problem of model selection.

11.2 Hierarchical methods

Hierarchical clustering procedures are the most commonly used means for summarising data structure. A hierarchical tree is a nested set of partitions represented by a tree diagram or *dendrogram* (Figure 11.1). Sectioning a tree at a particular level produces a partition into g disjoint groups. If two groups are chosen from different partitions (the results of partitioning at different levels) then either the groups are disjoint or one group wholly contains the other.

An example of a hierarchical classification is the classification of the animal kingdom. Each species belongs to a series of nested clusters of increasing size with a decreasing number of common characteristics. In producing a tree diagram like that in Figure 11.1, it is necessary to order the points so that branches do not cross. This ordering is somewhat arbitrary, but does not alter the structure of the tree, only its appearance. There is a numerical value associated with each position up the tree where branches join. This is a measure of the distance or dissimilarity between two merged clusters. There are many different measures of distances between clusters and these give rise to different hierarchical structures, as we shall see in

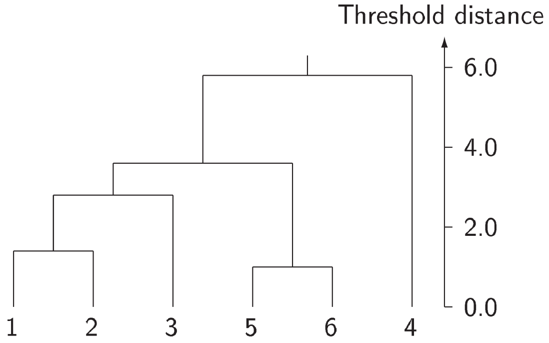


Figure 11.1 Dendrogram.

later sections of this chapter. Sectioning a tree partitions the data into a number of clusters of comparable homogeneity (as measured by the clustering criterion).

There are several different algorithms for finding a hierarchical tree. An *agglomerative algorithm* begins with n subclusters, each containing a single data point and at each stage merges the two most similar groups to form a new cluster, thus reducing the number of clusters by one. The algorithm proceeds until all the data fall within a single cluster. A *divisive algorithm* operates by successively splitting groups beginning with a single group and continuing until there are n groups, each of a single individual. Generally, divisive algorithms are computationally inefficient (except where most of the variables are binary attribute variables).

From the tree diagram, a new set of distances between individuals may be defined with the distance between individual i and individual j being the distance between the two groups that contain them, when these two groups are amalgamated (i.e. the distance level of the lowest link joining them). Thus, the procedure for finding a tree diagram may be viewed as a transformation of the original set of dissimilarities d_{ij} to a new set $\hat{d}_{ij}$ where $\hat{d}_{ij}$ satisfy the *ultrametric inequality*

$$\hat{d}_{ij} \leq \max(\hat{d}_{ik}, \hat{d}_{jk}) \qquad \text{for all objects } i, j, k$$

This means that the distances between three groups can be used to define a triangle that is either equilateral or isosceles (either the three distances are the same or two are equal and the third smaller – see Figure 11.1, for example). A transformation $D : d \to \hat{d}$ is termed an *ultrametric transformation*. All of the methods in this section produce a clustering from a given dissimilarity matrix.

11.2.1 Single-link method

The single-link method is one of the oldest methods of cluster analysis. It is defined as follows. Two objects a and b belong to the same single-link cluster at level d if there exists a chain of intermediate objects $i_1, \ldots, i_{m-1}$ linking them such that all the distances

$$d_{i_k, i_{k+1}} \leq d \qquad \text{for } k = 0, \ldots, m-1$$

where $i_0 = \mathrm{a}$ and $i_m = \mathrm{b}$. The single-link groups for the data of Figure 11.1 for a threshold of $d = 2.0$, 3.0 and 5.0 are $\{(1, 2), (5, 6), (3), (4)\}$, $\{(1, 2, 3), (5, 6), (4)\}$ and $\{(1, 2, 3, 5, 6), (4)\}$.

We shall illustrate the method by example with an agglomerative algorithm in which, at each stage of the algorithm, the closest two groups are fused to form a new group where the distance between two groups, A and B, is the distance between their closest members, i.e.

$$d_{AB} = \min_{i \in A,\, j \in B} d_{ij} \tag{11.1}$$

Consider the dissimilarity matrix for each pair of objects in a set comprising six individuals:

	1	2	3	4	5	6
1	0	4	13	24	12	8
2		0	10	22	11	10
3			0	7	3	9
4				0	6	18
5					0	8.5
6						0

The closest two groups (which contain a single object each at this stage) are those containing the individuals 3 and 5. These are fused to form a new group $\{3, 5\}$ and the distances between this new group and the remaining groups calculated according to Equation (11.1) so that $d_{1,(3,5)} = \min\{d_{13}, d_{15}\} = 12$; $d_{2,(3,5)} = \min\{d_{23}, d_{25}\} = 10$, $d_{4,(3,5)} = 6$, $d_{6,(3,5)} = 8.5$ giving the new dissimilarity matrix:

	1	2	(3, 5)	4	6
1	0	4	12	24	8
2		0	10	22	10
(3, 5)			0	6	8.5
4				0	18
6					0

The closest two groups now are those containing objects 1 and 2; therefore these are fused to form a new group (1, 2). We now have four clusters (1, 2), (3, 5), 4 and 6. The distance between the new group and the other three clusters is calculated: $d_{(1,2)(3,5)} = \min\{d_{13}, d_{23}, d_{15}, d_{25}\} = 10$; $d_{(1,2)4} = \min\{d_{14}, d_{24}\} = 22$; $d_{(1,2)6} = \min\{d_{16}, d_{26}\} = 8$. The new dissimilarity matrix is:

	(1, 2)	(3, 5)	4	6
(1, 2)	0	10	22	8
(3, 5)		0	6	8.5
4			0	18
6				0

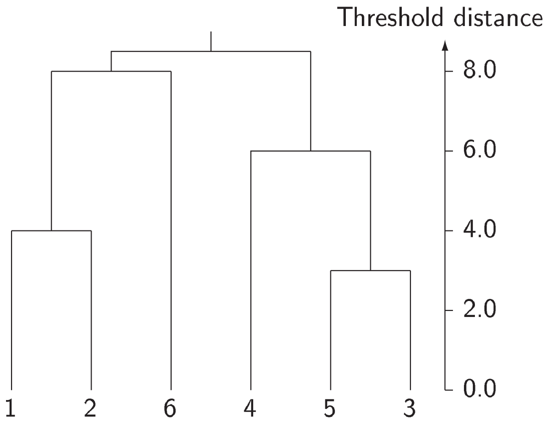


Figure 11.2 Single-link dendrogram.

The closest two groups are now those containing 4 and (3, 5). These are fused to form (3, 4, 5) and a new dissimilarity matrix calculated. This is shown with the result of fusing the next two groups:

	(1, 2)	(3, 4, 5)	6
(1, 2)	0	10	8
(3, 4, 5)		0	8.5
6			0

	(1, 2, 6)	(3, 4, 5)
(1, 2, 6)	0	8.5
(3, 4, 5)		0

The single-link dendrogram is given in Figure 11.2.

The above agglomerative algorithm for a single-link method illustrates the fact that it takes only a single link to join two distinct groups and that the distance between two groups is the distance of their closest neighbours. Hence the alternative name of *nearest-neighbour method*. A consequence of this joining together by a single link is that some groups can become elongated with some distant points, having little in common, being grouped together because there is a chain of intermediate objects. This drawback of *chaining* is illustrated in Figures 11.3 and 11.4.

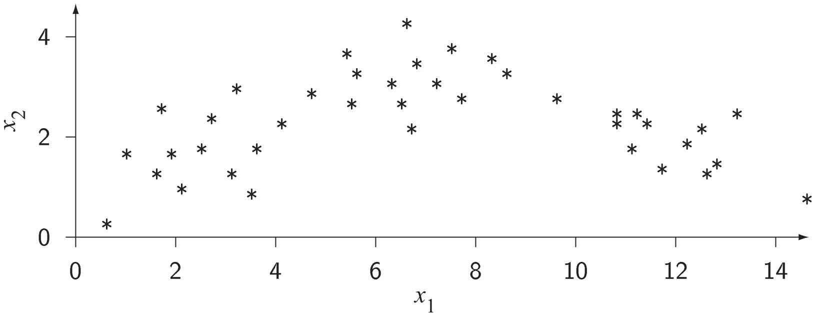


Figure 11.3 Illustration of chaining with the single-link method.

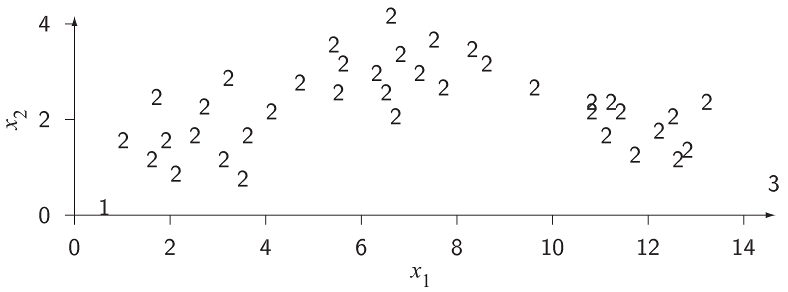


Figure 11.4 Single-link three-group solution for the data of Figure 11.3.

Figure 11.3 shows a distribution of data samples. Figure 11.4 shows the single-link three-group solution for the data in Figure 11.3. These groups do not correspond to those suggested by the data in Figure 11.3.

There are many algorithms for finding a single-link tree. Some are agglomerative, like the procedure described above, some are divisive; some are based on an ultrametric transformation and others generate the single-link tree via the minimum spanning tree[1] [see Rohlf (1982) for a review of algorithms]. The algorithms vary in their computational efficiency, storage requirements and ease of implementation. Sibson's (1973) algorithm uses the property that only local changes in the reduced dissimilarity result when two clusters are merged and it has been extended to the complete-link method discussed in the following section. It has computational requirements $\mathcal{O}(n^2)$, for n objects. More time-efficient algorithms are possible if knowledge of the metric properties of the space in which the data lie is taken into account. In such circumstances, it is not necessary to compute all dissimilarity coefficients. Also, preprocessing the data to facilitate searches for nearest neighbours can reduce computational complexity.

11.2.2 Complete-link method

In the *complete-link* or *furthest-neighbour method* the distance between two groups A and B is the distance between the two furthest points, one taken from each group

$$d_{AB} = \max_{i \in A, j \in B} d_{ij}$$

In the example used to illustrate the single-link method, the second stage dissimilarity matrix (after merging the closest groups 3 and 5 using the complete-link rule above) becomes:

	1	2	(3, 5)	4	6
1	0	4	13	24	8
2		0	11	22	10
(3, 5)			0	7	9
4				0	18
6					0

The final complete-link dendrogram is shown in Figure 11.5.

[1] The minimum spanning tree is the tree connecting all data points such that the sum of the distances between pairs of points joined by a branch of the tree is a minimum.

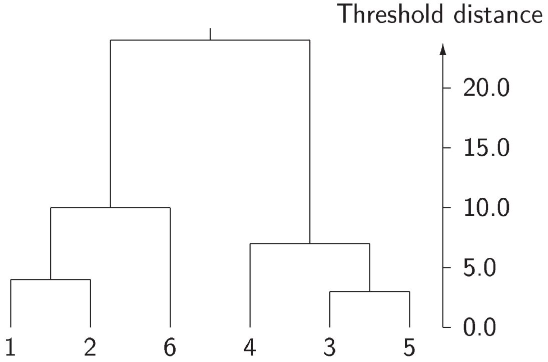


Figure 11.5 Complete-link dendrogram.

At each stage, the closest groups are merged of course. The difference between this method and the single-link method is the measure of distance between groups. The groups found by sectioning the complete-link dendrogram at level h have the property that $d_{ij} < h$ for all members in the group. The method concentrates on the internal *cohesion* of groups in contrast to the single-link method, which seeks isolated groups. Sectioning a single-link dendrogram at a level h gives groups with the property that they are separated from each other by at least a 'distance' h.

Defays (1977) provides an algorithm for the complete-link method using the same representation as Sibson. It should be noted that the algorithm is sensitive to the ordering of the data, and consequently has several solutions. Thus it provides only an approximate complete link clustering.

11.2.3 Sum-of-squares method

The sum-of-squares method is appropriate for the clustering of points in Euclidean space. The aim is to minimise the total within-group sum of squares. *Ward's hierarchical clustering method* (Ward, 1963) uses an agglomerative algorithm to produce a set of hierarchically nested partitions that can be represented by a dendrogram. However, the *optimal* sum-of-squares partitions for different numbers of groups are not necessarily hierarchically nested. Thus Ward's algorithm is suboptimal.

At each stage of the algorithm, the two groups that produce the smallest increase in the total within-group sum of squares are amalgamated. The dissimilarity between two groups is defined to be the increase in the total sum of squares that would result if they were amalgamated. The updating formula for the dissimilarity matrix is

$$d_{i+j,k} = \frac{n_k + n_i}{n_k + n_i + n_j} d_{ik} + \frac{n_k + n_j}{n_k + n_i + n_j} d_{jk} - \frac{n_k}{n_k + n_i + n_j} d_{ij}$$

where $d_{i+j,k}$ is the distance between the amalgamated groups $i + j$ and the group k and n_i is the number of objects in group i. Initially, each group contains a single object and the element of the dissimilarity matrix, d_{ij}, is the squared Euclidean distance between the ith and the jth object (Everitt *et al.*, 2011).

11.2.4 General agglomerative algorithm

Many agglomerative algorithms for producing hierarchical trees can be expressed as a special case of a single algorithm. The algorithms differ in the way that the dissimilarity matrix is updated. The Lance–Williams recurrence formula expresses the dissimilarity between a cluster k and the cluster formed by joining i and j as

$$d_{i+j,k} = a_i d_{ik} + a_j d_{jk} + b d_{ij} + c|d_{ik} - d_{jk}|$$

where a_i, b and c are parameters that, if chosen appropriately, will give an agglomerative algorithm for implementing some of the more commonly used methods (Table 11.1).

Centroid distance
This defines the distance between two clusters to be the distance between the cluster means or (centroids).

Median distance
When a small cluster is joined to a larger one, the centroid of the result will be close to the centroid of the larger cluster. For some problems this may be a disadvantage. This measure attempts to overcome this by defining the distance between two clusters to be the distance between the medians of the clusters.

Group average link
In the group average method, the distance between two clusters is defined to be the average of the dissimilarities between all pairs of individuals, one from each group

$$d_{AB} = \frac{1}{n_i n_j} \sum_{i \in A, j \in B} d_{ij}$$

11.2.5 Properties of a hierarchical classification

What desirable properties should a hierarchical clustering method possess? It is difficult to write down a set of properties on which everyone will agree. What might be a set of

Table 11.1 Special cases of the general agglomerative algorithm.

	a_i	b	c
Single link	$\frac{1}{2}$	0	$-\frac{1}{2}$
Complete link	$\frac{1}{2}$	0	$\frac{1}{2}$
Centroid	$\frac{n_i}{n_i+n_j}$	$-\frac{n_i n_j}{(n_i+n_j)^2}$	0
Median	$\frac{1}{2}$	$-\frac{1}{4}$	0
Group average link	$\frac{n_i}{n_i+n_j}$	0	0
Ward's method	$\frac{n_i+n_k}{n_i+n_j+n_k}$	$-\frac{n_k}{n_i+n_j+n_k}$	0

commonsense properties to one person may be the extreme position of another. Jardine and Sibson (1971) suggest a set of six mathematical conditions that an ultrametric transformation should satisfy; for example, that the results of the method should not depend on the labelling of the data. They show that the single-link method is the only one to satisfy all their conditions and recommend this method of clustering. However, this method has its drawbacks (as do all methods) which has led people to question the plausibility of the set of conditions proposed by Jardine and Sibson. We shall not list the conditions here but refer to Jardine and Sibson (1971) and Williams *et al*. (1971) for further discussion.

11.2.6 Example application study

The problem

This application involves the analysis of microarray gene expression data of *Toxoplasma gondii* (Gautam *et al*., 2010). *Toxoplasma gondii* is a parasite that can infect a wide range of warm-blooded animals, including humans, and develop into a chronic infection that cannot be eliminated by currently used drugs. The aim of the study is to find out new possible drug targets (key molecules) through microarray data analysis.

The data

Microarray data were provided by the Stanford Microarray Database. The data were filtered to 327 genes for analysis.

The model

The assumption is that if the expression of one gene is similar to another, then it is likely that they are related in function. Thus, a method for grouping expression data is implemented. A hierarchical clustering method was implemented (the complete-link method) and results compared with *k*-means clustering (Section 11.5).

Results

For the *Toxoplasma gondii* dataset, five main clusters were identified using the complete-link method. One particular cluster was identified as important for further investigation since it included surface antigens (SAGs), known to be important. Most of the SAG gene family members present in this cluster were also present in a single cluster produced by the *k*-means clustering algorithm. The study identified some possible drug targets for the treatment of toxoplasmosis, since the genes have an important role in the immune system.

11.2.7 Summary

The concept of having a hierarchy of nested clusters was developed primarily in the biological field and may be inappropriate to model the structure in some data. Each hierarchical method imposes its own structure on the data. The single-link method seeks isolated clusters, but is generally not favoured, even though it is the only one satisfying the conditions proposed by Jardine and Sibson. It is subject to the chaining effect, which can result in long straggly groups. This may be useful in some circumstances if the clusters sought are not homogeneous, but it can mean that distinct groups are not resolved because of intermediate points present between the groups. The group average, complete-link and Ward's method tend to concentrate on internal cohesion, producing homogeneous, compact (often spherical) groups.

The centroid and median methods may lead to *inversions* (a reduction in the dissimilarity between an object and a cluster when the cluster increases in size) which make the dendrogram difficult to interpret. Also, ties in the dissimilarities may lead to multiple solutions (*nonuniqueness*) of which the user should be aware (Morgan and Ray, 1995).

Hierarchical agglomerative methods are one of the most common clustering techniques employed. Divisive algorithms are less popular, but efficient algorithms have been proposed, based on recursive partitioning of the cluster with largest diameter (Guénoche *et al*., 1991).

11.3 Quick partitions

Many of the techniques described subsequently in this chapter are implemented by algorithms that require an initial partition of the data. The normal mixture methods require initial estimates of means and covariance matrices. These could be sample-based estimates derived from an initial partition. The k-means algorithm also requires an initial set of means. Hierarchical vector quantisation (see Section 11.5.3) requires initial estimates of the code vectors. In the context of discrimination, radial basis functions, introduced in Chapter 4, require initial estimates for 'centres'. These could be derived using the quick partition methods of this section or be the result of a more principled clustering approach (which in turn may need to be initialised).

Let us suppose that we have a set of n data samples and we wish to find an initial partition into k groups, or to find k seed vectors. We can always find a seed vector, given a group of objects, by taking the group mean. Also, we can partition a set, given k vectors, using a nearest-neighbour assignment rule. There are many heuristic partition methods. We shall consider a few of them.

Random k selection

We wish to have k different vectors, so we select one randomly from the whole dataset, the another from the remaining $n - 1$ samples in the dataset, and so on. In a supervised classification problem, these vectors should be spread across all classes ideally.

Variable division

Choose a single variable. This may be selected from one of the measured variables or be a linear combination of variables; for example, the first principal component. Divide it into k equal intervals that span the range of the variable. The data are partitioned according to which bin they fall in and k seed vectors are found from the means of each group.

Leader algorithm

The leader cluster algorithm (Hartigan, 1975; Späth, 1980) partitions a dataset such that for each group there is a leader object and all other objects within the group are within a distance T of the leading example.

Figure 11.6 illustrates a partition in two dimensions. The first data point, A, is taken as the centre of the first group. Successive data points are examined. If they fall inside the circle centred at A of radius T then they are assigned to group 1. The first data sample examined to fall outside the circle, say at B, is taken as the leader of the second group. Further data points are examined to see if they fall within the first two clusters. The first one to fall outside, say at C, is taken as the centre of the third cluster and so on.

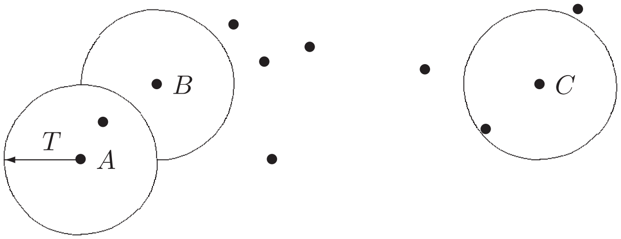


Figure 11.6 Leader clustering.

Points to note about the algorithm:

1. All cluster centres are at least a distance T from each other.
2. It is fast, requiring only one pass through the dataset.
3. It can be applied to a given dissimilarity matrix.
4. It is dependent on the ordering of the dataset. The first point is always a cluster leader. Also, initial clusters tend to be larger than later ones.
5. The distance T is specified, not the number of clusters.

11.4 Mixture models

11.4.1 Model description

In the mixture method of clustering, each different group in the population is assumed to be described by a different probability distribution. These different probability distributions may belong to the same family but differ in the values they take for the parameters of the distribution. Alternatively, the population may comprise sums of different component densities (for modelling different effects such as a signal and noise). The population is said to be described by a *finite mixture distribution* of the form

$$p(\boldsymbol{x}) = \sum_{i=1}^{g} \pi_i p(\boldsymbol{x}; \boldsymbol{\theta}_i)$$

where π_i are the mixing proportions ($\sum_{i=1}^{g} \pi_i = 1$) and $p(\boldsymbol{x}; \boldsymbol{\theta}_i)$ is a p-dimensional probability function depending on a parameter vector $\boldsymbol{\theta}_i$. There are three sets of parameters to estimate: the values of π_i, the components of the vectors $\boldsymbol{\theta}_i$ and the value of g, the number of groups in the population.

Many forms of mixture distributions have been considered and there are many methods for estimating their parameters. An example of a mixture distribution for continuous variables is the mixture of normal distributions

$$p(\boldsymbol{x}) = \sum_{i=1}^{g} \pi_i p(\boldsymbol{x}; \boldsymbol{\Sigma}_i, \boldsymbol{\mu}_i)$$

where $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the means and covariance matrices of a multivariate normal distribution (see Section 2.3.1). A mixture for binary variables is

$$p(\boldsymbol{x}) = \sum_{i=1}^{g} \pi_i p(\boldsymbol{x}; \boldsymbol{\theta}_i)$$

where

$$p(\boldsymbol{x}; \boldsymbol{\theta}_j) = \prod_{l=1}^{p} \theta_{jl}^{x_l} (1 - \theta_{jl})^{1-x_l}$$

is the multivariate Bernoulli density. The value of θ_{jl} is the probability that variable l in the jth group is unity.

Maximum likelihood procedures for estimating the parameters of normal mixture distributions were given in Chapter 2. Other examples of continuous and discrete mixture distributions, and methods of parameter estimation, can be found in Everitt and Hand (1981) and Titterington *et al.* 1985). Also, in some applications, variables are often of a mixed type – both continuous and discrete.

The usual approach to clustering using finite mixture distributions is first of all to specify the form of the component distributions, $p(\boldsymbol{x}, \boldsymbol{\theta}_i)$. Then the number of clusters, g, is prescribed. The parameters of the model are now estimated and the objects are grouped on the basis of their estimated posterior probabilities of group membership; that is the object $\boldsymbol{x}$ is assigned to group i if

$$\pi_i p(\boldsymbol{x}; \boldsymbol{\theta}_i) \geq \pi_j p(\boldsymbol{x}; \boldsymbol{\theta}_j) \qquad \text{for all } j \neq i,\ j = 1, \ldots, g$$

Clustering using a normal mixture model may be achieved by using the EM algorithm described in Chapter 2, to which we refer for further details.

The main difficulty with the method of mixtures concerns the number of components, g (see Chapter 2). This is the question of model selection we return to many times in this book. Many algorithms require g to be specified before the remaining parameters can be estimated. Several test statistics have been put forward. Many apply to special cases such as assessing the question as to whether or not the data come from a single component distribution or a two-component mixture. However, others have been proposed based on likelihood ratio tests (Everitt and Hand, 1981, chapter 5; Titterington *et al.*, 1985, chapter 5).

Another problem with a mixture model approach is that there may be many local minima of the likelihood function and several initial configurations may have to be tried before a satisfactory clustering is produced. In any case, it is worthwhile trying several initialisations, since agreement between the resulting classifications lends more weight to the chosen solution.

11.4.2 Example application study

The problem

This application considers clustering of gene expression data for gene function discovery (Dai *et al.*, 2009). The assumption is that genes having similar expression patterns should have similar cellular functions. Therefore, a clustering approach is developed to cluster gene profiles.

Summary
The study develops a model for data arising from multiple sources. It is applied to real data providing biologically plausible results.

The data
Real and simulated datasets were used to assess the approach. The real data comprised mouse protein DNA binding data and gene expression data. There were 1775 genes in both datasets, which were filtered to provide 673 genes for analysis.

The model
An approach based on mixture models is developed. The mixture components are modelled as a product of two independent terms. That is, if $\boldsymbol{x} = [\boldsymbol{y}^T, \boldsymbol{z}^T]^T$, then

$$p(\boldsymbol{x}) = p(\boldsymbol{y})p(\boldsymbol{z})$$

and $p(\boldsymbol{y})$ is a Gaussian distribution and $p(\boldsymbol{z})$ is a beta distribution.

The motivation for this is that the observation $\boldsymbol{x}$ arises from two data sources, each generated from a different probability distribution.

Training procedure
The model parameters were determined by maximising the likelihood using the EM algorithm (see Chapter 2). Three forms of EM algorithm are considered: the standard EM algorithm, an approximated EM and a hybrid EM. The latter two approaches were developed to reduce the computational requirement.

Four model selection criteria (to determine the number of mixture components) were assessed: Akaike Information Criterion (AIC); a modified AIC (AIC3); the Bayesian Information Criterion (BIC); and the integrated classification likelihood-BIC (ICL-BIC).

Results
The approach yielded more biologically plausible results, independent of the model selection criteria, than using simpler mixture models. Three important groups of genes were identified.

11.5 Sum-of-squares methods

Sum-of-squares methods find a partition of the data that maximises a predefined clustering criterion based on the within-class and between-class scatter matrices. The methods differ in the choice of clustering criterion optimised and the optimisation procedure adopted. However, the problem all methods seek to solve is: *Given a set of n data samples, partition the data into g clusters so that the clustering criterion is optimised*.

Most methods are suboptimal. Computational requirements prohibit optimal schemes, even for moderate values of n. Therefore we require methods that, although producing a suboptimal partition, give a value of the clustering criterion that is not much worse than the optimal one. First of all, let us consider the various criteria that have been proposed.

11.5.1 Clustering criteria

Let the n data samples be $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. The sample covariance matrix, $\hat{\boldsymbol{\Sigma}}$, is given by

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - \boldsymbol{m})(\boldsymbol{x}_i - \boldsymbol{m})^T$$

where $\boldsymbol{m} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i$, the sample mean. Let there be g clusters. The *within-class scatter matrix* or *pooled within-group scatter matrix* is

$$\boldsymbol{S}_W = \frac{1}{n}\sum_{j=1}^{g}\sum_{i=1}^{n} z_{ji}(\boldsymbol{x}_i - \boldsymbol{m}_j)(\boldsymbol{x}_i - \boldsymbol{m}_j)^T,$$

the sum of the sums of squares and cross-products (scatter) matrices over the g groups, where $z_{ji} = 1$ if $\boldsymbol{x}_i \in$ group j, 0 otherwise, $\boldsymbol{m}_j = \frac{1}{n_j}\sum_{i=1}^{n} z_{ji}\boldsymbol{x}_i$ is the mean of cluster j and $n_j = \sum_{i=1}^{n} z_{ji}$, the number in cluster j. The between-class scatter matrix is

$$\boldsymbol{S}_B = \hat{\boldsymbol{\Sigma}} - \boldsymbol{S}_W = \sum_{j=1}^{g}\frac{n_j}{n}(\boldsymbol{m}_j - \boldsymbol{m})(\boldsymbol{m}_j - \boldsymbol{m})^T$$

and describes the scatter of the cluster means about the total mean.

The most popular optimisation criteria are based on univariate functions of the above matrices and are similar to the criteria given in Chapter 10 on feature selection and extraction. The two areas of clustering and feature selection are very much related. In clustering we are seeking clusters that are internally cohesive but isolated from other clusters. We do not know the number of clusters. In feature selection or extraction, we have labelled data from a known number of groups or classes and we seek a transformation that makes the classes distinct. Therefore one that transforms the data into isolated clusters will achieve this.

1. Minimisation of Tr$(\boldsymbol{S}_W)$

The trace of $\boldsymbol{S}_W$ is the sum of the diagonal elements

$$\begin{aligned}\mathrm{Tr}(\boldsymbol{S}_W) &= \frac{1}{n}\sum_{j=1}^{g}\sum_{i=1}^{n} z_{ij}|\boldsymbol{x}_i - \boldsymbol{m}_j|^2 \\ &= \frac{1}{n}\sum_{j=1}^{g} S_j\end{aligned}$$

where $S_j = \sum_{i=1}^{n} z_{ji}|\boldsymbol{x}_i - \boldsymbol{m}_j|^2$, the within-group sum of squares for group j. Thus, the minimisation of Tr$(\boldsymbol{S}_W)$ is equivalent to minimising the total within-group sum of squares about the g centroids. Clustering methods that minimise this quantity are sometimes referred to as sum-of-squares or *minimum-variance* methods. They tend to produce clusters that are hyperspherical in shape. The criterion is not invariant to the scale of the axes and usually some form of standardisation of the data must be performed prior to application of the method. Alternatively, criteria that are not invariant to linear transformations of the data may be employed.

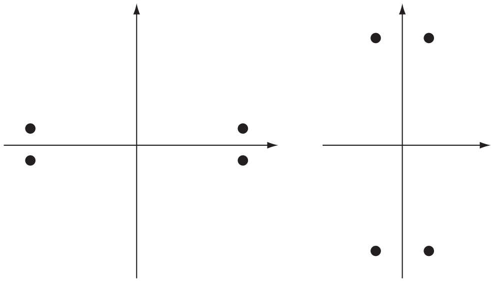


Figure 11.7 Example of effects of scaling in clustering.

2. Minimisation of $|S_W|/|\hat{\Sigma}|$
This criterion is invariant to nonsingular linear transformations of the data. For a given dataset, it is equivalent to finding the partition of the data that minimises $|S_W|$ (the matrix $\hat{\mathbf{\Sigma}}$ is independent of the partition).

3. Maximisation of $\mathrm{Tr}(S_W^{-1}S_B)$
This is a generalisation of the sum-of-squares method in that the clusters are no longer hyperspherical, but hyperellipsoidal. It is equivalent to minimising the sum of squares under the Mahalanobis metric. It is also invariant to nonsingular transformations of the data.

4. Minimisation of $\mathrm{Tr}(\hat{\Sigma}^{-1}S_W)$
This is identical to minimising the sum of squares for data that have been normalised to make the total scatter matrix equal to the identity.

Note that the two examples in Figure 11.7 would be clustered differently by the sum-of-squares method (criterion 1 above). However, since they only differ from each other by a linear transformation, they must both be local optima of a criterion invariant to linear transformations. Thus, it is not necessarily an advantage to use a method that is invariant to linear transformations of the data since structure may be lost. The final solution will depend very much on the initial assignment of points to clusters.

11.5.2 Clustering algorithms

The problem we are addressing is one in combinatorial optimisation. We seek a nontrivial partition of n objects into g groups for which the chosen criterion is optimised. However, to find the optimum partition requires the examination of every possible partition. The number of nontrivial partitions of n objects into g groups is

$$\frac{1}{g!}\sum_{i=1}^{g}(-1)^{g-i}\binom{g}{i}i^n$$

with the final term in the summation being most significant if $n \gg g$. This increases rapidly with the number of objects. For example, there are $2^{59} - 1 \approx 6 \times 10^{17}$ partitions of 60 objects into two groups. This makes exhaustive enumeration of all possible subsets infeasible. In fact, even the branch and bound procedure described in Chapter 10 is impractical for moderate values of n. Therefore, suboptimal solutions must be derived.

We now describe some of the more popular approaches. Many of the procedures require initial partitions of the data, from which group means may be calculated, or initial estimates of group means (from which an initial partition may be deduced using a nearest-class-mean rule). These were discussed in Section 11.3.

11.5.2.1 *k*-means

The aim of the *k-means* (which also goes by the names of the *c-means* or *iterative relocation* or *basic ISODATA*) algorithm is to partition the data into k clusters so that the within-group sum of squares (criterion 1) is minimised. The simplest form of the k-means algorithm is based on alternating two procedures. The first is one of assignment of objects to groups. An object is usually assigned to the group to whose mean it is closest in the Euclidean sense. The second procedure is the calculation of new group means based on the assignments. The process terminates when no movement of an object to another group will reduce the within-group sum of squares. Let us illustrate with a very simple example.

Consider the two-dimensional data shown in Figure 11.8. Let us set $k = 2$ and choose two vectors from the dataset as initial cluster mean vectors. Those selected are points 5 and 6. We now cycle through the dataset and allocate individuals to groups A and B represented by the initial vectors 5 and 6 respectively. Individuals 1, 2, 3, 4 and 5 are allocated to A and individual 6 to B. New means are calculated and the within-group sum of squares is evaluated, giving 6.4. The results of this iteration are summarised in Table 11.2.

The process is now repeated, using the new mean vectors as the reference vectors. This time, individuals 1, 2, 3, and 4 are allocated to group A and 5 and 6 to group B. The within-group sum-of-squares has now decreased to 4.0. A third iteration produces no change in the within-group sum-of-squares.

The iterative procedure of allocating objects to groups on a nearest-group-mean basis, followed by recalculation of group means, gives the version of the k-means called HMEANS by Späth (1980). It is also termed Forgy's method or the basic ISODATA method.

There are two main problems with HMEANS. It may lead to empty groups and it may lead to a partition for which the sum-square error could be reduced by moving an individual from one group to another. Thus the partition of the data by HMEANS is not necessarily one

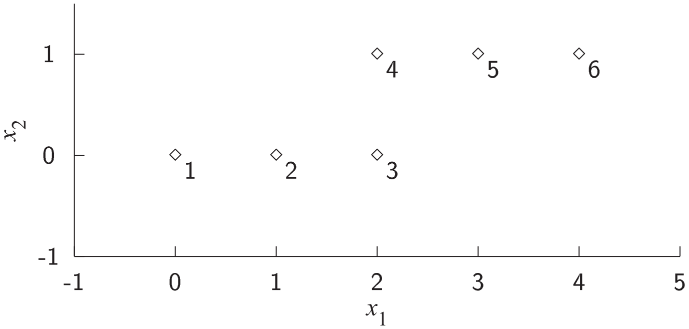


Figure 11.8 Data to illustrate the k-means procedure.

Table 11.2 Summary of k-means iterations.

	Group A		Group B		
Step	Membership	Means	Membership	Means	Tr($\boldsymbol{W}$)
1	1, 2, 3, 4, 5	(1.6, 0.4)	6	(4.0, 1.0)	6.4
2	1, 2, 3, 4	(1.25, 0.25)	5, 6	(3.5, 1.0)	4.0
3	1, 2, 3, 4	(1.25, 0.25)	5, 6	(3.5, 1.0)	4.0

for which the within-group sum of squares is a minimum [see Selim and Ismail (1984a) for a treatment of the convergence of this algorithm].

For example, in Figure 11.9 four data points and two groups are illustrated. The means are at positions (1.0, 0.0) and (3.0, 1.0), with a sum-square error of 4.0. Repeated iterations of the algorithm HMEANS will not alter that allocation. However, if we allocate object 2 to the group containing objects 3 and 4, the means are now at (0.0, 0.0) and (8/3, 2/3), and the sum-square error is reduced to 10/3. This suggests an iterative procedure that cycles through the data points and allocates each to a group for which the within-group sum of squares is reduced the most. Allocation takes place on a sample-by-sample basis, rather than after a pass through the entire dataset. An individual $\boldsymbol{x}_i$ (in group l) is assigned to group r if

$$\frac{n_l}{n_l - 1} d_{il}^2 > \frac{n_r}{n_r + 1} d_{ir}^2$$

where d_{il} is the distance to the lth centroid and n_l is the number in group l. The greatest decrease in the sum-square error is achieved by choosing the group for which $n_r d_{ir}^2/(n_r + 1)$ is a minimum. This is the basis of the k-means algorithm.

There are many variants of the k-means algorithm to improve efficiency of the algorithm in terms of computing time and of achieving smaller error. Some algorithms allow new clusters to be created and existing ones deleted during the iterations. Others may move an object to another cluster on the basis of the best improvement in the objective function. Alternatively, the first encountered improvement during the pass through the dataset could be used.

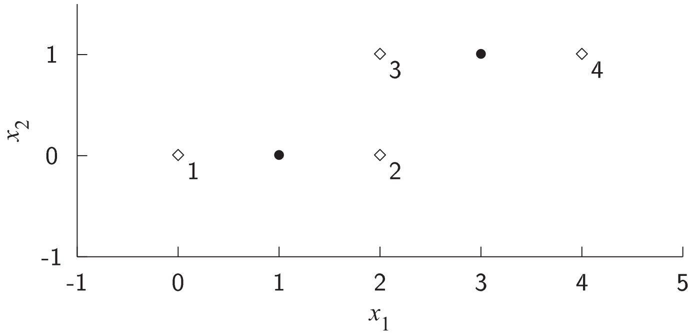


Figure 11.9 Data to illustrate the k-means local optimum.

11.5.2.2 Nonlinear optimisation

The within-groups sum-of-squares criterion may be written in the form

$$\mathrm{Tr}(\boldsymbol{S}_W) = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{g} z_{ki}\sum_{j=1}^{p}(x_{ij} - m_{kj})^2 \tag{11.2}$$

where x_{ij} is the jth coordinate of the ith point ($i = 1,\ldots, n$; $j = 1,\ldots, p$); m_{kj} is the jth coordinate of the mean of the kth group and $z_{ki} = 1$ if the ith point belongs to the kth group and 0 otherwise. The mean quantities m_{kj} may be written as

$$m_{kj} = \frac{\sum_{i=1}^{n} z_{ki}x_{ij}}{\sum_{i=1}^{n} z_{ki}} \tag{11.3}$$

To obtain an optimal partition, we must find the values of z_{ki} (either 0 or 1) for which the expression (11.2) is a minimum.

The approach of Gordon and Henderson (1977) is to regard the $g \times n$ matrix $\boldsymbol{Z}$ with (ij) element z_{ij} as consisting of real-valued quantities (as opposed to binary quantities) with the property

$$\sum_{k=1}^{g} z_{ki} = 1 \qquad \text{and} \qquad z_{ki} \geq 0 \qquad (i = 1,\ldots, n; k = 1,\ldots, g) \tag{11.4}$$

Minimisation of the expression (11.2) with respect to z_{ki}, ($i = 1,\ldots, n$; $k = 1,\ldots, g$), subject to the constraints above, yields a final solution for $\boldsymbol{Z}$ with elements that are all 0 or 1 (even though they are not constrained to be binary-valued). Therefore we can obtain a partition by minimising (11.2) subject to the constraints (11.4) and assigning objects to groups on the basis of the values z_{ik}. Thus, m_{kj} is not equal to a group mean until the iteration has converged.

The problem can be transformed to one of unconstrained optimisation by writing z_{ji} as

$$z_{ji} = \frac{\exp(\nu_{ji})}{\sum_{k=1}^{g}\exp(\nu_{ki})} \quad (j = 1,\ldots, g; i = 1,\ldots, n)$$

that is, we regard $\mathrm{Tr}\{\boldsymbol{S}_W\}$ as a nonlinear function of parameters ν_{ki}, $i = 1,\ldots, n$; $k = 1,\ldots, g$, and seek a minimum of $\mathrm{Tr}\{\boldsymbol{S}_W(\nu)\}$. Other forms of transformation to unconstrained optimisation are possible. However, for the particular form given above, the gradient of $\mathrm{Tr}\{\boldsymbol{S}_W(\nu)\}$ with respect to ν_{ab} has the simple form

$$\frac{\partial\mathrm{Tr}\{\boldsymbol{S}_W(\nu)\}}{\partial\nu_{ab}} = \frac{1}{n}\sum_{k=1}^{g} z_{kb}(\delta_{ka} - z_{ab})|\boldsymbol{x}_b - \boldsymbol{m}_k|^2 \tag{11.5}$$

where $\delta_{ka} = 0$, $k \neq a$ and 1 otherwise. There are many nonlinear optimisation schemes that can be used. The parameters ν_{ij} must be given initial values. Gordon and Henderson (1977) suggest choosing an initial set of random values z_{ji} uniformly distributed in the range $[1, 1 + a]$ and scaled so that their sum is unity. A value of about 2 is suggested for the parameter a. Then ν_{ji} is initialised to $\nu_{ji} = \log(z_{ji})$.

11.5.2.3 Fuzzy *k*-means

The partitioning methods described so far in this chapter have the property that each object belongs to one group only, though the mixture model can be regarded as providing degrees of cluster membership. Indeed, the early work on fuzzy clustering was closely related to multivariate mixture models. The basic idea of the fuzzy clustering method is that patterns are allowed to belong to all clusters with different degrees of membership. The first generalisation of the k-means algorithm was presented by Dunn (1974). The *fuzzy k-means* (or fuzzy c-means) algorithm attempts to find a solution for parameters y_{ji} ($i = 1, \ldots, n; j = 1, \ldots, g$) for which

$$J_r = \sum_{i=1}^{n} \sum_{j=1}^{g} y_{ji}^r |\boldsymbol{x}_i - \boldsymbol{m}_j|^2 \tag{11.6}$$

is minimised subject to the constraints

$$\sum_{j=1}^{g} y_{ij} = 1 \quad 1 \le i \le n$$

$$y_{ji} \ge 0 \quad i = 1, \ldots, n; \quad j = 1, \ldots, g$$

The parameter y_{ji} represents the *degree of association* or *membership function* of the ith pattern or object with the jth group. In the above expression (11.6), r is a scalar termed the *weighting exponent* which controls the 'fuzziness' of the resulting clusters ($r \ge 1$) and $\boldsymbol{m}_j$ is the 'centroid' of the jth group

$$\boldsymbol{m}_j = \frac{\sum_{i=1}^{n} y_{ji}^r \boldsymbol{x}_i}{\sum_{i=1}^{n} y_{ji}^r} \tag{11.7}$$

A value of $r = 1$ gives the same problem as the nonlinear optimisation scheme presented earlier. In that case, we know that a minimum of (11.6) gives values for the y_{ji} that are either 0 or 1.

The basic algorithm is iterative and can be stated as follows (Bezdek, 1981):

1. Select r ($1 < r < \infty$); initialise the membership function values y_{ji}, $i = 1, \ldots, n; j = 1, \ldots, g$.
2. Compute the cluster centres $\boldsymbol{m}_j$, $j = 1, \ldots, g$, according to (11.7).
3. Compute the distances d_{ij}, $i = 1, \ldots, n; j = 1, \ldots, g$, where $d_{ij} = |\boldsymbol{x}_i - \boldsymbol{m}_j|$.
4. Compute the membership function according to

   If $d_{il} = 0$ for some l, $y_{li} = 1$, and $y_{ji} = 0$, for all $j \ne l$

   $$\text{otherwise } y_{ji} = \frac{1}{\sum_{k=1}^{g} \left(\frac{d_{ij}}{d_{ik}}\right)^{\frac{2}{r-1}}}$$

5. If not converged, go to step 2.

As $r \rightarrow 1$, this algorithm tends to the basic k-means algorithm. Improvements to this basic algorithm, resulting in faster convergence, are described by Kamel and Selim (1994).

Several stopping rules have been proposed (Ismail, 1988). One is to terminate the algorithm when the relative change in the centroid values becomes small; that is, terminate when

$$D_z \triangleq \left\{ \sum_{j=1}^{g} |\boldsymbol{m}_j(k) - \boldsymbol{m}_j(k-1)|^2 \right\}^{\frac{1}{2}} < \epsilon$$

where $\boldsymbol{m}_j(k)$ is the value of the jth centroid on the kth iteration and ϵ is a user-specified threshold. Alternative stopping rules are based on changes in the membership function values, y_{ji}, or the cost function, J_r. Another condition based on the local optimality of the cost function is given by Selim and Ismail (1986). It is proposed to stop when

$$\max_{1 \le i \le n} \alpha_i < \epsilon$$

where

$$\alpha_i = \max_{1 \le j \le g} y_{ji}^{r-1} |\boldsymbol{x}_i - \boldsymbol{m}_j|^2 - \min_{1 \le j \le g} y_{ji}^{r-1} |\boldsymbol{x}_i - \boldsymbol{m}_j|^2$$

since at a local minimum, $\alpha_i = 0$, $i = 1, \ldots, n$.

11.5.2.4 Complete search

Complete search of the space of partitions of n objects into g groups is impractical for all but very small datasets. The branch and bound method (described in a feature subset selection context in Chapter 10) is one approach for finding the partition that results in the minimum value of the clustering criterion, without exhaustive enumeration. Nevertheless, it may still be impractical. Koontz *et al.* (1975) have developed an approach that extends the range of problems to which branch and bound can be applied. The criterion they seek to minimise is $\mathrm{Tr}\{\boldsymbol{S}_W\}$. Their approach is to divide the dataset into 2^m independent sets. The branch and bound method is applied to each set separately and then sets are combined in pairs (to give 2^{m-1} sets) and the branch and bound method applied to each of these combined sets, using the results obtained from the branch and bound application to the constituent parts. This is continued until the branch and bound procedure is applied to the entire set. This hierarchical approach results in a considerable saving in computer time.

Other approaches based on global optimisation algorithms such as simulated annealing have also been proposed. Simulated annealing is a stochastic relaxation technique in which a randomly selected perturbation to the current configuration is accepted or rejected probabilistically. Selim and Al-Sultan (1991) apply the method to the minimisation of $\mathrm{Tr}\{\boldsymbol{S}_W\}$. Generally, the method is slow, but it can lead to effective solutions.

11.5.3 Vector quantisation

Vector quantisation is not a *method* of producing clusters or partitions of a dataset but rather an *application* of many of the clustering algorithms already presented. Indeed, many clustering techniques have been rediscovered in the vector quantisation literature. On the other hand, there are some important algorithms in the vector quantisation literature that are not found in the standard texts on clustering. This section is included in the section on optimisation methods

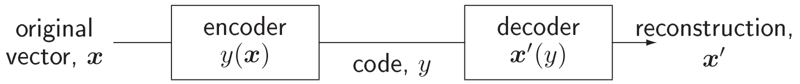


Figure 11.10 The encoding–decoding operation in vector quantisation.

since in vector quantisation a distortion measure (often, but by no means exclusively, based on the Euclidean distance) is optimised during training. A comprehensive and very readable account of the fundamentals of vector quantisation is given by Gersho and Gray (1992).

Vector quantisation is the encoding of a p-dimensional vector $\boldsymbol{x}$ as one from a *codebook* of g vectors, $z_1, \ldots, z_g$, termed the *code vectors* or the *codewords*. The purpose of vector quantisation is primarily to perform data compression. A vector quantiser consists of two components: an *encoder* and a *decoder* (Figure 11.10).

The encoder maps an input vector $\boldsymbol{x}$ to a scalar variable, y, taking discrete values $1, \ldots, g$. After transmission of the *index*, y, the inverse operation of reproducing an approximation $\boldsymbol{x}'$ to the original vector takes place. This is termed decoding and is a mapping from the index set $\mathcal{I} = \{1, \ldots, g\}$ to the codebook $\mathcal{C} = \{z_1, \ldots, z_g\}$. Codebook design is the problem of determining the codebook entries given a set of training samples. From a clustering point of view, we may regard the problem of codebook design as one of clustering the data and then choosing a representative vector for each cluster. These vectors could be cluster means for example and they form the entries in the codebook. They are indexed by integer values. Then the code vector for a given input vector $\boldsymbol{x}$ is the representative vector, say z of the cluster to which $\boldsymbol{x}$ belongs. Membership of a cluster may be determined on a nearest-to-cluster-mean basis. The distortion or error in approximation is then $d(\boldsymbol{x}, z)$, the distance between $\boldsymbol{x}$ and z (Figure 11.11).

The problem in vector quantisation is to find a set of codebook vectors that characterise a dataset. This is achieved by choosing the set of vectors for which a distortion measure between an input vector, $\boldsymbol{x}$, and its quantised vector, $\boldsymbol{x}'$, is minimised. Many distortion measures have been proposed, the most common being based on the square error measure giving average distortion, D_2,

$$\begin{aligned} D_2 &= \int p(\boldsymbol{x}) d(\boldsymbol{x}, \boldsymbol{x}') d\boldsymbol{x} \\ &= \int p(\boldsymbol{x}) \left\| \boldsymbol{x}'(y(\boldsymbol{x})) - \boldsymbol{x} \right\|^2 d\boldsymbol{x} \end{aligned} \tag{11.8}$$

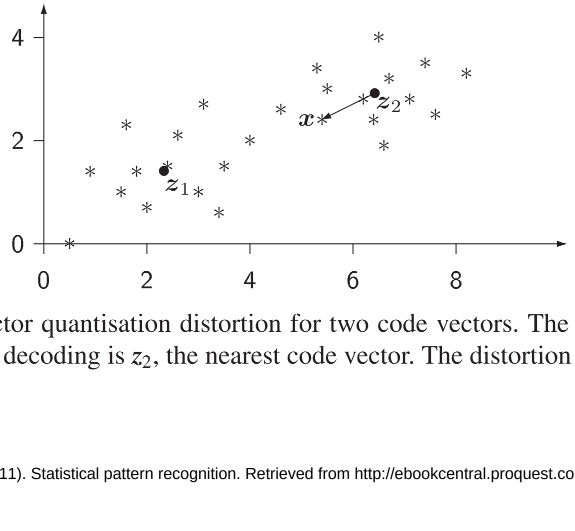


Figure 11.11 Vector quantisation distortion for two code vectors. The reconstruction of $\boldsymbol{x}$ after encoding and decoding is z_2, the nearest code vector. The distortion is $d(\boldsymbol{x}, z_2)$.

Table 11.3 Some distortion measures used in vector quantisation.

Type of norm	$d(\boldsymbol{x}, \boldsymbol{x}')$
L_2, Euclidean	$\lvert\boldsymbol{x}' - \boldsymbol{x}\rvert$
L_ν	$\{\sum_{i=1}^{p} \lvert\boldsymbol{x}' - \boldsymbol{x}\rvert^{\nu}\}^{\frac{1}{\nu}}$
Minkowski	$\max\limits_{1 \leq i \leq p} \lvert x_i' - x_i\rvert$
Quadratic (for positive definite symmetric $\boldsymbol{B}$)	$(\boldsymbol{x}' - \boldsymbol{x})^T \boldsymbol{B}(\boldsymbol{x}' - \boldsymbol{x})$

where $p(\boldsymbol{x})$ is the probability density function over samples $\boldsymbol{x}$ used to train the vector quantiser and $\|.\|$ denotes the norm of a vector. Other distortion measures are given in Table 11.3.

For a finite number of training samples, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, we may write the distortion as

$$D = \sum_{j=1}^{g} \sum_{\boldsymbol{x}_i \in S_j} d(\boldsymbol{x}_i, \boldsymbol{z}_j)$$

where S_j is the set of training vectors for which $y(\boldsymbol{x}) = j$, i.e. those that map onto the jth code vector, $\boldsymbol{z}_j$. For a given set of code vectors $\boldsymbol{z}_j$, the partition that minimises the average distortion is constructed by mapping each $\boldsymbol{x}_i$ to the $\boldsymbol{z}_j$ for which $d(\boldsymbol{x}_i, \boldsymbol{z}_j)$ is a minimum over all $\boldsymbol{z}_j$ – i.e. choosing the minimum distortion or nearest-neighbour code vector. Alternatively, for a given partition the code vector of a set S_j, $\boldsymbol{z}_j$, is defined to be the vector for which

$$\sum_{\boldsymbol{x}_i \in S_j} d(\boldsymbol{u}, \boldsymbol{x}_i)$$

is a minimum with respect to $\boldsymbol{u}$. This vector is called the centroid (for the square error measure it is the mean of the vectors $\boldsymbol{x}_i$).

This suggests an iterative algorithm for a vector quantiser:

1. Initialise the code vectors.
2. Given a set of code vectors, determine the minimum distortion partition.
3. Find the optimal set of code vectors for a given partition.
4. If the algorithm has not converged, then go to step 2.

This is clearly a variant of the k-means algorithms given earlier. It is identical to the basic k-means algorithm provided that the distortion measure used is the square error distortion since all the training vectors are considered at each iteration rather than making an adjustment of code vectors by considering each in turn. This is known as the *generalised Lloyd algorithm* in the vector quantisation literature (Gersho and Gray, 1992) or the LBG algorithm in the data compression literature. One of the main differences between the LBG algorithm and some of the k-means implementations is the method of initialisation of the centroid vectors. The LBG algorithm (Linde *et al.*, 1980) given below starts with a one-level quantiser (a single cluster) and after obtaining a solution for the code vector $\boldsymbol{z}$, 'splits' the vector $\boldsymbol{z}$ into two close vectors

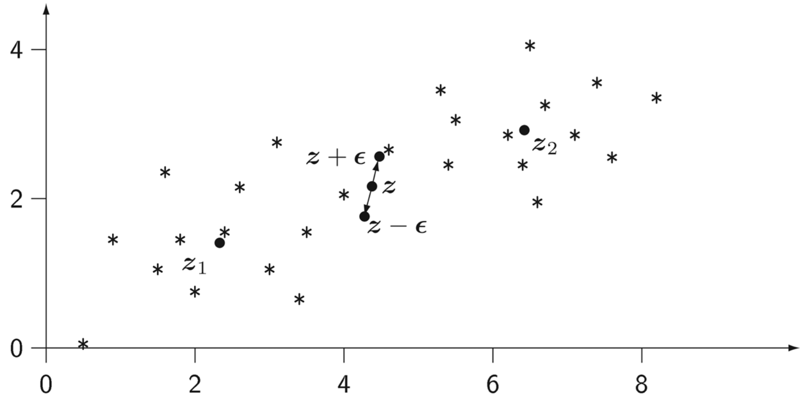


Figure 11.12 LBG algorithm illustration. z denotes the group centroid; z_1 and z_2 denote the code vectors for a two-level quantiser.

that are used as seed vectors for a two-level quantiser. This is run until convergence and a solution is obtained for the two-level quantiser. Then these two codewords are split to give four seed vectors for a four-level quantiser. The process is repeated so that finally quantisers of $1, 2, 4, \ldots, N$ levels are obtained (Figure 11.12). The steps in the algorithm are:

1. Initialise a code vector z_1 to be the group mean; initialise $\boldsymbol{\epsilon}$.
2. Given a set of m code vectors, 'split' each vector z_i to form $2m$ vectors, $z_i + \boldsymbol{\epsilon}$ and $z_i - \boldsymbol{\epsilon}$. Set $m = 2m$; relabel the code vectors as $\boldsymbol{x}'_i$, $i = 1, \ldots, m$.
3. Given the set of code vectors, determine the minimum distortion partition.
4. Find the optimal set of code vectors for a given partition.
5. Repeat steps 3 and 4 until convergence.
6. If $m \neq N$, the desired number of levels, go to step 2.

Although it appears that all we have achieved with the introduction of vector quantisation in this chapter is yet another version of the k-means algorithm, the vector quantisation framework allows us to introduce two important concepts: that of tree-structured codebook search that reduces the search complexity in vector quantisation and that of topographic mappings in which a topology is imposed on the code vectors.

11.5.3.1 Tree-structured vector quantisation

Tree-structured vector quantisation is a way of structuring the codebook in order to reduce the amount of computation required in the encoding operation. It is a special case of the classification trees or decision trees discussed in a discrimination context in Chapter 7. Here we shall consider *fixed-rate* coding, in which there are the same number of bits used to represent each code vector. *Variable-rate coding*, which allows pruning of the tree, will not be addressed. Pruning methods for classification trees are described in Chapter 7, and in the vector quantisation context by Gersho and Gray (1992).

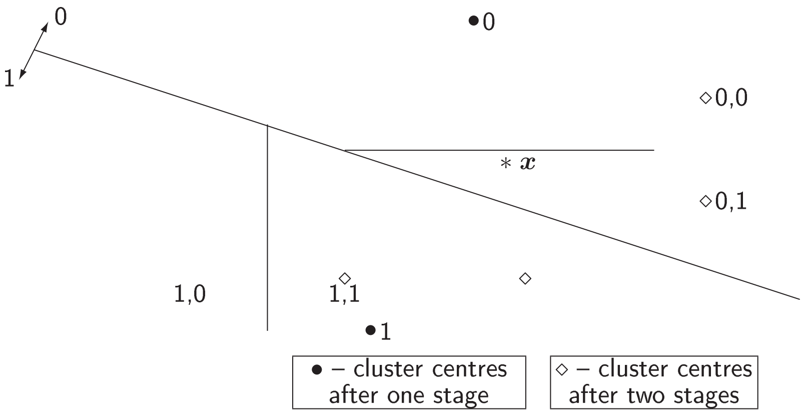

0
1
●0
◇0,0
∗ x
◇0,1
1,0
◇
1,1
◇
●1
● – cluster centres after one stage
◇ – cluster centres after two stages

Figure 11.13 Tree-structured vector quantisation.

We shall begin our description of tree-structured vector quantisation with a simple *binary tree* example. The first stage in the design procedure is to run the k-means algorithm on the entire dataset to partition the set into two parts. This leads to two code vectors (the means of each cluster) (Figure 11.13).

Each group is considered in turn and the k-means algorithm applied to each group, partitioning each group into two parts again. This second stage then produces four code vectors and four associated clusters. The mth stage produces 2^m code vectors. The *total* number of code vectors produced in an m-stage design algorithm is $\sum_{i=1}^{m} 2^i = 2^{m+1} - 2$. This process produces a hierarchical clustering in which two clusters are disjoint or one wholly contains the other.

Encoding of a given vector, $\boldsymbol{x}$, proceeds by starting at the root of the tree (labelled A_0 in Figure 11.14) and comparing $\boldsymbol{x}$ with each of the two level 1 code vectors, identifying the nearest.

We then proceed along the branch to A_1 and compare the vector $\boldsymbol{x}$ with the two code vectors at this level which were generated from members of the training set in this group. Thus there are m comparisons in an m-stage encoder. This compares with 2^m code vectors at the final level. Tree-structured vector quantisation may not be optimal in the sense that the

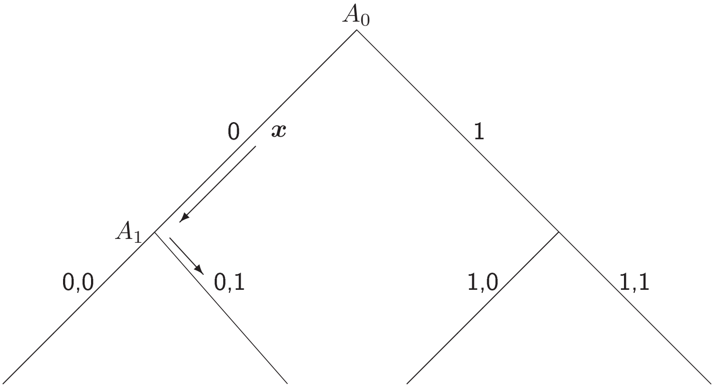


Figure 11.14 Tree-structured vector quantisation tree.

nearest neighbour of the final level code vectors is not necessarily found (the final partition in Figure 11.13 does not consist of nearest-neighbour regions). However, the code has the property that it is a progressively closer approximation as it is generated and the method can lead to a considerable saving in encoding time.

11.5.3.2 Self-organising feature maps

Self-organising feature maps are a special kind of vector quantisation in which there is an ordering or topology imposed on the code vectors. The aim of self-organisation is to represent high-dimensional data as a low-dimensional array of numbers (usually a one- or two-dimensional array) that captures the structure in the original data. Distinct clusters of data points in the data space will map to distinct clusters of code vectors in the array, although the converse is not necessarily true: separated clusters in the array do not necessarily imply separated clusters of data points in the original data. In some ways, self-organising feature maps may be regarded as a method of exploratory data analysis in keeping with those described in Chapter 10. The basic algorithm has the k-means algorithm as a special case.

Figures 11.15 and 11.16 illustrate the results of the self-organising feature map algorithm applied to data in two dimensions.

- In Figure 11.15, 50 data samples are distributed in three groups in two dimensions and we have used a self-organisation process to obtain a set of nine ordered cluster centres in one dimension. By a set of ordered cluster centres we mean that centre z_i is close in some sense to z_{i-1} and z_{i+1}. In the k-means algorithm, the order that the centres are stored in the computer is quite arbitrary and depends on the initialisation of the procedure.
- In Figure 11.16, the data (not shown) comprise 500 samples drawn from a uniform distribution over a square ($[-1 \leq x_1, x_2 \leq 1]$) and do not lie on (or close to) a one-dimensional manifold in the two-dimensional space. Again, we have imposed a one-dimensional topology on the cluster centres, which are joined by straight lines. In this case, we have obtained a space-filling curve.

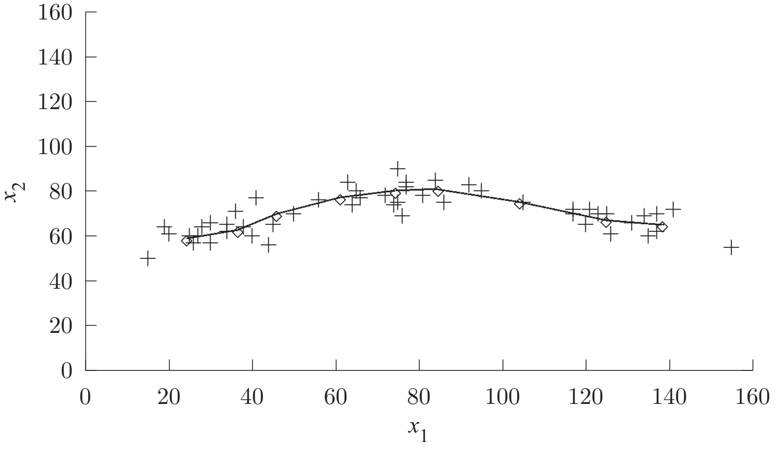


Figure 11.15 Topographic mapping. Adjacent cluster centres (◇) in the stored array of code vectors are joined.

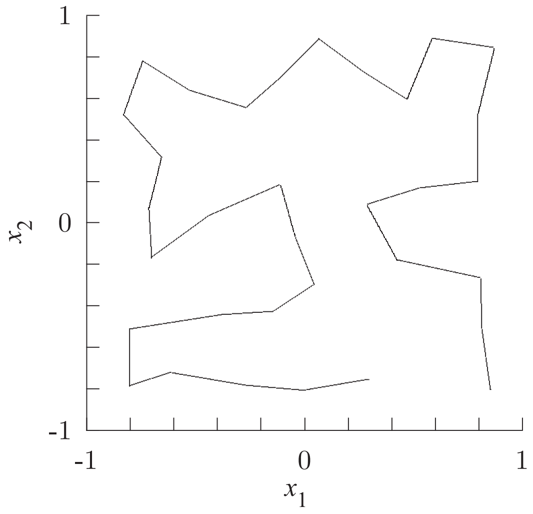


Figure 11.16 Topographic mapping for data uniformly distributed over a square. Thirty-three centres are determined and adjacent cluster centres in the stored array are joined.

In Figures 11.15 and 11.16, a transformation to a reduced dimension is achieved using a topographic mapping in which there is an ordering on the cluster centres. Each point in the data space is mapped to the ordered index of its nearest cluster centre. The mapping is nonlinear, and for purposes of illustration we considered mappings to one dimension only. If the data do lie on a reduced-dimension manifold within a high-dimensional space, then it is possible for topographic mappings to capture the structure in the data and present it in a form that may aid interpretation. In a supervised classification problem, it is possible to label each cluster centre with a class label according to the majority of the objects for which that cluster centre is the nearest. Of course we can do this even if there were no ordering on the cluster centres, but the ordering does allow the relationships between classes (according to decision boundaries) to be viewed easily.

The algorithm for determining the cluster centres may take many forms. The basic approach is to cycle through the dataset and adjust the cluster centres in the neighbourhood (suitably defined) of each data point. The algorithm is often presented as a function of time, where time refers to the number of presentations of a data sample. One algorithm is as follows:

1. Decide on the topology of the cluster centres (code vectors). Initialise α, the neighbourhood and the cluster centres $z_1, \dots, z_N$.
2. Repeat until convergence:
   (a) Select a data sample $\boldsymbol{x}$ (one of the training samples), find the closest centre: let $d_{j^*} = \min_j (d_j)$ where

$$d_j = |\boldsymbol{x} - z_j| \quad j = 1, \dots, N$$

(b) Update the code vectors in the neighbourhood, $\mathcal{N}_{j^*}$ of code vector z_{j^*}

$$z(t+1) = z(t) + \alpha(t)(\boldsymbol{x}(t) - z(t)) \quad \text{for all centres } z \in \mathcal{N}_{j^*}$$

where α is a learning rate that decreases with iteration number, t $(0 \leq \alpha \leq 1)$.

(c) Decrease the neighbourhood and the learning parameter, α.

In order to apply the algorithm, an initial set of code vectors, the learning rate $\alpha(t)$ and the change with t of the neighbourhoods must be chosen.

Definition of topology

The choice of topology of the cluster centres requires some prior knowledge of the data structure. For example, if you suspect circular topology in your data, then the topology of your cluster centres should reflect this. Alternatively, if you wish to map your data onto a two-dimensional surface, then a regular lattice structure for the code vectors may be sufficient.

Learning rate

The learning rate, α, is a slowly decreasing function of t. It is suggested by Kohonen (1989) that it could be a linear function of t, stopping when α reaches 0, but there are no hard and fast rules for choosing $\alpha(t)$. It could be linear, inversely proportional to t or exponential. Haykin (1994) describes two phases: the *ordering* phase, of about 1000 iterations, when α starts close to unity and decreases, but remains above 0.1; and the *convergence* phase when α decreases further and is maintained at a small value – 0.01 or less – for typically thousands of iterations.

Initialisation of code vectors

Code vectors z_i are initialised to $\boldsymbol{m} + \boldsymbol{\epsilon}_i$ where $\boldsymbol{m}$ is the sample mean and $\boldsymbol{\epsilon}_i$ is a vector of small random values.

Decreasing the neighbourhood

The topological neighbourhood $\mathcal{N}_j$ of a code vector z_j is itself a function of t and decreases as the number of iterations proceeds. Initially, the neighbourhood may cover most of the code vectors ($z_{j-r}, \ldots, z_{j-1}, z_{j+1}, \ldots, z_{j+r}$ for large r), but towards the end of the iterations, it covers the nearest (topological) neighbours, z_{j-1} and z_{j+1} only. Finally it shrinks to zero. The problem is how to initialise the neighbourhood and how to reduce it as a function of t. During the *ordering* phase, the neighbourhood is decreased to cover only a few neighbours.

An alternative approach proposed by Luttrell (1989) is to fix the neighbourhood size and to start off with a few code vectors. The algorithm is run until convergence, and then the number of vectors is increased by adding vectors intermediate to those already calculated. The process is repeated and continued until a mapping of the desired size has been grown. Although the neighbourhood size is fixed, it starts off by covering a large area (since there are few centres) and the physical extent is reduced as the mapping grows. Specifically, given a data sample $\boldsymbol{x}$, if the nearest neighbour is z^*, then all code vectors z in the neighbourhood of z^* are updated according to

$$z \rightarrow z + \pi(z, z^*)(\boldsymbol{x} - z) \tag{11.9}$$

where π (> 0) is a function that depends on the position of z in the neighbourhood of z^*. For example, with a one-dimensional topology, we may take

$$\pi(z, z^*) = \begin{cases} 0.1 & \text{for } z = z^* \\ 0.01 & \text{for } z \text{ a topographic neighbour of } z^* \end{cases}$$

The Luttrell algorithm for a one-dimensional topographic mapping is as follows:

```
1. Initialise two code vectors, $z_1$ and $z_2$; set $m = 2$. Define the neighbourhood function $\pi$;
   set the number of updates per code vector, $u$.
2. Repeat until the distortion is small enough or the maximum number of code vectors is reached:
   (a) For $j = 1$ to $m \times u$ do
       • Sample from the dataset $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, say $\boldsymbol{x}$.
       • Determine the nearest-neighbour code vector, say $z^*$.
       • Update the code vectors according to: $z \rightarrow z + \pi(z, z^*)(\boldsymbol{x} - z)$.
   (b) Define $2m - 2$ new code vectors ($z_1$ remains unaltered): for $j = m - 1$ down to 1 do
       • $z_{2j+1} = z_{j+1}$
       • $z_{2j} = \frac{z_j + z_{j+1}}{2}$
   (c) Set $m = 2m - 1$.
```

Topographic mappings have received widespread use as a means of exploratory data analysis (Kraaijveld *et al.*, 1992; Roberts and Tarassenko, 1992). They have also been misused and applied when the ordering of the resulting cluster centres is irrelevant in any subsequent data analysis and a simple k-means approach could have been adopted. An assessment of the method and its relationship to other methods of multivariate analysis is provided by Murtagh and Hernández-Pajares (1995). Luttrell (1989) has derived the basic learning algorithm from a vector quantisation approach assuming a minimum distortion (Euclidean) and a robustness to noise on the codes. This puts the approach on a firmer mathematical footing. Also, the requirement for *ordered* cluster centres is demonstrated for a hierarchical vector quantiser.

11.5.3.3 Learning vector quantisation

Vector quantisation or clustering (in the sense of partitioning a dataset, not seeking meaningful groupings of objects) is often performed as a preprocessor to supervised classification. There are several ways in which vector quantisers or self-organising maps have been used with labelled training data. In the radar target classification example of Luttrell (1995), each class is modelled separately using a self-organising map. Test data are classified by comparing each pattern with the prototype patterns in each of the self-organising maps (codebook entries) and classifying on a nearest-neighbour rule basis.

An alternative approach that uses vector quantisers in a supervised way is to model the whole of the training data with a single vector quantiser (rather than each class separately).

Each training pattern is assigned to the nearest code vector, which is then labelled with the class of the majority of the patterns assigned to it. A test pattern is then classified using a nearest-neighbour rule using the labelled codebook entries.

Learning vector quantisation is a supervised generalisation of vector quantisation that takes account of class labels *in the training process*. The basic algorithm is as follows:

1. Initialise cluster centres (or code vectors), $z_1, \dots, z_N$, and labels of cluster centres, $\omega_1, \dots, \omega_N$.
2. Select a sample $\boldsymbol{x}$ from the training dataset with associated class $\omega_{\boldsymbol{x}}$ and find the closest centre: let $d_{j^*} = \min_j (d_j)$ where

$$d_j = |\boldsymbol{x} - z_j| \quad j = 1, \dots, N$$

   with corresponding centre z_{j^*} and class ω_{j^*}.

3. If $\omega_{\boldsymbol{x}} = \omega_{j^*}$ then update the nearest vector, z_{j^*}, according to

$$z_{j^*}(t+1) = z_{j^*}(t) + \alpha(t)(\boldsymbol{x}(t) - z_{j^*}(t))$$

   where $0 < \alpha_t < 1$ and decreases with t, starting at ≈ 0.1.

4. If $\omega_{\boldsymbol{x}} \neq \omega_{j^*}$ then update the nearest vector, z_{j^*}, according to

$$z_{j^*}(t+1) = z_{j^*}(t) - \alpha(t)(\boldsymbol{x}(t) - z_{j^*}(t))$$

5. Go to step 1 and repeat until several passes have been made through the dataset.

Correct classification of a pattern in the dataset leads to a refinement of the codeword in the direction of the pattern. Incorrect classification leads to a movement of the codeword away from the training pattern.

11.5.3.4 Stochastic vector quantisation

Stochastic vector quantisation (Luttrell, 1999) is a generalisation of the standard approach in which an input vector, $\boldsymbol{x}$, is encoded as a *vector* of code indices, $\boldsymbol{y}$, (rather than as a *single* code index) that are stochastically sampled from a probability distribution $p(\boldsymbol{y}|\boldsymbol{x})$ that depends on the input vector, $\boldsymbol{x}$. The decoding operation that produces a reconstruction, $\boldsymbol{x}'$, is also probabilistic, with $\boldsymbol{x}'$ being a sample drawn from $p(\boldsymbol{x}|\boldsymbol{y})$ given by

$$p(\boldsymbol{x}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})}{\int p(\boldsymbol{y}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z}} \tag{11.10}$$

One of the key factors motivating the development of the stochastic vector quantisation approach is that of scalability to high dimensions. A problem with standard vector quantisation is that the codebook grows exponentially in size as the dimensionality of the input vector is increased, assuming that the contribution to the reconstruction error from each dimension is held constant. This means that such vector quantisers are not appropriate for encoding extremely high-dimensional input vectors, such as images. An advantage of using the stochastic approach is that it automates the process of splitting high-dimensional input vectors into low-dimensional blocks before encoding them, because minimising the mean Euclidean reconstruction error can encourage different stochastically sampled code indices to become associated with different input subspaces.

11.5.4 Example application study

The problem

This application involves a study into the detection of breast lesions in magnetic resonance imaging (MRI) imagery and the discrimination of malignant breast lesions from benign lesions (Lee *et al.*, 2007).

Summary

A k-means algorithm (Section 11.5.2) was applied to measured data obtained from a set of MRI images. The approach appeared to be useful, in detecting heterogeneous patterns within a malignant tumour, but failed when malignant lesions presented patterns similar to benign tumours. It was concluded that morphological analysis is required.

The data

The data comprised a set of five MRI breast images from each of 13 patients. For each patient, signal measurements were made on a voxel-by-voxel basis from the malignant tumour area. The total number of patterns was 1735; each pattern was five-dimensional.

The model

A k-means clustering approach was adopted.

Training procedure

Classification was achieved by labelling cluster centroids and a pattern from a random set of images assigned to the nearest reference centroid.

Results

The results showed that it is not possible to discriminate between benign and malignant lesions using the presented approach. There are two possible reasons:

- Further features, for example generated through a morphological analysis, are required.
- A supervised approach is likely to offer improved performance rather than using an unsupervised approach with a postprocessing stage that labels the centroid vectors with class labels.

11.5.5 Further developments

Procedures for reducing the computational load of the k-means algorithm are discussed by Venkateswarlu and Raju (1992). Further developments of k-means procedures to other metric spaces (with l_1 and l_∞ norms) are described by Bobrowski and Bezdek (1991). Juan and Vidal (1994) propose a fast k-means algorithm (based on the approximating and eliminating search algorithm, AESA) for the case when data arise in the form of a dissimilarity. That is, the data cannot be represented in a suitable vector space (without performing a multidimensional scaling procedure), though the dissimilarity between points is available. Termed the *k-centroids* procedure, it determines the ‘most centred sample’ as a centroid of a cluster. Popescu-Borodin (2008) develops a fast k-means algorithm for the quantisation of still images (see also Chen *et al*., 2008).

There have been many developments of the basic fuzzy clustering approach and many algorithms have been proposed. Sequential approaches are described by de Màntaras and Aguilar-Martín (1985). In ‘semi fuzzy’ or ‘soft’ clustering (Selim and Ismail, 1984b; Ismail, 1988) patterns are considered to belong to some, though not necessarily all, clusters. In thresholded fuzzy clustering (Kamel and Selim, 1991) membership values below a threshold are set to zero, with the remaining being normalised and an approach that performs a fuzzy classification with prior assumptions on the number of clusters is reported by Gath and Geva (1989). Developments of the fuzzy clustering approach to data arising in the form of dissimilarity matrices are described by Hathaway and Bezdek (1994).

11.5.6 Summary

The techniques described in this section minimise a square error objective function. The k-means procedure is a special case of all of the techniques. It is widely used in pattern recognition, forming the basis for many supervised classification techniques. It produces a ‘crisp’ coding of the data in that a pattern belongs to one cluster only.

Fuzzy k-means is a development that allows a pattern to belong to more than one cluster. This is controlled by a membership function. The clusters resulting from a fuzzy k-means approach are softly overlapping, in general, with the degree of overlap controlled by a user-specified parameter. The learning procedure determines the partition.

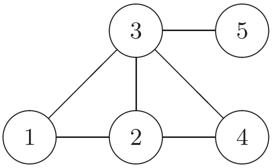
The vector quantisation approaches are application driven. The aim is to produce a crisp coding and algorithms such as tree-structured vector quantisation are motivated by the need for fast coding. Self-organising feature maps produce an ordered coding.

11.6 Spectral clustering

In Section 11.2 we examined approaches to cluster analysis that exploited the structure of a (dis-)similarity matrix to produce a clustering. In Section 11.5, the k-means algorithm and variants for clustering patterns in a high-dimensional space was presented. *Spectral clustering* refers to a class of techniques that brings together these two concepts by exploiting the eigenstructure of a similarity matrix, or other matrices derived from it, to partition a set of patterns into disjoint clusters such that patterns in the same cluster have high similarity and patterns in different clusters have low similarity. It is a heuristic that often works well in practice.

11.6.1 Elementary graph theory

Many complex datasets arise as a result of interactions between entities of various kinds. For example, social networks of interactions between people; networks of computers; online purchasing between people and vendors. The use of pattern recognition techniques to analyse such transactional data is considered further in Chapter 12. Such complex data can be



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

Figure 11.17 Undirected graph illustration with its adjacency matrix.

represented using the mathematical concept of a graph with nodes representing entities of interest (people, computers, shops) and the edges a transaction (email message, purchase, telephone call) between those nodes. Spectral clustering also exploits some results from graph theory and therefore here we introduce some basic graph theory notation.

A graph, G, comprises a set of vertices (or nodes) and edges, $G = \{V, E\}$. The structure of the graph is described by the *adjacency matrix*, A, where (also, see Section 12.2)

$$A_{ij} = 1 \quad \text{if node } i \text{ and node } j \text{ are connected}$$

$$A_{ij} = 0 \quad \text{if node } i \text{ and node } j \text{ are not connected}$$

Figure 11.17 gives an illustration of a graph and its adjacency matrix. The graph may be directed (the edges are directed from one node to an other) or undirected (there is no direction associated with the edge between two nodes). For undirected graphs, the adjacency matrix is symmetrical. In addition, there may be weights associated with the edges in a graph. These weights may represent the degree of interaction between nodes (for example, amount of email traffic between two computers, or level of purchasing from an online shop) and may change with time. In this case, the binary adjacency matrix is replaced by a (time-varying) weighted adjacency matrix.

The graph Laplacian is defined in terms of the adjacency matrix as

$$L = D - A$$

where D is a diagonal matrix, $D = \text{Diag}(d_{1,} \ldots, d_n)$, where

$$d_i = \sum_{j=1}^{n} A_{ij}$$

which, for a binary adjacency matrix, is the degree of vertex i. The (unnormalised graph) Laplacian has the following properties:

1. L is symmetric and positive semi-definite.
2. All eigenvalues are real and non-negative. The smallest eigenvalue is zero. Thus the eigenvalues can be ordered:

$$\lambda_n \geq \lambda_{n-1} \geq \cdots \geq \lambda_1 = 0$$

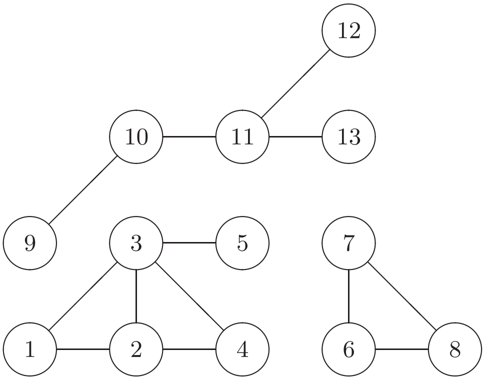


Figure 11.18 Graph with disjointed components.

3. The multiplicity of the eigenvalue with value 0 is equal to the number of connected components of the graph. The corresponding eigenvectors provide indicator vectors to show which component a given node belongs to. For example, for the graph shown in Figure 11.18, the eigenvalue $\lambda = 0$ of the Laplacian has multiplicity 3, indicating three separate components. The corresponding eigenvectors are shown in Table 11.4. The table shows that nodes 1, 2, 3, 4 and 5 form one component; nodes 6, 7 and 8 form a second; and nodes 9, 10, 11, 12 and 13 form a third. Thus, the rows of the eigenvector matrix provide an indication of which group a node belongs to.

This definition extends to a weighted adjacency matrix, for which a normalised graph Laplacian is defined (see Section 11.6.5). The rows of the eigenvectors corresponding to zero eigenvalues can be used to identify disjoint components.

Table 11.4 Eigenvectors v_1, v_2, v_3 corresponding to the eigenvalue $\lambda = 0$ for the graph of Figure 11.18

	v_1	v_2	v_3
1	0	0	1
2	0	0	1
3	0	0	1
4	0	0	1
5	0	0	1
6	0	1	0
7	0	1	0
8	0	1	0
9	1	0	0
10	1	0	0
11	1	0	0
12	1	0	0
13	1	0	0

11.6.2 Similarity matrices

In order to apply the results of graph theory to clustering, we first define the weights on the edge between two nodes to be a measure of similarity between the nodes. This may be defined in several ways (von Luxburg, 2007). We assume that we have a set of patterns, $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, and we wish to define a weighted adjacency matrix, A, where $A_{ij} = s(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and $s(\boldsymbol{x}, \boldsymbol{y})$ is a measure of similarity between patterns $\boldsymbol{x}$ and $\boldsymbol{y}$.

ϵ-neighbourhood

Here, edges are drawn between nodes i and j when the distance between the corresponding patterns, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is less than a specified value, ϵ. Thus

$$s(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & |\boldsymbol{x} - \boldsymbol{y}| < \epsilon \\ 0 & |\boldsymbol{x} - \boldsymbol{y}| \geq \epsilon \end{cases}$$

Fully connected graph

Each node is connected to every other node with an edge weight that is the similarity of the patterns associated with each of the nodes. There are different definitions of a similarity function, $s(\boldsymbol{x}, \boldsymbol{y})$. For example, the Gaussian similarity function for modelling local neighbourhood relationships is given by

$$s(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{|\boldsymbol{x} - \boldsymbol{y}|^2}{2\sigma^2}\right) \tag{11.11}$$

for parameter σ controlling the width of the neighbourhood. The weight on an edge connecting node i to node j is $A_{ij} = s(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

***k*-nearest neighbour**

There are two basic models. The first is the *k-nearest-neighbour graph*:

$$A_{ij} = \begin{cases} s(\boldsymbol{x}_i, \boldsymbol{x}_j) & \boldsymbol{x}_i \text{ is one of the } k \text{ nearest neighbours of } \boldsymbol{x}_j \\ & \text{OR } \boldsymbol{x}_j \text{ is one of the } k \text{ nearest neighbours of } \boldsymbol{x}_I \\ 0 & \text{otherwise} \end{cases}$$

In this model, node i is connected to node j if node j is one of the k-nearest neighbours of node i or node i is one of the k-nearest neighbours of node j. The weight on the edge is the similarity between the patterns associated with the nodes.

The second model, the *mutual k-nearest-neighbour graph* is described by the weighted adjacency matrix:

$$A_{ij} = \begin{cases} s(\boldsymbol{x}_i, \boldsymbol{x}_j) & \boldsymbol{x}_i \text{ is one of the } k \text{ nearest neighbours of } \boldsymbol{x}_j \\ & \text{AND } \boldsymbol{x}_j \text{ is one of the } k \text{ nearest neighbours of } \boldsymbol{x}_I \\ 0 & \text{otherwise} \end{cases}$$

11.6.3 Application to clustering

We have seen that the graph Laplacian can be used to group nodes in a graph through the analysis of the eigenvectors corresponding to the $\lambda = 0$ eigenvalues: nodes in the same graph component are grouped together; nodes in disjoint components are not. This is also the case for a Laplacian constructed from a weighted adjacency matrix.

This suggests an approach to clustering patterns:

1. Construct a matrix of similarities between patterns;
2. Construct the graph Laplacian based on such a matrix of similarities;
3. Analyse the eigenvectors corresponding to the smallest eigenvalues. In general, there will not be completely disjoint components, but components that are 'almost disjoint'. In this case, the smallest eigenvalues will not be zero.

11.6.4 Spectral clustering algorithm

The basic spectral clustering algorithm is given in Algorithm 11.1. Step one constructs a similarity matrix. There are several ways of constructing this given patterns. It may be a simple transformation of a distance between patterns, or one of the measures given in Section 11.6.2. Given this similarity matrix, the graph Laplacian is calculated. Step 3 calculates the eigenvectors and eigenvalues of the Laplacian. Only the eigenvectors corresponding to the k lowest eigenvalues are required. These are then used to define an $n \times k$ data matrix, V. The final step is to use a k-means algorithm (Section 11.5) to apply to the n k-dimensional patterns defined by the rows.

Figure 11.19 shows the results of a spectral clustering applied to some points in two dimensions. A Gaussian similarity function was used [Equation (11.11)]. Points in different clusters are labelled with different symbols.

11.6.5 Forms of graph Laplacian

For a weighted adjacency matrix, a normalised graph Laplacian is used (see Section 12.2). A symmetric form (Ng *et al*., 2002) is given by

$$L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

Algorithm 11.1 Spectral clustering.

1. Form a matrix, S, of similarities, using a definition of similarity between patterns, such as those given in Section 11.6.2.
2. Compute the graph Laplacian from the similarity matrix.
3. Perform an eigendecomposition of the Laplacian, selecting the $n \times k$ matrix of eigenvectors, $V = [v_1, \ldots, v_k]$, corresponding to the k smallest eigenvalues.
4. Perform the k-means algorithm on the n k-dimensional patterns corresponding to the rows of the eigenvector matrix to cluster the patterns into k clusters.

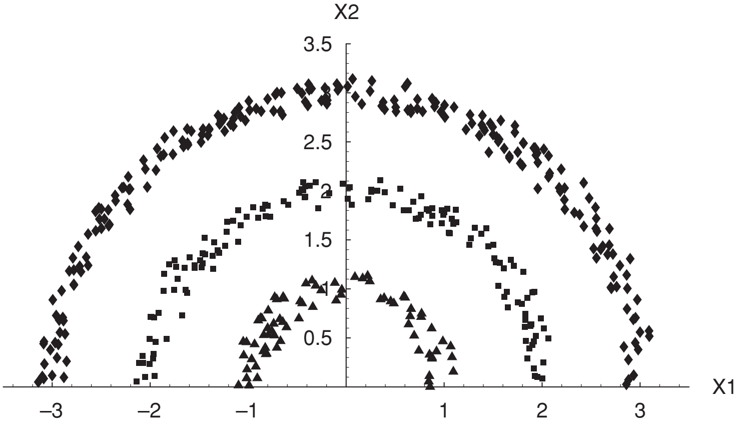


Figure 11.19 Clustering of points in two dimensions: a value of $k = 3$ was taken. Points in different clusters are highlighted by different symbols.

A nonsymmetric form (proposed by Shi and Malik, 2000) is

$$L_{nonsym} = D^{-1}L = I - D^{-1}A$$

These normalised forms have the following properties:

1. The normalised Laplacians are positive semidefinite.
2. All eigenvalues are real and non-negative. The smallest eigenvalue is zero. Thus the eigenvalues can be ordered:

$$\lambda_n \geq \lambda_{n-1} \geq \cdots \geq \lambda_1 = 0$$

3. The multiplicity of the eigenvalue with value 0 is equal to the number of connected components of the graph. For L_{nonsym}, the eigenspace is spanned by eigenvectors of the form $\mathbf{1}_i$, where the vector has a ‘1’ in the jth position if node j is in component i and zero otherwise. The eigenvectors of L_{sym} are $D^{-1}\mathbf{1}_i$.
4. The eigenvectors, v, and eigenvalues, λ, of L_{nonsym} satisfy the generalised symmetric eigenvector equation $Lv = \lambda Dv$.

A clustering algorithm for a normalised graph Laplacian, L_{nonsym}, is given in Algorithm 11.2.

11.6.6 Example application study

The problem

This application involves recovering signals from a linear mixture using partial knowledge of the mixing process and the signals (blind signal separation) using a spectral clustering approach (Bach and Jordan, 2006). This is applied to the problem of one-microphone blind source separation of speech.

Algorithm 11.2 Spectral clustering for normalised Laplacian, L_{nonsym}.

1. Form a matrix, S, of similarities, using a definition of similarity between patterns, such as those given in Section 11.6.2.
2. Compute the graph Laplacian from the similarity matrix.
3. Solve the generalised symmetric eigenvector equation

   $$Lv = \lambda Dv$$

   and select the $n \times k$ matrix of eigenvectors, $V = [v_1, \dots, v_k]$, corresponding to the k smallest eigenvalues.
4. Perform the k-means algorithm on the n k-dimensional patterns corresponding to the rows of the eigenvector matrix to cluster the patterns into k clusters.

Summary

Blind signal separation is a problem that occurs in many fields, such as radar signal processing, speech processing and imaging. An extreme case is when there is only one sensor and two or more signals to estimate. The aim of this study was to separate speech mixtures from a single microphone without specific speaker models. A cost function is derived with the feature that minimisation with respect to the partition of the data leads to a spectral clustering algorithm. Alternatively, for labelled training data (i.e. the partition is known), minimising with respect to the (parameters of a) similarity matrix produces an algorithm for learning the similarity matrix.

The data

The training data comprise speech from four male and female speakers of duration 3 s. For testing, mixes of speakers different from those in the training data were used.

The model

The speech separation problem is formulated as a segmentation problem in the time-frequency plane (speech is represented as a spectrogram). There are many methods for segmenting image-like data, but speech segments have their own characteristics and therefore speech-specific features are derived. These features are used to define parametrised similarity matrices.

Training procedure

There are two stages to deriving a partition of a spectrogram. The first is to use labelled training data from datasets of known partitions to learn the parameters of the similarity matrix. The next stage uses these parameters in an algorithm to cluster previously unseen datasets.

Results

The study showed that a mixture of two English speakers could be successfully demixed, with the segmentation algorithm producing audible signals of reasonable quality.

11.6.7 Further developments

Developments include approaches to address the application of spectral clustering for large datasets. Spectral clustering requires the computation of the eigenvectors of an $n \times n$ matrix, which is an operation of the order of n^3. Therefore, it is infeasible for problems where n is very large. A fast approximate spectral clustering approach is described by Yan *et al*. (2009). An approach for online applications, in which the spectral clustering is updated for small changes in the dataset, is presented by Ning *et al*. (2007). Zare *et al*. (2010) describe a data reduction method, whereby the original data are sampled to produce a smaller set, which is then clustered.

11.6.8 Summary

Spectral clustering uses the eigenvectors of a graph Laplacian for embedding the data into a space that captures the underlying structure. It requires the calculation of the smallest k eigenvectors of an $n \times n$ matrix. The clusters that the approach produces are not always convex sets (as in k-means), but there are several parameters to be set. These include the parameters of the similarity function (ϵ for ϵ-neighbourhood, σ for fully connected graph, k for k nearest neighbour) as well as the form of the graph Laplacian. The normalised form, L_{nonsym} is recommended by von Luxburg (2007).

11.7 Cluster validity

11.7.1 Introduction

Cluster validity is the procedure by which the results of a clustering are evaluated. A clustering algorithm will partition a dataset of objects even if there are no natural clusters within the data. Different clustering methods (or the same method with different parameter settings) may produce very different classifications. How do we know whether the structure is a property of the dataset and not imposed by the particular method that we have chosen? This is the problem of *cluster validity*, which is one that is full of difficulties and rarely straightforward. In some applications of clustering techniques we may not be concerned with groupings in the dataset. For example, in vector quantisation we may be concerned with the average distortion in reconstructing the original data or in the performance of any subsequent analysis technique. This may be measured by the error rate in a discrimination problem or diagnostic performance in image reconstruction (see the examples in the following section). In these situations, clustering is simply a means of obtaining a partition, not to discover structure in the data.

Yet, if we are concerned with discovering groupings within a dataset, how do we validate the clustering? For two-dimensional data, the user can easily verify the result visually. For higher-dimensional data, a simple approach is to view the clustering in a low-dimensional representation of the data. Linear and nonlinear data projection methods have been discussed in Chapter 10. Alternatively, we may perform several analyses using different clustering methods and compare the resulting classifications to see whether the derived structure is an artifact of a particular method. More formal procedures may also be applied and we discuss some approaches in this section.

There are three approaches to cluster validity based on three types of clustering criteria:

Internal criteria
The clustering results are evaluated in terms of quantities that involve the dataset; for example, the dissimilarity matrix.

External criteria
The clustering is evaluated using information that was not used in performing the clustering; for example, a user-specified structure.

Relative criteria
These are criteria to evaluate a clustering by comparing it with that produced by different algorithms (or the same algorithm with different parameters).

The first two cluster validity approaches are based on statistical tests using models for different types of *a priori* structure. The approach based on relative criteria aims to find the best clustering scheme according to a defined metric.

11.7.2 Statistical tests

11.7.2.1 Hypothesis testing

Internal and external criteria approaches are based on statistical hypothesis testing in which the null hypothesis, H_0, concerns the randomness of the data and the test statistic, q, is calculated using the given dataset. In many cases, the probability density function of the test statistic under the given hypothesis is unknown. A sample-based approach is adopted whereby the probability density function is calculated from samples obtained via a Monte Carlo approach.

11.7.2.2 Monte Carlo

For a one-sided test (in which a high value of the test statistic means evidence against the null hypothesis – a right-sided test), we seek the probability that extreme values of the statistic occur by chance under the null hypothesis,

$$Pr(q \geq Q|H_0)$$

where Q is the observed value of the test statistic on the dataset.

In a Monte Carlo approach, sample datasets, X_i, $i = 1, \ldots, r$, are generated and the value of q calculated for each dataset, to produce a set of sample values, $\{q_i, i = 1, \ldots, r\}$. If the significance level is ρ, then the decision is

> Reject the null hypothesis if q is greater than $(1 - \rho)r$ of the values, q_i.

Similarly, for a left-sided test. For a two-sided test, for which the significance level is

$$Pr(|q| \geq Q)$$

then the decision is

> Reject the null hypothesis if q is greater than $(1 - \rho/2)r$ of the values, q_i or if q is less than $(\rho/2)r$ of the values, q_i.

11.7.2.3 Null models

Gordon (1999) identifies five types of null model associated with pattern or dissimilarity matrices for the complete absence of structure in a dataset (see also Gordon 1994b, 1996a).

- *Poisson model* (Bock, 1985) or *random position hypothesis* (Jain and Dubes, 1988). Objects are represented as points uniformly distributed in some region A of the p-dimensional data space. The main problem with this model is the specification of A. Standard definitions include the unit hypercube and the hypersphere.

- *Unimodal model* (Bock, 1985). The variables have a unimodal distribution. The difficulty here is the specification of this density function. Standard definitions include a spherical multivariate normal.

- *Random permutation model*. In this model, the entries in each column of an $n \times p$ data matrix, X, are independently permuted giving $(n!)^{p-1}$ essentially different matrices, each regarded as equally likely.

- *Random dissimilarity matrix* (Ling, 1973) or *random graph hypothesis* (Jain and Dubes, 1988). This is based on data arising in the form of dissimilarities. The elements of the (lower triangle of the) dissimilarity matrix are ranked in random order, all orderings being regarded as equally likely. Viewed as a graph, with the nodes representing patterns and the values on the edges representing dissimilarities between patterns, the edges below a specified threshold are inserted into the graph in random order. One of the problems with this model is that if objects i and j are close (d_{ij} is small), you would expect d_{ik} and d_{jk} to have similar ranks for each object k, but such correlation is not accounted for.

- *Random labels model* (Jain and Dubes, 1988). This assumes that all possible permutations of the labels associated with each pattern (obtained as a result of a clustering) are equally likely. The observed value of the test statistic is compared with the distribution obtained by randomly permuting the labels.

11.7.3 Absence of class structure

Tests for the absence of class structure address the question:

> Are the data $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ sampled from a homogeneous population?

Thus, the null hypothesis is a statement about the randomness of the data. There have been many test statistics proposed (for a review see Gordon, 1998).

For the Poisson model, example test statistics are:

- the number of interpoint distances less than a specified threshold;

- the largest nearest-neighbour distance within a set of objects.

For the random dissimilarity matrix model, example test statistics include:

- the minimum number of edges required to connect a random graph (the number of edges before the graph comprises a single component);

- the number of components in the graph when it contains a specified number of edges.

11.7.4 Validity of individual clusters

Determining whether individual clusters are valid requires the specification of properties of a cluster that an ideal cluster is expected to possess. Such properties of a cluster will be dataset dependent. An alternative approach is to define a statistic and examine its distribution under some null model that the data does not have class structure. This is the approach we present here. The question we address is:

> Is a cluster $\mathcal{C}$ (of size c) defined by
>
> $$\mathcal{C} = \{i : d_{ij} < d_{ik} \text{ for all } j \in \mathcal{C}, k \notin \mathcal{C}\}$$
>
> a valid cluster?

Gordon (1999) describes a Monte Carlo approach to individual cluster validation based on a U statistic:

$$U_{ijkl} = \begin{cases} 0 & \text{if } d_{ij} < d_{kl} \\ \frac{1}{2} & \text{if } d_{ij} = d_{kl} \\ 1 & \text{if } d_{ij} > d_{kl} \end{cases}$$

and

$$U = \sum_{(i,j)\in W} \sum_{(k,l)\in B} U_{ijkl}$$

for subsets W and B of ordered pairs $(i, j) \in W$ and $(k, l) \in B$. W is taken to be those pairs where objects i and j both belong to the cluster $\mathcal{C}$ and B comprises pairs where one element belongs to $\mathcal{C}$ and the other does not.

The basic algorithm is defined as follows:

1. Evaluate U for the cluster $\mathcal{C}$; denote it by U^*.
2. Generate a random $n \times p$ pattern matrix and cluster it using the same algorithm used to produce $\mathcal{C}$.
3. Calculate $U(k)$ for each cluster of size k ($k = 2, \ldots, n - 1$) (arising through the partitioning of a dendrogram, for example). If there is more than one of a given size, select one of that size randomly.
4. Repeat steps 2 and 3 until there are $(m - 1)$ values of $U(k)$ for each value of k.
5. If U^* is less than the jth smallest value of $U(c)$, the null hypothesis of randomness is rejected at the $100(j/m)\%$ level of significance.

Note that the values of $U(k)$ are independent of the dataset.

Gordon takes $m = 100$ and evaluates the above approach under both the Poisson and unimodal (spherical multivariate normal) models. The clusterings using Ward's method are assessed on four datasets. Results for the approach are encouraging. Further refinements could include the use of other test statistics and developments of the null models.

11.7.5 Hierarchical clustering

Here we address the question of the internal validation of a hierarchical clustering:

> Is the hierarchical clustering an accurate summary of the data?

The *cophenetic correlation* coefficient is a measure of the agreement between a set of dissimilarities and a hierarchical classification derived from the data. It is the correlation between the dissimilarities, d_{ij}, and the ultrametric distances, $\hat{d}_{ij}$

$$\frac{\sum_{i<j}\left(d_{ij}-\bar{d}\right)\left(\hat{d}_{ij}-\bar{\hat{d}}\right)}{\left[\sum_{i<j}\left(d_{ij}-\bar{d}\right)^2\sum_{i<j}\left(\hat{d}_{ij}-\bar{\hat{d}}\right)^2\right]^{\frac{1}{2}}}$$

It is a measure of how well the hierarchical classification represented by the dendrogram preserves inter-point distances.

The distribution of this statistic is difficult to calculate and a Monte Carlo approach under a Poisson null model, for example, may be adopted.

11.7.6 Validation of individual clusterings

Here we address the validation of an individual clustering, not a hierarchical clustering, produced by a clustering algorithm. The question we ask is:

> Is the clustering, $\mathcal{C}$, of data into m clusters consistent with information in the dataset X?

Define an $n \times n$ matrix Y:

$$Y_{ij} = \begin{cases} 1 & \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ belong to different clusters} \\ 0 & \text{otherwise} \end{cases}$$

Then, the Γ statistic,

$$\Gamma(D, Y) = \frac{1}{M}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} d_{ij}Y_{ij} \tag{11.12}$$

or the normalised Γ statistic, $\hat{\Gamma}$,

$$\hat{\Gamma}(D, Y) = \frac{1}{M}\frac{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\left(d_{ij}-\mu_D\right)\left(Y_{ij}-\mu_Y\right)}{\sigma_D\sigma_Y} \tag{11.13}$$

is used to measure the agreement between the dissimilarity matrix, D, and the matrix Y, where μ_D and μ_Y are the means and σ_D and σ_Y are the standard deviations of the D and Y matrix elements and $M = N(N-1)/2$.

A Monte Carlo approach under the Poisson null model may be taken. For each dataset generated, the clustering algorithm that was used to produce the clustering $\mathcal{C}$ is applied to the dataset, X, and the statistic calculated. The decision to reject, or not, the null hypothesis at a given significant level, is based on the distribution of the statistics.

11.7.7 Partitions

The questions we seek to address here are those requiring external validation tests and are of the form:

> Does the clustering produced by an algorithm agree with an externally prescribed partition?
>
> Does the proximity matrix of X agree with a predetermined partition?

The approach adopted is the standard one: define statistics that measure the degree to which a partition agrees with either a clustering or a proximity matrix and use a Monte Carlo approach for the estimation of the probability density function of the statistic. Denoting the clustering by $\mathcal{C} = \{C_1, \dots, C_m\}$ and the independent partition by $\mathcal{P} = \{P_1, \dots, P_s\}$ (Theodoridis and Koutroumbas, 2009), and introducing the quantities a, b, c and d as

> a is the number of pairs of pattern vectors that belong to the same cluster in $\mathcal{C}$ and to the same group in $\mathcal{P}$
>
> b is the number of pairs of pattern vectors that belong to the same cluster in $\mathcal{C}$ and to different groups in $\mathcal{P}$
>
> c is the number of pairs of pattern vectors that belong to different clusters in $\mathcal{C}$ and to the same group in $\mathcal{P}$
>
> d is the number of pairs of pattern vectors that belong to different clusters in $\mathcal{C}$ and to different groups in $\mathcal{P}$

then the total number of distinct pairs is $M = a + b + c + d$ and we define the following statistics:

Rand

$$R = (a + d)/M$$

Jaccard

$$J = a/\,(a + b + c)$$

Both statistics have the property that they lie between 0 and 1 and the larger their value then the closer the agreement between $\mathcal{C}$ and $\mathcal{P}$.

The Γ and $\hat{\Gamma}$ statistics may also be used. Defining $X_{ij} = 1$ if the patterns $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are in the same cluster in $\mathcal{C}$ and zero otherwise; and $Y_{ij} = 1$ if the patterns $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are in the same group in $\mathcal{P}$ and zero otherwise, then $\Gamma(X, Y)$ and $\hat{\Gamma}(X, Y)$ have the property that the larger the magnitude the closer the agreement between $\mathcal{C}$ and $\mathcal{P}$.

11.7.8 Relative criteria

Relative criteria are used to compare the results of different clustering algorithms and seek to address the question:

> Which clustering, out of a set produced by applying a clustering algorithm with different parameter values to a dataset X, best fits the data, X?

The basis of the approach is the definition of a validity index.

11.7.8.1 Validity indices

Dunn and Dunn-like indices

The Dunn index is defined as

$$D = \min_{i=1,\ldots,g} \left\{ \min_{j=i+1,\ldots,g} \left(\frac{d\left(C_i, C_j\right)}{\max\limits_{k=1,\ldots,g} \left(\text{diam}\left(C_k\right)\right)} \right) \right\}$$

where $d\left(C_i, C_j\right) = \min\limits_{x \in C_i, y \in C_j} \{d(x, y)\}$ is the distance between two clusters defined as the minimum interpoint distance, with the members of the pair coming from different clusters; and $\text{diam}\,(C_i) = \max\limits_{x, y \in C_i} \{d(x, y)\}$ is the cluster diameter. It is large for compact and well-separated clusters.

The Dunn index can be sensitive to noise (the cluster diameter can be large in a noisy environment). Variations have been proposed using different definitions of cluster distance and cluster diameter that are more robust to noise.

Davies–Bouldin index

The Davies–Bouldin index for a clustering of g clusters is defined as

$$DB = \frac{1}{g} \sum_{i=1}^{g} R_i$$

where

$$R_i = \max_{j=1,\ldots,g,\, j \neq i} \left(R_{ij}\right)$$

and R_{ij} is a measure of within-cluster spread to between-cluster distance given by

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

where d_{ij} is the distance between two clusters, defined as the distance between the cluster centroids, v_i, and s_i is a measure of the spread of cluster C_i

$$d_{ij} = d(v_i, v_j), \qquad s_i = \frac{1}{\|C_i\|} \sum_{x \in C_i} d(x, v_i)$$

and $\|C_i\|$ is the number of patterns in cluster C_i. The Davies–Bouldin index measures average of the similarity between a cluster and its most similar one. Small values of the index are indicative of compact and separated clusters.

Γ statistic

The Γ statistic and its normalised form, $\hat{\Gamma}$, given by Equations (11.12) and (11.13), can be used to measure the relationship between the dissimilarity matrix, D, and the matrix Q, whose (i, j) element is the distance between the cluster centres to which patterns $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong.

$$Q_{ij} = d\left(v_{c_i}, v_{c_i}\right)$$

where c_i is the index of the cluster that contains the pattern $\boldsymbol{x}_i$. Large values are an indication of compact clusters.

Hierarchical clustering indices

In assessing particular hierarchical schemes we must consider how well the structure in the original data can be described by a dendrogram. However, since the structure in the data is not known (this is precisely what we are trying to determine) and since each clustering is simply a method of exploratory data analysis that imposes its own structure on the data, this is a difficult question to address. One approach is to examine various measures of distortion. Many measures have been proposed [see, for example, Cormack (1971) for a summary]. They are based on differences between the dissimilarity matrix d and d^*, the ultrametric dissimilarity coefficient, where $d^*{}_{ij}$ is the distance between the groups containing i and j when the groups are amalgamated. Jardine and Sibson (1971) propose several goodness of fit criteria. One scale-free measure of classifiability is defined by

$$\Delta_1 = \frac{\sum_{i<j} |d_{ij} - d^*_{ij}|}{\sum_{i<j} d_{ij}}$$

Small values of Δ_1 are indicative that the data are amenable to the classification method that produced d^*.

There are many other measures of distortion, both for hierarchical and nonhierarchical schemes. Milligan (1981) performed an extensive Monte Carlo study of thirty internal criterion measures applied to the results of hierarchical clusterings, although the results may also apply to nonhierarchical methods.

11.7.9 Choosing the number of clusters

The problem of deciding how many clusters are present in the data is one common to all clustering methods. If we regard the number of clusters, g, as a model parameter then the cluster validity indices may be used to compare the value of the index function for different values of g. Plotting the validity index against g, there may be a value for g for which a significant local change in the index occurs. When applied to hierarchical schemes, these procedures for determining the number of clusters g are sometimes referred to as *stopping rules*. Many intuitive schemes have been proposed for hierarchical methods; for example, we may look at the plot of fusion level against the number of groups, g (Figure 11.20), and look for a flattening of the curve showing that little improvement in the description of the data structure is to be gained above a particular value of g. Defining $\alpha_j, j = 0, \ldots, n-1$, to be the fusion level corresponding to the stage with $n-j$ clusters, Mojena (1977) proposes a stopping rule that selects the number of groups as g where α_{n-g} is the lowest value of α for which

$$\alpha_{n-g} > \overline{\alpha} + ks_\alpha$$

where $\overline{\alpha}$ and s_α are the mean and the unbiased standard deviation of the fusion levels α; k is a constant, suggested by Mojena to be in the range 2.75–3.5.

Milligan and Cooper (1985) examine thirty procedures applied to classifications of datasets containing 2, 3, 4, or 5 distinct nonoverlapping clusters by four hierarchical schemes. They

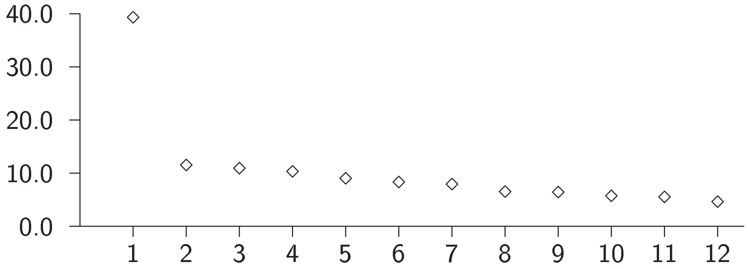


Figure 11.20 Fusion level against the number of clusters.

find that Mojena's rule performs poorly with only two groups present in the data and the best performance is for 3, 4 or 5 groups with a value of k of 1.25. One of the better criteria that Milligan and Cooper (1985) assess is that of Calinski and Harabasz (1974). The number of groups is taken to be the value of g that corresponds to the maximum of C, given by

$$C = \frac{\mathrm{Tr}(\boldsymbol{S}_B)}{\mathrm{Tr}(\boldsymbol{S}_W)}\left(\frac{n-g}{g-1}\right)$$

This is evaluated further by Atlas and Overall (1994) who compare it with a split-sample replication rule of Overall and Magee (1992). This gave improved performance over the Calinski and Harabasz criterion.

Dubes (1987) also reports the results of a Monte Carlo study on the effectiveness of two internal criterion measures in determining the number of clusters. Jain and Moreau (1987) propose a method to estimate the number of clusters in a dataset using the bootstrap technique. A clustering criterion based on the within and between-group scatter matrices (developing a criterion of Davies and Bouldin, 1979) is proposed and the k-means and three hierarchical algorithms are assessed. The basic method of determining the number of clusters using a bootstrap approach can be used with any cluster method.

Several authors have considered the problem of testing for the number of components of a normal mixture (see Chapter 2). Ismail (1988) reports the results of cluster validity studies within the context of soft clustering and lists nine validity functionals that provide useful tools in determining cluster structure (see also Pal and Bezdek, 1995).

It is not reasonable to expect a single statistic to be suitable for all problems in cluster validity. Many different factors are involved and since clustering is essentially a method of exploratory data analysis, we should not put too much emphasis on the results of a single classification, but perform several clusterings using different algorithms and measures of fit.

11.8 Application studies

Applications of hierarchical methods of cluster analysis include the following:

- Flight monitoring. Eddy *et al.* (1996) consider single-link clustering of large datasets (more than 40 000 observations) of high-dimensional data relating to aircraft flights over the USA.

- Clinical data. D'Andrea *et al*. (1994) apply the nearest centroid method to data relating to adult children of alcoholics.
- In a comparative study of seven methods of hierarchical cluster analysis on 20 datasets, Morgan and Ray (1995) examine the extent of inversions in dendrograms and nonuniqueness. They conclude that inversions are expected to be encountered and recommend against the use of the median and centroid methods. Also, nonuniqueness is a real possibility for many datasets.

The k-means clustering approach is widely used as a preprocessor to supervised classification to reduce the number of prototypes:

- Coal petrography. In a study to classify the different constituents (macerals) of coal (Mukherjee *et al*., 1994), the k-means algorithm was applied to training images (training vectors consist of RGB level values) to determine four clusters (identified as one of known types – vitrinite, inertinite, exinite and background). These clusters are labelled and test images are classified using the labelled training vectors.
- Crop classification. Conway *et al*. (1991) use a k-means algorithm to segment synthetic aperture radar images as part of a study into crop classification. k-means is used to identify sets of image regions that share similar attributes prior to labelling. Data were gathered from a field of five known crop types and could be clearly separated into two clusters – one containing the broad-leaved crops and the other the narrow-leaved crops.

k-means is also used for image and speech coding applications (see below).

Examples of fuzzy c-means applications are:

- Medical diagnosis. Li *et al*. (1993) use a fuzzy c-means algorithm for image segmentation in a study on automatic classification and tissue labelling of two-dimensional magnetic resonance human brain images.
- Acoustic quality control. Meier *et al*. (1994) describe the application of fuzzy c-means to cluster six-dimensional feature vectors as part of a quality control system of ceramic tiles. The signals are derived by hitting the tiles and digitising and filtering the recorded signal. The resulting classes are interpreted as good or bad tiles.
- Water quality. Mukherjee *et al*. (1995) compared fuzzy c-means with two alternative approaches to image segmentation in a study to identify and count bacterial colonies from images.

See also the survey on fuzzy clustering by Yang (1993) for further references to applications of fuzzy c-means.

Bayesian approaches to mixture modelling:

- Some applications of mixture models are given in Chapter 2. Dellaportas (1998) considers the application of mixture modelling to the classification of neolithic ground stone tools. A Bayesian methodology is adopted and developed in three main ways to apply to data (147 measurements on four variables) consisting of variables of mixed type – continuous and categorical; to handle missing values and measurement errors (errors in variables) in the continuous variables. Gibbs sampling is used to generate samples from the posterior densities of interest and classification to one of two classes is based on the mean of the indicator variable, z_i. After a 'burn-in' of 4000 iterations, 4000 further samples were used as the basis for posterior inference.

Examples of self-organising feature map applications include:

- Engineering applications. Kohonen *et al*. (1996) review the self-organising map algorithm and describe several engineering applications including fault detection, process analysis and monitoring, computer vision, speech recognition, robotic control and in the area of telecommunications.

- Human protein analysis. Ferrán *et al*. (1994) use a self-organising map to cluster protein sequences into families. Using 1758 human protein sequences, they cluster using two-dimensional maps of various sizes and label the nodes in the grid using proteins belonging to known sequences.

- Radar target classification. Stewart *et al*. (1994) develop a self-organising map and a learning vector quantisation approach to radar target classification using turntable data of four target types. The data consist of 33-dimensional feature vectors (33 range gates) and 36 000 patterns per target were used. Performance as a function of the number of cluster centres is reported with the learning vector quantisation performance better than a simplistic nearest-neighbour algorithm.

- Fingerprint classification. Halici and Ongun (1996), in a study on automatic fingerprint classification, use a self-organising map, and one modified by preprocessing the feature vectors by combining them with 'certainty' vectors that encode uncertainties in the fingerprint images. Results show an improvement on previous studies using a multilayer perceptron on a database of size 2000.

Sum-of-squares method application:

- Language disorder. Powell *et al*. (1979) use a normal mixture approach and a sum-of-squares method in a study of 86 aphasic cases referred to a speech therapy unit. Four groups are found, which are labelled as severe, high–moderate, low–moderate and mild aphasia.

Vector quantisation has been widely applied as a preprocessor in many studies:

- Speech recognition. Zhang *et al*. (1994) assess three different vector quantisers (including the LBG algorithm and an algorithm based on normal mixture modelling) as preprocessors for a hidden Markov model based recogniser in a small speech recognition problem. They found that the normal mixture model gave the best performance of the subsequent classifier. See also Bergh *et al*. (1985).

- Medical diagnosis. Cosman *et al*. (1993) assess the quality of tree-structured vector quantisation images by the diagnostic performance of radiologists in a study on lung tumour and lymphadenopathy identification. Initial results suggest that a 12 bits per pixel (bpp) computerised tomography chest scan image can be compressed to between 1 bpp and 2 bpp with no significant change in diagnostic accuracy: subjective quality seems to degrade sooner than diagnostic accuracy falls off.

- Speaker recognition. Speaker recognition approaches are reviewed by Furui (1997). Vector quantisation methods are used to compress the training data and produce codebooks of representative feature vectors characterising speaker-specific features. A codebook is generated for each speaker by clustering training feature vectors. At the speaker recognition stage, an input utterance is quantised using the codebook of each speaker and recognition performed by assigning the utterance to the speaker whose codebook produces minimum distortion. A tutorial on vector quantisation for speech coding is given by Makhoul *et al*. (1985).

Spectral clustering applications include the following:

- Analysis of flow cytometry (measurements of microscopic particles in a fluid) data. Zare *et al*. (2010) develop a modification of the spectral clustering approach to large datasets and apply it to flow cytometry data as an example of data containing potentially hundreds of thousands of data points.
- Monitoring blog communities. Ning *et al*. (2007) develop an incremental algorithm that extends the existing offline spectral clustering algorithms to evolving data such as that occurring in the real-time monitoring of the evolving communities of blogs and their links.
- Tissue classification (Crum, 2009).
- Language analysis. Brew and Schulte im Walde (2002) apply spectral methods to the clustering of German verbs. The distance between two ‘verb frames’ is calculated using different measures and transformed to a similarity matrix using a Gaussian similarity function [Equation (11.11)]. The clustering solution gave as good alignment with the distance measure as a ‘gold standard’ human solution.

11.9 Summary and discussion

In this chapter we have covered a wide range of techniques for partitioning a dataset. This has included approaches based on cluster analysis methods and vector quantisation methods. Although both approaches have much in common – they both produce a dissection of a given dataset – there are differences. In cluster analysis, we tend to look for ‘natural’ groupings in the data that may be labelled in terms of the subject matter of the data. In contrast, the vector quantisation methods are developed to optimise some appropriate criterion from communication theory. One area of common ground we have discussed in this chapter is that of optimisation methods with specific implementations in terms of the k-means algorithm in cluster analysis and the LBG algorithm in vector quantisation.

As far as cluster analysis or classification is concerned, there is no single best technique. Different clustering methods can yield different results and some methods will fail to detect obvious clusters. The reason for this is that each method implicitly forces a structure on the given data. For example, the sum-of-squares methods will tend to produce hyperspherical clusters. Also, the fact that there is a wide range of available methods partly stems from the lack of a single definition of the word cluster. There is no universal agreement as to what constitutes a cluster and so a single definition is insufficient.

A further difficulty with cluster analysis is in deciding the number of clusters present. This is a tradeoff between parsimony and some measure of increase in within-cluster homogeneity. This problem is partly due to the difficulty in deciding what a cluster actually is and partly because clustering algorithms tend to produce clusters even when applied to random data.

Both of the above difficulties may be overcome to some degree by considering several possible classifications or comparing classifications on each half of a dataset (McIntyre and Blashfield, 1980; Breckenridge, 1989). The interpretation of these is more important than a rigid inference of the number of groups. But which methods should we employ? There are advantages and disadvantages of all the approaches we have described. The optimisation methods tend to require a large amount of computer time (and consequently may be infeasible for large datasets, though this is becoming less critical these days). Of the hierarchical methods, the single link is preferred by many users. It is the only one to satisfy the Jardine–Sibson conditions, yet with noisy data it can join separate clusters (chaining effect). It is also invariant under monotone transformations of the dissimilarity measure. Ward's method is also popular. The centroid and median methods should be avoided since inversions may make the resulting classification difficult to interpret.

There are several aspects of cluster analysis that we have mentioned only briefly in this chapter and we must refer the reader to the literature on cluster analysis for further details. An important problem is the choice of technique for mixed mode data (i.e. containing both numerical and categorical measurements). Everitt and Merette (1990) (see also Everitt, 1988) propose a finite mixture model approach for clustering mixed mode data, but computational considerations may mean that it is not practically viable when the datasets contain a large number of categorical variables.

The techniques described in this chapter all apply to the clustering of objects. However, there may be some situations where clustering of variables, or simultaneous clustering of objects and variables, is required. In clustering of variables, we seek subsets of variables that are so highly correlated that each can be replaced by any one of the subset, or perhaps a (linear or nonlinear) combination of the members. Many of the techniques described in this chapter can be applied to the clustering of variables and therefore we require a measure of similarity or dissimilarity between variables. Of course, techniques for feature extraction (for example, principal components analysis) perform this process.

Another point to reiterate about cluster analysis is that it is essentially an exploratory method of multivariate data analysis providing a description of the measurements. Once a solution or interpretation has been obtained then the investigator must re-examine and assess the dataset. This may allow further hypotheses (perhaps concerning the variables used in the study, the measures of dissimilarity and the choice of technique) to be generated. These may be tested on a new sample of individuals.

Both cluster analysis and vector quantisation are means of reducing a large amount of data to a form in which it is either easier to describe or represent in a machine. Clustering of data may be performed prior to supervised classification. For example, the number of stored prototypes in a k-nearest-neighbour classifier may be reduced by a clustering procedure. The new prototypes are the cluster means, and the class of the new prototype is decided on a majority basis of the members of the cluster. A development of this approach that adjusts the decision surface by modifying the prototypes is *learning vector quantisation* (Kohonen, 1989).

Self-organising maps may be viewed as a form of *constrained classification*: clustering in which there is some form of constraint on the solution. In this particular case, the constraint is an ordering on the cluster centres. Other forms of constraint may be that objects within a cluster are required to comprise a spatially contiguous set of objects (for example, in some texture segmentation applications). This is an example of *contiguity constrained clustering* [Murtagh, 1992; Gordon, 1999; see also Murtagh (1995), for an application to the outputs of the self-organising map]. Other forms of constraint may be on the topology of the dendrogram or the size or composition of the classes. We refer to Gordon (1996b) for a review.

Spectral clustering has become a popular method in recent years. It is simple to implement and methods to overcome its computational burden when applied to large datasets have been developed (Ning *et al*., 2007; Yan *et al*., 2009).

Gordon (1994, 1996a) provides reviews of approaches to cluster validation; see also Bock (1989), Jain and Dubes (1988), Theodoridis and Koutroumbas (2009) and the papers by Kovács *et al*. (2006) and Halkidi *et al*. (2001).

11.10 Recommendations

There is a large number of techniques to choose from and the availability of computer packages means that analyses can be readily performed. Nevertheless, there are some general guidelines to follow when carrying out a classification.

1. Detect and remove outliers. Many clustering techniques are sensitive to the presence of outliers. Therefore, some of the techniques discussed in Chapter 13 should be used to detect and possibly remove these outliers.

2. Plot the data in two dimensions if possible in order to understand structure in the data. It might be helpful to use the first two principal components (see Chapter 10).

3. Carry out any preprocessing of the data. This may include a reduction in the number of variables or standardisation of the variables to zero mean and unit variance.

4. If the data are not in the form of a dissimilarity matrix then a dissimilarity measure must be chosen (for some techniques) and a dissimilarity matrix formed.

5. Choose an appropriate technique. Of the hierarchical methods, some studies favour the use of the group average link method, but the single-link method gives solutions that are invariant to a monotone transformation measure. The single-link method is the only one to satisfy all the conditions laid down by Jardine and Sibson (1971) and is their preferred method. We recommend against the use of the centroid and median methods since inversions are likely to arise.

6. Evaluate the method. Assess the results of the clustering method you have employed. How do the clusters differ? We recommend that you split the dataset into two parts and compare the results of clustering on each subset. Similar results would suggest that useful structure has been found. Also, use several methods and compare results. With many of the methods some parameters must be specified and it is worthwhile carrying out clustering over a range of parameter values to assess stability.

7. In using vector quantisation as a preprocessor for a supervised classification problem, model each class separately rather than model the whole dataset and label the resulting codewords.

8. Spectral clustering should be considered when clusters may not be compact/concave.

9. If you require some representative prototypes, we recommend using the *k*-means algorithm.

Finally, we reiterate that cluster analysis is usually the first stage in an analysis and unquestioning acceptance of the results of clustering is extremely unwise.

11.11 Notes and references

There is a vast literature on cluster analysis. A very good starting point is the book by Everitt *et al*. (2011). Now in its fifth edition, this book is a mainly nonmathematical account of the most common clustering techniques, together with their advantages and disadvantages. The practical advice is supported by several empirical investigations. Another good introduction to methods and assessments of classification is the book by Gordon (1999). McLachlan and Basford (1988) discuss the mixture model approach to clustering in some detail.

Of the review papers, that by Cormack (1971) is worth reading and provides a good summary of the methods and problems of cluster analysis. The article by Diday and Simon (1976) gives a more mathematical treatment of the methods, together with descriptive algorithms for their implementation.

Several books have an orientation towards biological and ecological matters. Jardine and Sibson (1971) give a mathematical treatment. The book by Sneath and Sokal (1973) is a comprehensive account of cluster analysis and the biological problems to which it can be applied. The book by Clifford and Stephenson (1975) is a nonmathematical general introduction to the ideas and principles of numerical classification and data analysis, though it does not cover many of the approaches described in this chapter, concentrating on hierarchical classificatory procedures. The book by Jain and Dubes (1988) has a pattern recognition emphasis. McLachlan (1992b) reviews cluster analysis in medical research.

A more specialist book is that of Zupan (1982). This monograph is concerned with the problem of implementing hierarchical techniques on large datasets.

The literature on fuzzy techniques in cluster analysis is reviewed by Bezdek and Pal (1992). This book contains a collection of some of the important papers on fuzzy models for pattern recognition, including cluster analysis and supervised classifier design, together with fairly extensive bibliographies. A survey of fuzzy clustering and its applications is provided by Yang (1993). An interesting probabilistic perspective of fuzzy methods is provided by Laviolette *et al*. (1995).

Tutorials and surveys of self-organising maps are given by Kohonen (1990, 1997), Kohonen *et al*. (1996) and Ritter *et al*. (1992).

There are various books on techniques for implementing methods, which give algorithms in the form of FORTRAN code, or pseudo-code. The books by Anderberg (1973), Hartigan (1975), Späth (1980) and Jambu and Lebeaux (1983) all provide a description of a clustering algorithm, FORTRAN source code and a supporting mathematical treatment, sometimes with case studies. The book by Murtagh (1985) covers more recent developments and is also concerned with implementation on parallel machines.

An introduction to spectral clustering is provided by von Luxburg (2007), Spielman and Teng (1996) and Chung (1997).

Exercises

Dataset 1: Generate $n = 500$ samples (x_i, y_i), $i = 1, \dots, n$, according to

$$x_i = \frac{i}{n}\pi + n_x$$

$$y_i = \sin\left(\frac{i}{n}\pi\right) + n_y$$

where n_x and n_y are normally distributed with mean 0.0 and variance 0.01.

Dataset 2: Generate n samples from a multivariate normal (p variables) of diagonal covariance matrix $\sigma^2\boldsymbol{I}$, $\sigma^2 = 1$ and zero mean. Take $n = 40$, $p = 2$.

Dataset 3: This consists of data comprising two classes: class ω_1 is distributed as $0.5N((0,0),\boldsymbol{I}) + 0.5N((2,2),\boldsymbol{I})$ and class $\omega_2 \sim N((2,0),\boldsymbol{I})$ (generate 500 samples for training and test sets, $p(\omega_1) = p(\omega_2) = 0.5$).

1. Is the square of the Euclidean distance a metric? Does it matter for any clustering algorithm?

2. Observations on six variables are made for seven groups of canines and given in Table 11.5 (Manly, 1986; Krzanowski and Marriott, 1994). Construct a dissimilarity matrix using Euclidean distance after standardising each variable to unit variance. Carry out a single-link cluster analysis.

3. Compare the single-link method of clustering with k-means, discussing computational requirements, storage, and applicability of the methods.

4. A mixture of two normals divided by a normal density having the same mean and variance as the mixed density is always bimodal. Prove this for the univariate case.

5. Implement a k-means algorithm and test it on two-dimensional normally distributed data (dataset 2 with $n = 500$). Also, use the algorithm within a tree-structured vector quantiser and compare the two methods.

6. Using dataset 1, code the data using the Luttrell self-organising feature map algorithm (Section 11.5.3). Plot the positions of the centres for various numbers of code vectors.

Table 11.5 Data on mean mandible measurements (from Manly, 1986).

Group	x_1	x_2	x_3	x_4	x_5	x_6
Modern Thai dog	9.7	21.0	19.4	7.7	32.0	36.5
Golden jackal	8.1	16.7	18.3	7.0	30.3	32.9
Chinese wolf	13.5	27.3	26.8	10.6	41.9	48.1
Indian wolf	11.5	24.3	24.5	9.3	40.0	44.6
Cuon	10.7	23.5	21.4	8.5	28.8	37.6
Dingo	9.6	22.6	21.1	8.3	34.4	43.1
Prehistoric dog	10.3	22.1	19.1	8.1	32.3	35.0

Compute the average distortion as a function of the number of code vectors. How would you modify the algorithm for data having circular topology?

7. Using dataset 1, construct a tree-structured vector quantiser, partitioning the clusters with the largest sum-square error at each stage. Compute the average distortion.

8. Using dataset 2, cluster the data using Ward's method and Euclidean distance. Using Gordon's approach for identifying genuine clusters (unimodal null model), how many clusters are valid at the 5% level of significance?

9. Implement a learning vector quantisation algorithm on dataset 3. Plot performance as a function of the number of cluster centres. What would be the advantages and disadvantages of using the resulting cluster centres as centres in a radial basis function network?

10. Show that the single-link dendrogram is invariant to a nonlinear monotone transformation of the dissimilarities.

11. For a distance between two clusters A and B of objects given by $d_{AB} = |\boldsymbol{m}_A - \boldsymbol{m}_b|^2$, where $\boldsymbol{m}_A$ is the mean of the objects in cluster A, show that the formula expressing the distance between a cluster k and a cluster formed by joining i and j is

$$d_{i+j,k} = \frac{n_i}{n_i + n_j} d_{ik} + \frac{n_j}{n_i + n_j} d_{jk} - \frac{n_i n_j}{(n_i + n_j)^2} d_{ij}$$

where there are n_i objects in group i. This is the update rule for the centroid method.

12. Show that, if the number of clusters in a clustering is m and the number of groups in a partition is s and $s \neq m$, the Rand and Jaccard coefficients (Section 11.7.7) are less than 1.