Association rule 4: Support 15%, confidence 95.4% and lift 1.50.

$$\left[\begin{array}{rcl} \text{householder status} & = & \textit{rent} \\ \text{type of home} & \neq & \textit{house} \\ \text{number in household} & \leq & 2 \\ \text{occupation} & \notin & \{\textit{homemaker, student, unemployed}\} \\ \text{income} & \in & [\$20{,}000, \$150{,}000] \end{array}\right]$$
$$\Downarrow$$
$$\text{number of children} = 0$$

There are no great surprises among these particular rules. For the most part they verify intuition. In other contexts where there is less prior information available, unexpected results have a greater chance to emerge. These results do illustrate the type of information generalized association rules can provide, and that the supervised learning approach, coupled with a ruled induction method such as CART or PRIM, can uncover item sets exhibiting high associations among their constituents.

How do these generalized association rules compare to those found earlier by the Apriori algorithm? Since the Apriori procedure gives thousands of rules, it is difficult to compare them. However some general points can be made. The Apriori algorithm is exhaustive—it finds *all* rules with support greater than a specified amount. In contrast, PRIM is a greedy algorithm and is not guaranteed to give an "optimal" set of rules. On the other hand, the Apriori algorithm can deal only with dummy variables and hence could not find some of the above rules. For example, since `type of home` is a categorical input, with a dummy variable for each level, Apriori could not find a rule involving the set

$$\text{type of home} \neq \textit{apartment}.$$

To find this set, we would have to code a dummy variable for *apartment* versus the other categories of type of home. It will not generally be feasible to precode all such potentially interesting comparisons.

14.3 Cluster Analysis

Cluster analysis, also called data segmentation, has a variety of goals. All relate to grouping or segmenting a collection of objects into subsets or "clusters," such that those within each cluster are more closely related to one another than objects assigned to different clusters. An object can be described by a set of measurements, or by its relation to other objects. In addition, the goal is sometimes to arrange the clusters into a natural hierarchy. This involves successively grouping the clusters themselves so

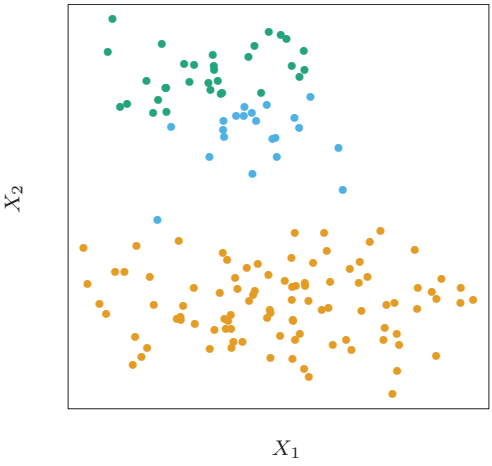

FIGURE 14.4. *Simulated data in the plane, clustered into three classes (represented by orange, blue and green) by the K-means clustering algorithm*

that at each level of the hierarchy, clusters within the same group are more similar to each other than those in different groups.

Cluster analysis is also used to form descriptive statistics to ascertain whether or not the data consists of a set distinct subgroups, each group representing objects with substantially different properties. This latter goal requires an assessment of the degree of difference between the objects assigned to the respective clusters.

Central to all of the goals of cluster analysis is the notion of the degree of similarity (or dissimilarity) between the individual objects being clustered. A clustering method attempts to group the objects based on the definition of similarity supplied to it. This can only come from subject matter considerations. The situation is somewhat similar to the specification of a loss or cost function in prediction problems (supervised learning). There the cost associated with an inaccurate prediction depends on considerations outside the data.

Figure 14.4 shows some simulated data clustered into three groups via the popular K-means algorithm. In this case two of the clusters are not well separated, so that "segmentation" more accurately describes the part of this process than "clustering." K-means clustering starts with guesses for the three cluster centers. Then it alternates the following steps until convergence:

- for each data point, the closest cluster center (in Euclidean distance) is identified;

- each cluster center is replaced by the coordinate-wise average of all data points that are closest to it.

We describe K-means clustering in more detail later, including the problem of how to choose the number of clusters (three in this example). K-means clustering is a *top-down* procedure, while other cluster approaches that we discuss are *bottom-up*. Fundamental to all clustering techniques is the choice of distance or dissimilarity measure between two objects. We first discuss distance measures before describing a variety of algorithms for clustering.

14.3.1 Proximity Matrices

Sometimes the data is represented directly in terms of the proximity (alikeness or affinity) between pairs of objects. These can be either *similarities* or *dissimilarities* (difference or lack of affinity). For example, in social science experiments, participants are asked to judge by how much certain objects differ from one another. Dissimilarities can then be computed by averaging over the collection of such judgments. This type of data can be represented by an $N \times N$ matrix $\mathbf{D}$, where N is the number of objects, and each element $d_{ii'}$ records the proximity between the ith and i'th objects. This matrix is then provided as input to the clustering algorithm.

Most algorithms presume a matrix of dissimilarities with nonnegative entries and zero diagonal elements: $d_{ii} = 0,\ i = 1, 2, \ldots, N$. If the original data were collected as similarities, a suitable monotone-decreasing function can be used to convert them to dissimilarities. Also, most algorithms assume symmetric dissimilarity matrices, so if the original matrix $\mathbf{D}$ is not symmetric it must be replaced by $(\mathbf{D}+\mathbf{D}^T)/2$. Subjectively judged dissimilarities are seldom *distances* in the strict sense, since the triangle inequality $d_{ii'} \le d_{ik}+d_{i'k}$, for all $k \in \{1, \ldots, N\}$ does not hold. Thus, some algorithms that assume distances cannot be used with such data.

14.3.2 Dissimilarities Based on Attributes

Most often we have measurements x_{ij} for $i = 1, 2, \ldots, N$, on variables $j = 1, 2, \ldots, p$ (also called *attributes*). Since most of the popular clustering algorithms take a dissimilarity matrix as their input, we must first construct pairwise dissimilarities between the observations. In the most common case, we define a dissimilarity $d_j(x_{ij}, x_{i'j})$ between values of the jth attribute, and then define

$$D(x_i, x_{i'}) = \sum_{j=1}^{p} d_j(x_{ij}, x_{i'j}) \tag{14.20}$$

as the dissimilarity between objects i and i'. By far the most common choice is squared distance

$$d_j(x_{ij}, x_{i'j}) = (x_{ij} - x_{i'j})^2. \tag{14.21}$$

However, other choices are possible, and can lead to potentially different results. For nonquantitative attributes (e.g., categorical data), squared distance may not be appropriate. In addition, it is sometimes desirable to weigh attributes differently rather than giving them equal weight as in (14.20).

We first discuss alternatives in terms of the attribute type:

Quantitative variables. Measurements of this type of variable or attribute are represented by continuous real-valued numbers. It is natural to define the "error" between them as a monotone-increasing function of their absolute difference

$$d(x_i, x_{i'}) = l(|x_i - x_{i'}|).$$

Besides squared-error loss $(x_i - x_{i'})^2$, a common choice is the identity (absolute error). The former places more emphasis on larger differences than smaller ones. Alternatively, clustering can be based on the correlation

$$\rho(x_i, x_{i'}) = \frac{\sum_j (x_{ij} - \bar{x}_i)(x_{i'j} - \bar{x}_{i'})}{\sqrt{\sum_j (x_{ij} - \bar{x}_i)^2 \sum_j (x_{i'j} - \bar{x}_{i'})^2}}, \tag{14.22}$$

with $\bar{x}_i = \sum_j x_{ij}/p$. Note that this is averaged over *variables*, not observations. If the *observations* are first standardized, then $\sum_j (x_{ij} - x_{i'j})^2 \propto 2(1 - \rho(x_i, x_{i'}))$. Hence clustering based on correlation (similarity) is equivalent to that based on squared distance (dissimilarity).

Ordinal variables. The values of this type of variable are often represented as contiguous integers, and the realizable values are considered to be an ordered set. Examples are academic grades (A, B, C, D, F), degree of preference (can't stand, dislike, OK, like, terrific). Rank data are a special kind of ordinal data. Error measures for ordinal variables are generally defined by replacing their M original values with

$$\frac{i - 1/2}{M}, \; i = 1, \ldots, M \tag{14.23}$$

in the prescribed order of their original values. They are then treated as quantitative variables on this scale.

Categorical variables. With unordered categorical (also called nominal) variables, the degree-of-difference between pairs of values must be delineated explicitly. If the variable assumes M distinct values, these can be arranged in a symmetric $M \times M$ matrix with elements $L_{rr'} = L_{r'r}, L_{rr} = 0, L_{rr'} \geq 0$. The most common choice is $L_{rr'} = 1$ for all $r \neq r'$, while unequal losses can be used to emphasize some errors more than others.

14.3.3 Object Dissimilarity

Next we define a procedure for combining the p-individual attribute dissimilarities $d_j(x_{ij}, x_{i'j})$, $j = 1, 2, \ldots, p$ into a single overall measure of dissimilarity $D(x_i, x_{i'})$ between two objects or observations $(x_i, x_{i'})$ possessing the respective attribute values. This is nearly always done by means of a weighted average (convex combination)

$$D(x_i, x_{i'}) = \sum_{j=1}^{p} w_j \cdot d_j(x_{ij}, x_{i'j}); \quad \sum_{j=1}^{p} w_j = 1. \tag{14.24}$$

Here w_j is a weight assigned to the jth attribute regulating the relative influence of that variable in determining the overall dissimilarity between objects. This choice should be based on subject matter considerations.

It is important to realize that setting the weight w_j to the same value for each variable (say, $w_j = 1 \; \forall \; j$) does *not* necessarily give all attributes equal influence. The influence of the jth attribute X_j on object dissimilarity $D(x_i, x_{i'})$ (14.24) depends upon its relative contribution to the average object dissimilarity measure over all pairs of observations in the data set

$$\bar{D} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} D(x_i, x_{i'}) = \sum_{j=1}^{p} w_j \cdot \bar{d}_j,$$

with

$$\bar{d}_j = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} d_j(x_{ij}, x_{i'j}) \tag{14.25}$$

being the average dissimilarity on the jth attribute. Thus, the relative influence of the jth variable is $w_j \cdot \bar{d}_j$, and setting $w_j \sim 1/\bar{d}_j$ would give all attributes equal influence in characterizing overall dissimilarity between objects. For example, with p quantitative variables and squared-error distance used for each coordinate, then (14.24) becomes the (weighted) squared Euclidean distance

$$D_I(x_i, x_{i'}) = \sum_{j=1}^{p} w_j \cdot (x_{ij} - x_{i'j})^2 \tag{14.26}$$

between pairs of points in an $\mathbb{R}^p$, with the quantitative variables as axes. In this case (14.25) becomes

$$\bar{d}_j = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} (x_{ij} - x_{i'j})^2 = 2 \cdot \mathrm{var}_j, \tag{14.27}$$

where var_j is the sample estimate of $\mathrm{Var}(X_j)$. Thus, the relative importance of each such variable is proportional to its variance over the data

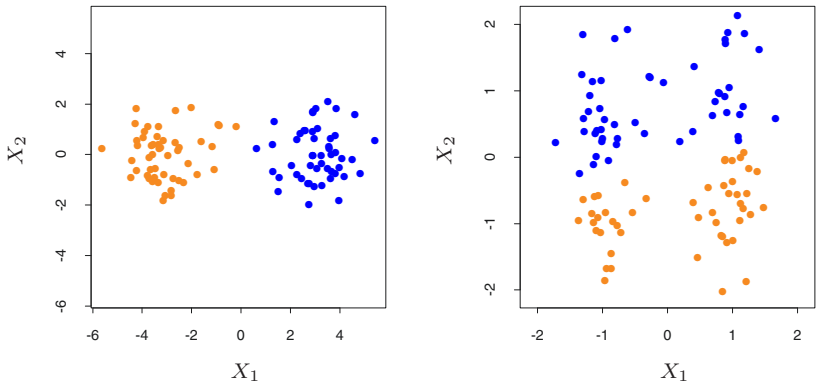

FIGURE 14.5. *Simulated data: on the left, K-means clustering (with K=2) has been applied to the raw data. The two colors indicate the cluster memberships. On the right, the features were first standardized before clustering. This is equivalent to using feature weights* $1/[2 \cdot \text{var}(X_j)]$. *The standardization has obscured the two well-separated groups. Note that each plot uses the same units in the horizontal and vertical axes.*

set. In general, setting $w_j = 1/\bar{d}_j$ for all attributes, irrespective of type, will cause each one of them to equally influence the overall dissimilarity between pairs of objects $(x_i, x_{i'})$. Although this may seem reasonable, and is often recommended, it can be highly counterproductive. If the goal is to segment the data into groups of similar objects, all attributes may not contribute equally to the (problem-dependent) notion of dissimilarity between objects. Some attribute value differences may reflect greater actual object dissimilarity in the context of the problem domain.

If the goal is to discover natural groupings in the data, some attributes may exhibit more of a grouping tendency than others. Variables that are more relevant in separating the groups should be assigned a higher influence in defining object dissimilarity. Giving all attributes equal influence in this case will tend to obscure the groups to the point where a clustering algorithm cannot uncover them. Figure 14.5 shows an example.

Although simple generic prescriptions for choosing the individual attribute dissimilarities $d_j(x_{ij}, x_{i'j})$ and their weights w_j can be comforting, there is no substitute for careful thought in the context of each individual problem. Specifying an appropriate dissimilarity measure is far more important in obtaining success with clustering than choice of clustering algorithm. This aspect of the problem is emphasized less in the clustering literature than the algorithms themselves, since it depends on domain knowledge specifics and is less amenable to general research.

Finally, often observations have *missing values* in one or more of the attributes. The most common method of incorporating missing values in dissimilarity calculations (14.24) is to omit each observation pair $x_{ij}, x_{i'j}$ having at least one value missing, when computing the dissimilarity between observations x_i and $x_{i'}$. This method can fail in the circumstance when both observations have no measured values in common. In this case both observations could be deleted from the analysis. Alternatively, the missing values could be imputed using the mean or median of each attribute over the nonmissing data. For categorical variables, one could consider the value "missing" as just another categorical value, if it were reasonable to consider two objects as being similar if they both have missing values on the same variables.

14.3.4 Clustering Algorithms

The goal of cluster analysis is to partition the observations into groups ("clusters") so that the pairwise dissimilarities between those assigned to the same cluster tend to be smaller than those in different clusters. Clustering algorithms fall into three distinct types: combinatorial algorithms, mixture modeling, and mode seeking.

Combinatorial algorithms work directly on the observed data with no direct reference to an underlying probability model. *Mixture modeling* supposes that the data is an *i.i.d* sample from some population described by a probability density function. This density function is characterized by a parameterized model taken to be a mixture of component density functions; each component density describes one of the clusters. This model is then fit to the data by maximum likelihood or corresponding Bayesian approaches. *Mode seekers* ("bump hunters") take a nonparametric perspective, attempting to directly estimate distinct modes of the probability density function. Observations "closest" to each respective mode then define the individual clusters.

Mixture modeling is described in Section 6.8. The PRIM algorithm, discussed in Sections 9.3 and 14.2.5, is an example of mode seeking or "bump hunting." We discuss combinatorial algorithms next.

14.3.5 Combinatorial Algorithms

The most popular clustering algorithms directly assign each observation to a group or cluster without regard to a probability model describing the data. Each observation is uniquely labeled by an integer $i \in \{1, \cdots, N\}$. A prespecified number of clusters $K < N$ is postulated, and each one is labeled by an integer $k \in \{1, \ldots, K\}$. Each observation is assigned to one and only one cluster. These assignments can be characterized by a many-to-one mapping, or *encoder* $k = C(i)$, that assigns the ith observation to the kth cluster. One seeks the particular encoder $C^*(i)$ that achieves the

required goal (details below), based on the dissimilarities $d(x_i, x_{i'})$ between every pair of observations. These are specified by the user as described above. Generally, the encoder $C(i)$ is explicitly delineated by giving its value (cluster assignment) for each observation i. Thus, the "parameters" of the procedure are the individual cluster assignments for each of the N observations. These are adjusted so as to *minimize* a "loss" function that characterizes the degree to which the clustering goal is *not* met.

One approach is to directly specify a mathematical loss function and attempt to minimize it through some combinatorial optimization algorithm. Since the goal is to assign close points to the same cluster, a natural loss (or "energy") function would be

$$W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')=k} d(x_i, x_{i'}). \tag{14.28}$$

This criterion characterizes the extent to which observations assigned to the same cluster tend to be close to one another. It is sometimes referred to as the "within cluster" point scatter since

$$T = \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N} d_{ii'} = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\left(\sum_{C(i')=k} d_{ii'} + \sum_{C(i')\neq k} d_{ii'}\right),$$

or

$$T = W(C) + B(C),$$

where $d_{ii'} = d(x_i, x_{i'})$. Here T is the *total* point scatter, which is a constant given the data, independent of cluster assignment. The quantity

$$B(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')\neq k} d_{ii'} \tag{14.29}$$

is the *between-cluster* point scatter. This will tend to be large when observations assigned to different clusters are far apart. Thus one has

$$W(C) = T - B(C)$$

and minimizing $W(C)$ is equivalent to *maximizing* $B(C)$.

Cluster analysis by combinatorial optimization is straightforward in principle. One simply minimizes W or equivalently maximizes B over all possible assignments of the N data points to K clusters. Unfortunately, such optimization by complete enumeration is feasible only for very small data sets. The number of distinct assignments is (Jain and Dubes, 1988)

$$S(N, K) = \frac{1}{K!}\sum_{k=1}^{K}(-1)^{K-k}\binom{K}{k}k^N. \tag{14.30}$$

For example, $S(10, 4) = 34,105$ which is quite feasible. But, $S(N, K)$ grows very rapidly with increasing values of its arguments. Already $S(19, 4) \simeq$

10^{10}, and most clustering problems involve much larger data sets than $N = 19$. For this reason, practical clustering algorithms are able to examine only a very small fraction of all possible encoders $k = C(i)$. The goal is to identify a small subset that is likely to contain the optimal one, or at least a good suboptimal partition.

Such feasible strategies are based on iterative greedy descent. An initial partition is specified. At each iterative step, the cluster assignments are changed in such a way that the value of the criterion is improved from its previous value. Clustering algorithms of this type differ in their prescriptions for modifying the cluster assignments at each iteration. When the prescription is unable to provide an improvement, the algorithm terminates with the current assignments as its solution. Since the assignment of observations to clusters at any iteration is a perturbation of that for the previous iteration, only a very small fraction of all possible assignments (14.30) are examined. However, these algorithms converge to *local* optima which may be highly suboptimal when compared to the global optimum.

14.3.6 K-means

The K-means algorithm is one of the most popular iterative descent clustering methods. It is intended for situations in which all variables are of the quantitative type, and squared Euclidean distance

$$d(x_i, x_{i'}) = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = ||x_i - x_{i'}||^2$$

is chosen as the dissimilarity measure. Note that weighted Euclidean distance can be used by redefining the x_{ij} values (Exercise 14.1).

The within-point scatter (14.28) can be written as

$$\begin{aligned} W(C) &= \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} ||x_i - x_{i'}||^2 \\ &= \sum_{k=1}^{K} N_k \sum_{C(i)=k} ||x_i - \bar{x}_k||^2, \end{aligned} \tag{14.31}$$

where $\bar{x}_k = (\bar{x}_{1k}, \ldots, \bar{x}_{pk})$ is the mean vector associated with the kth cluster, and $N_k = \sum_{i=1}^{N} I(C(i) = k)$. Thus, the criterion is minimized by assigning the N observations to the K clusters in such a way that within each cluster the average dissimilarity of the observations from the cluster mean, as defined by the points in that cluster, is minimized.

An iterative descent algorithm for solving

Algorithm 14.1 *K-means Clustering.*

1. For a given cluster assignment C, the total cluster variance (14.33) is minimized with respect to $\{m_1, \ldots, m_K\}$ yielding the means of the currently assigned clusters (14.32).

2. Given a current set of means $\{m_1, \ldots, m_K\}$, (14.33) is minimized by assigning each observation to the closest (current) cluster mean. That is,
$$C(i) = \underset{1 \le k \le K}{\operatorname{argmin}} ||x_i - m_k||^2. \tag{14.34}$$

3. Steps 1 and 2 are iterated until the assignments do not change.

$$C^* = \min_C \sum_{k=1}^{K} N_k \sum_{C(i)=k} ||x_i - \bar{x}_k||^2$$

can be obtained by noting that for any set of observations S

$$\bar{x}_S = \underset{m}{\operatorname{argmin}} \sum_{i \in S} ||x_i - m||^2. \tag{14.32}$$

Hence we can obtain C^* by solving the enlarged optimization problem

$$\min_{C, \{m_k\}_1^K} \sum_{k=1}^{K} N_k \sum_{C(i)=k} ||x_i - m_k||^2. \tag{14.33}$$

This can be minimized by an alternating optimization procedure given in Algorithm 14.1.

Each of steps 1 and 2 reduces the value of the criterion (14.33), so that convergence is assured. However, the result may represent a suboptimal local minimum. The algorithm of Hartigan and Wong (1979) goes further, and ensures that there is no single switch of an observation from one group to another group that will decrease the objective. In addition, one should start the algorithm with many different random choices for the starting means, and choose the solution having smallest value of the objective function.

Figure 14.6 shows some of the K-means iterations for the simulated data of Figure 14.4. The centroids are depicted by "O"s. The straight lines show the partitioning of points, each sector being the set of points closest to each centroid. This partitioning is called the *Voronoi tessellation*. After 20 iterations the procedure has converged.

14.3.7 Gaussian Mixtures as Soft K-means Clustering

The K-means clustering procedure is closely related to the EM algorithm for estimating a certain Gaussian mixture model. (Sections 6.8 and 8.5.1).

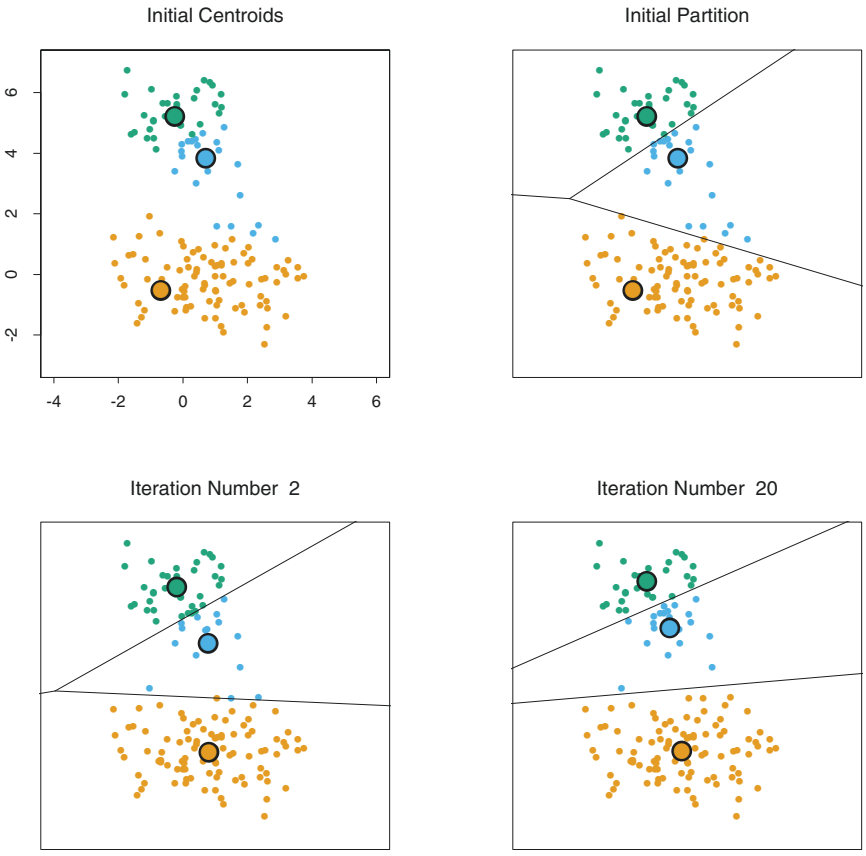

FIGURE 14.6. *Successive iterations of the K-means clustering algorithm for the simulated data of Figure 14.4.*

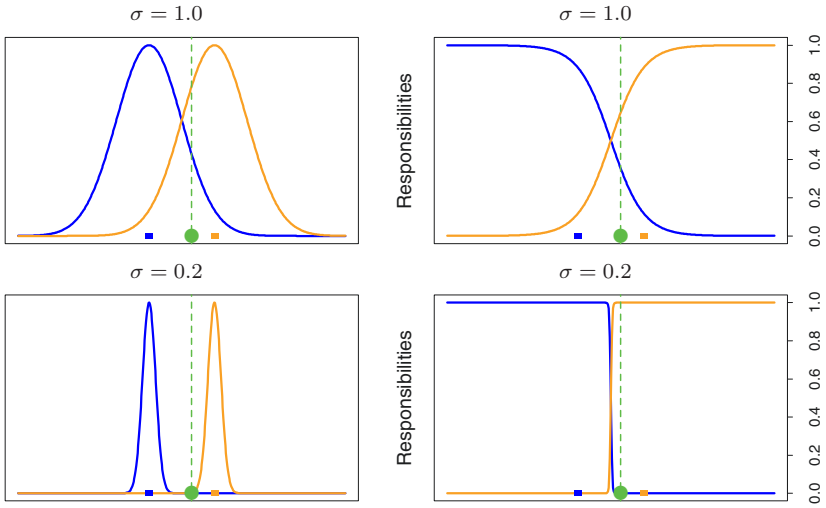

FIGURE 14.7. *(Left panels:) two Gaussian densities $g_0(x)$ and $g_1(x)$ (blue and orange) on the real line, and a single data point (green dot) at $x = 0.5$. The colored squares are plotted at $x = -1.0$ and $x = 1.0$, the means of each density. (Right panels:) the relative densities $g_0(x)/(g_0(x) + g_1(x))$ and $g_1(x)/(g_0(x) + g_1(x))$, called the "responsibilities" of each cluster, for this data point. In the top panels, the Gaussian standard deviation $\sigma = 1.0$; in the bottom panels $\sigma = 0.2$. The EM algorithm uses these responsibilities to make a "soft" assignment of each data point to each of the two clusters. When σ is fairly large, the responsibilities can be near 0.5 (they are 0.36 and 0.64 in the top right panel). As $\sigma \to 0$, the responsibilities $\to 1$, for the cluster center closest to the target point, and 0 for all other clusters. This "hard" assignment is seen in the bottom right panel.*

The E-step of the EM algorithm assigns "responsibilities" for each data point based in its relative density under each mixture component, while the M-step recomputes the component density parameters based on the current responsibilities. Suppose we specify K mixture components, each with a Gaussian density having scalar covariance matrix $\sigma^2\mathbf{I}$. Then the relative density under each mixture component is a monotone function of the Euclidean distance between the data point and the mixture center. Hence in this setup EM is a "soft" version of K-means clustering, making probabilistic (rather than deterministic) assignments of points to cluster centers. As the variance $\sigma^2 \to 0$, these probabilities become 0 and 1, and the two methods coincide. Details are given in Exercise 14.2. Figure 14.7 illustrates this result for two clusters on the real line.

14.3.8 Example: Human Tumor Microarray Data

We apply K-means clustering to the human tumor microarray data described in Chapter 1. This is an example of high-dimensional clustering.

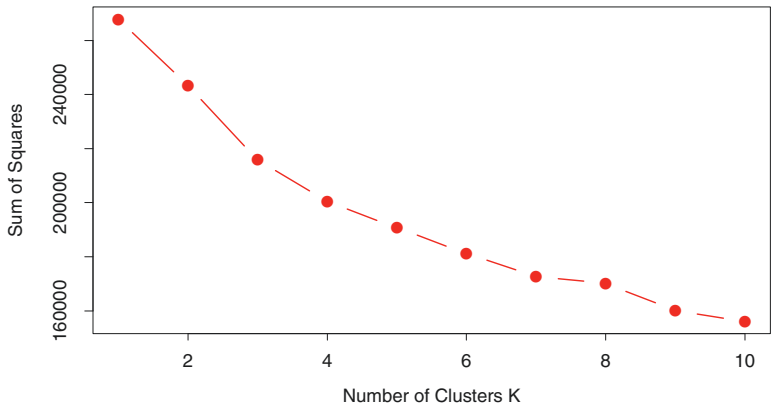

FIGURE 14.8. *Total within-cluster sum of squares for K-means clustering applied to the human tumor microarray data.*

TABLE 14.2. *Human tumor data: number of cancer cases of each type, in each of the three clusters from K-means clustering.*

Cluster	Breast	CNS	Colon	K562	Leukemia	MCF7
1	3	5	0	0	0	0
2	2	0	0	2	6	2
3	2	0	7	0	0	0
Cluster	Melanoma	NSCLC	Ovarian	Prostate	Renal	Unknown
1	1	7	6	2	9	1
2	7	2	0	0	0	0
3	0	0	0	0	0	0

The data are a 6830×64 matrix of real numbers, each representing an expression measurement for a gene (row) and sample (column). Here we cluster the samples, each of which is a vector of length 6830, corresponding to expression values for the 6830 genes. Each sample has a label such as `breast` (for breast cancer), `melanoma`, and so on; we don't use these labels in the clustering, but will examine *posthoc* which labels fall into which clusters.

We applied K-means clustering with K running from 1 to 10, and computed the total within-sum of squares for each clustering, shown in Figure 14.8. Typically one looks for a kink in the sum of squares curve (or its logarithm) to locate the optimal number of clusters (see Section 14.3.11). Here there is no clear indication: for illustration we chose $K = 3$ giving the three clusters shown in Table 14.2.

FIGURE 14.9. *Sir Ronald A. Fisher (*1890 − 1962*) was one of the founders of modern day statistics, to whom we owe maximum-likelihood, sufficiency, and many other fundamental concepts. The image on the left is a* 1024×1024 *grayscale image at* 8 *bits per pixel. The center image is the result of* 2×2 *block VQ, using* 200 *code vectors, with a compression rate of* 1.9 *bits/pixel. The right image uses only four code vectors, with a compression rate of* 0.50 *bits/pixel*

We see that the procedure is successful at grouping together samples of the same cancer. In fact, the two breast cancers in the second cluster were later found to be misdiagnosed and were melanomas that had metastasized. However, K-means clustering has shortcomings in this application. For one, it does not give a linear ordering of objects within a cluster: we have simply listed them in alphabetic order above. Secondly, as the number of clusters K is changed, the cluster memberships can change in arbitrary ways. That is, with say four clusters, the clusters need not be nested within the three clusters above. For these reasons, hierarchical clustering (described later), is probably preferable for this application.

14.3.9 Vector Quantization

The K-means clustering algorithm represents a key tool in the apparently unrelated area of image and signal compression, particularly in *vector quantization* or VQ (Gersho and Gray, 1992). The left image in Figure 14.9[2] is a digitized photograph of a famous statistician, Sir Ronald Fisher. It consists of 1024×1024 pixels, where each pixel is a grayscale value ranging from 0 to 255, and hence requires 8 bits of storage per pixel. The entire image occupies 1 megabyte of storage. The center image is a VQ-compressed version of the left panel, and requires 0.239 of the storage (at some loss in quality). The right image is compressed even more, and requires only 0.0625 of the storage (at a considerable loss in quality).

The version of VQ implemented here first breaks the image into small blocks, in this case 2×2 blocks of pixels. Each of the 512×512 blocks of four

[2]This example was prepared by Maya Gupta.

numbers is regarded as a vector in $\mathbb{R}^4$. A K-means clustering algorithm (also known as Lloyd's algorithm in this context) is run in this space. The center image uses $K = 200$, while the right image $K = 4$. Each of the 512×512 pixel blocks (or points) is approximated by its closest cluster centroid, known as a codeword. The clustering process is called the *encoding* step, and the collection of centroids is called the *codebook*.

To represent the approximated image, we need to supply for each block the identity of the codebook entry that approximates it. This will require $\log_2(K)$ bits per block. We also need to supply the codebook itself, which is $K \times 4$ real numbers (typically negligible). Overall, the storage for the compressed image amounts to $\log_2(K)/(4 \cdot 8)$ of the original (0.239 for $K = 200$, 0.063 for $K = 4$). This is typically expressed as a *rate* in bits per pixel: $\log_2(K)/4$, which are 1.91 and 0.50, respectively. The process of constructing the approximate image from the centroids is called the *decoding* step.

Why do we expect VQ to work at all? The reason is that for typical everyday images like photographs, many of the blocks look the same. In this case there are many almost pure white blocks, and similarly pure gray blocks of various shades. These require only one block each to represent them, and then multiple pointers to that block.

What we have described is known as *lossy* compression, since our images are degraded versions of the original. The degradation or *distortion* is usually measured in terms of mean squared error. In this case $D = 0.89$ for $K = 200$ and $D = 16.95$ for $K = 4$. More generally a rate/distortion curve would be used to assess the tradeoff. One can also perform *lossless* compression using block clustering, and still capitalize on the repeated patterns. If you took the original image and losslessly compressed it, the best you would do is 4.48 bits per pixel.

We claimed above that $\log_2(K)$ bits were needed to identify each of the K codewords in the codebook. This uses a fixed-length code, and is inefficient if some codewords occur many more times than others in the image. Using Shannon coding theory, we know that in general a variable length code will do better, and the rate then becomes $-\sum_{\ell=1}^{K} p_\ell \log_2(p_\ell)/4$. The term in the numerator is the entropy of the distribution p_ℓ of the codewords in the image. Using variable length coding our rates come down to 1.42 and 0.39, respectively. Finally, there are many generalizations of VQ that have been developed: for example, tree-structured VQ finds the centroids with a top-down, 2-means style algorithm, as alluded to in Section 14.3.12. This allows successive refinement of the compression. Further details may be found in Gersho and Gray (1992).

14.3.10 K-medoids

As discussed above, the K-means algorithm is appropriate when the dissimilarity measure is taken to be squared Euclidean distance $D(x_i, x_{i'})$

Algorithm 14.2 *K-medoids Clustering.*

1. For a given cluster assignment C find the observation in the cluster minimizing total distance to other points in that cluster:

$$i_k^* = \underset{\{i:C(i)=k\}}{\operatorname{argmin}} \sum_{C(i')=k} D(x_i, x_{i'}). \tag{14.35}$$

   Then $m_k = x_{i_k^*}$, $k = 1, 2, \ldots, K$ are the current estimates of the cluster centers.

2. Given a current set of cluster centers $\{m_1, \ldots, m_K\}$, minimize the total error by assigning each observation to the closest (current) cluster center:

$$C(i) = \underset{1 \le k \le K}{\operatorname{argmin}} D(x_i, m_k). \tag{14.36}$$

3. Iterate steps 1 and 2 until the assignments do not change.

(14.112). This requires all of the variables to be of the quantitative type. In addition, using *squared* Euclidean distance places the highest influence on the largest distances. This causes the procedure to lack robustness against outliers that produce very large distances. These restrictions can be removed at the expense of computation.

The only part of the K-means algorithm that assumes squared Euclidean distance is the minimization step (14.32); the cluster representatives $\{m_1, \ldots, m_K\}$ in (14.33) are taken to be the means of the currently assigned clusters. The algorithm can be generalized for use with arbitrarily defined dissimilarities $D(x_i, x_{i'})$ by replacing this step by an explicit optimization with respect to $\{m_1, \ldots, m_K\}$ in (14.33). In the most common form, centers for each cluster are restricted to be one of the observations assigned to the cluster, as summarized in Algorithm 14.2. This algorithm assumes attribute data, but the approach can also be applied to data described *only* by proximity matrices (Section 14.3.1). There is no need to explicitly compute cluster centers; rather we just keep track of the indices i_k^*.

Solving (14.32) for each provisional cluster k requires an amount of computation proportional to the number of observations assigned to it, whereas for solving (14.35) the computation increases to $O(N_k^2)$. Given a set of cluster "centers," $\{i_1, \ldots, i_K\}$, obtaining the new assignments

$$C(i) = \underset{1 \le k \le K}{\operatorname{argmin}} d_{ii_k^*} \tag{14.37}$$

requires computation proportional to $K \cdot N$ as before. Thus, K-medoids is far more computationally intensive than K-means.

Alternating between (14.35) and (14.37) represents a particular heuristic search strategy for trying to solve

TABLE 14.3. *Data from a political science survey: values are average pairwise dissimilarities of countries from a questionnaire given to political science students.*

	BEL	BRA	CHI	CUB	EGY	FRA	IND	ISR	USA	USS	YUG
BRA	5.58										
CHI	7.00	6.50									
CUB	7.08	7.00	3.83								
EGY	4.83	5.08	8.17	5.83							
FRA	2.17	5.75	6.67	6.92	4.92						
IND	6.42	5.00	5.58	6.00	4.67	6.42					
ISR	3.42	5.50	6.42	6.42	5.00	3.92	6.17				
USA	2.50	4.92	6.25	7.33	4.50	2.25	6.33	2.75			
USS	6.08	6.67	4.25	2.67	6.00	6.17	6.17	6.92	6.17		
YUG	5.25	6.83	4.50	3.75	5.75	5.42	6.08	5.83	6.67	3.67	
ZAI	4.75	3.00	6.08	6.67	5.00	5.58	4.83	6.17	5.67	6.50	6.92

$$\min_{C,\,\{i_k\}_1^K} \sum_{k=1}^{K} \sum_{C(i)=k} d_{ii_k}. \tag{14.38}$$

Kaufman and Rousseeuw (1990) propose an alternative strategy for directly solving (14.38) that provisionally exchanges each center i_k with an observation that is not currently a center, selecting the exchange that produces the greatest reduction in the value of the criterion (14.38). This is repeated until no advantageous exchanges can be found. Massart et al. (1983) derive a branch-and-bound combinatorial method that finds the global minimum of (14.38) that is practical only for very small data sets.

Example: Country Dissimilarities

This example, taken from Kaufman and Rousseeuw (1990), comes from a study in which political science students were asked to provide pairwise dissimilarity measures for 12 countries: Belgium, Brazil, Chile, Cuba, Egypt, France, India, Israel, United States, Union of Soviet Socialist Republics, Yugoslavia and Zaire. The average dissimilarity scores are given in Table 14.3. We applied 3-medoid clustering to these dissimilarities. Note that K-means clustering could not be applied because we have only distances rather than raw observations. The left panel of Figure 14.10 shows the dissimilarities reordered and blocked according to the 3-medoid clustering. The right panel is a two-dimensional multidimensional scaling plot, with the 3-medoid clusters assignments indicated by colors (multidimensional scaling is discussed in Section 14.8.) Both plots show three well-separated clusters, but the MDS display indicates that "Egypt" falls about halfway between two clusters.

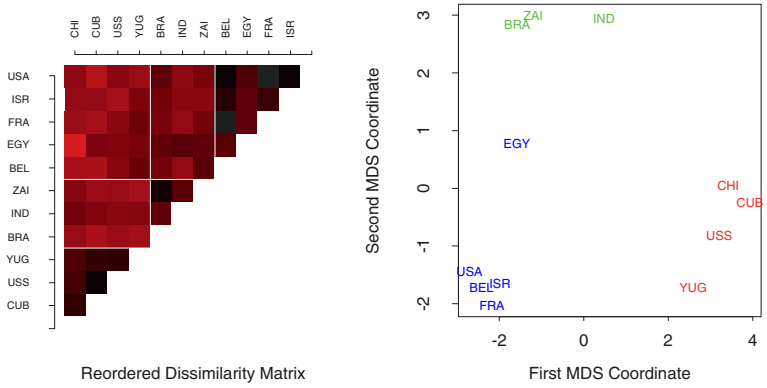

FIGURE 14.10. *Survey of country dissimilarities. (Left panel:) dissimilarities reordered and blocked according to 3-medoid clustering. Heat map is coded from most similar (dark red) to least similar (bright red). (Right panel:) two-dimensional multidimensional scaling plot, with 3-medoid clusters indicated by different colors.*

14.3.11 Practical Issues

In order to apply K-means or K-medoids one must select the number of clusters K^* and an initialization. The latter can be defined by specifying an initial set of centers $\{m_1, \ldots, m_K\}$ or $\{i_1, \ldots, i_K\}$ or an initial encoder $C(i)$. Usually specifying the centers is more convenient. Suggestions range from simple random selection to a deliberate strategy based on forward stepwise assignment. At each step a new center i_k is chosen to minimize the criterion (14.33) or (14.38), given the centers $i_1, \ldots, i_{k-1}$ chosen at the previous steps. This continues for K steps, thereby producing K initial centers with which to begin the optimization algorithm.

A choice for the number of clusters K depends on the goal. For data segmentation K is usually defined as part of the problem. For example, a company may employ K sales people, and the goal is to partition a customer database into K segments, one for each sales person, such that the customers assigned to each one are as similar as possible. Often, however, cluster analysis is used to provide a descriptive statistic for ascertaining the extent to which the observations comprising the data base fall into natural distinct groupings. Here the number of such groups K^* is unknown and one requires that it, as well as the groupings themselves, be estimated from the data.

Data-based methods for estimating K^* typically examine the within-cluster dissimilarity W_K as a function of the number of clusters K. Separate solutions are obtained for $K \in \{1, 2, \ldots, K_{\max}\}$. The corresponding values

$\{W_1, W_2, \ldots, W_{K_{\max}}\}$ generally decrease with increasing K. This will be the case even when the criterion is evaluated on an independent test set, since a large number of cluster centers will tend to fill the feature space densely and thus will be close to all data points. Thus cross-validation techniques, so useful for model selection in supervised learning, cannot be utilized in this context.

The intuition underlying the approach is that if there are actually K^* distinct groupings of the observations (as defined by the dissimilarity measure), then for $K < K^*$ the clusters returned by the algorithm will each contain a subset of the true underlying groups. That is, the solution will not assign observations in the same naturally occurring group to different estimated clusters. To the extent that this is the case, the solution criterion value will tend to decrease substantially with each successive increase in the number of specified clusters, $W_{K+1} \ll W_K$, as the natural groups are successively assigned to separate clusters. For $K > K^*$, one of the estimated clusters must partition at least one of the natural groups into two subgroups. This will tend to provide a smaller decrease in the criterion as K is further increased. Splitting a natural group, within which the observations are all quite close to each other, reduces the criterion less than partitioning the union of two well-separated groups into their proper constituents.

To the extent this scenario is realized, there will be a sharp decrease in successive differences in criterion value, $W_K - W_{K+1}$, at $K = K^*$. That is, $\{W_K - W_{K+1} \,|\, K < K^*\} \gg \{W_K - W_{K+1} \,|\, K \geq K^*\}$. An estimate $\hat{K}^*$ for K^* is then obtained by identifying a "kink" in the plot of W_K as a function of K. As with other aspects of clustering procedures, this approach is somewhat heuristic.

The recently proposed *Gap statistic* (Tibshirani et al., 2001b) compares the curve $\log W_K$ to the curve obtained from data uniformly distributed over a rectangle containing the data. It estimates the optimal number of clusters to be the place where the gap between the two curves is largest. Essentially this is an automatic way of locating the aforementioned "kink." It also works reasonably well when the data fall into a single cluster, and in that case will tend to estimate the optimal number of clusters to be one. This is the scenario where most other competing methods fail.

Figure 14.11 shows the result of the Gap statistic applied to simulated data of Figure 14.4. The left panel shows $\log W_K$ for $K = 1, 2, \ldots, 8$ clusters (green curve) and the expected value of $\log W_K$ over 20 simulations from uniform data (blue curve). The right panel shows the gap curve, which is the expected curve minus the observed curve. Shown also are error bars of halfwidth $s'_K = s_K\sqrt{1 + 1/20}$, where s_K is the standard deviation of $\log W_K$ over the 20 simulations. The Gap curve is maximized at $K = 2$ clusters. If $G(K)$ is the Gap curve at K clusters, the formal rule for estimating K^* is

$$K^* = \underset{K}{\operatorname{argmin}}\{K | G(K) \geq G(K+1) - s'_{K+1}\}. \tag{14.39}$$

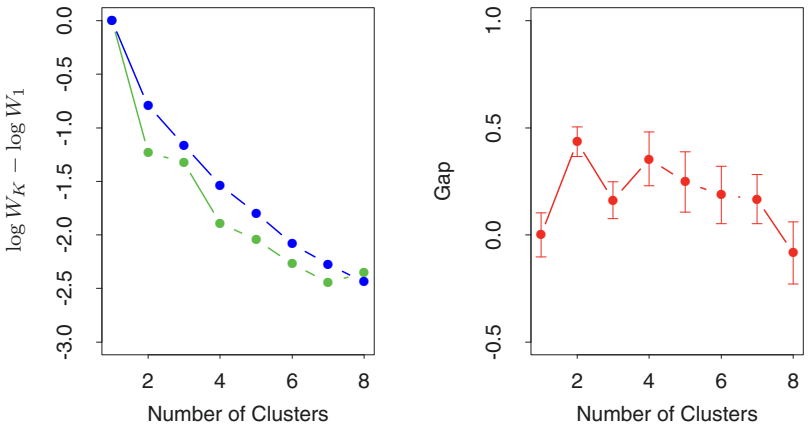

FIGURE 14.11. *(Left panel): observed (green) and expected (blue) values of* $\log W_K$ *for the simulated data of Figure 14.4. Both curves have been translated to equal zero at one cluster. (Right panel): Gap curve, equal to the difference between the observed and expected values of* $\log W_K$*. The Gap estimate* K^* *is the smallest* K *producing a gap within one standard deviation of the gap at* $K+1$*; here* $K^* = 2$*.*

This gives $K^* = 2$, which looks reasonable from Figure 14.4.

14.3.12 Hierarchical Clustering

The results of applying K-means or K-medoids clustering algorithms depend on the choice for the number of clusters to be searched and a starting configuration assignment. In contrast, hierarchical clustering methods do not require such specifications. Instead, they require the user to specify a measure of dissimilarity between (disjoint) *groups* of observations, based on the pairwise dissimilarities among the observations in the two groups. As the name suggests, they produce hierarchical representations in which the clusters at each level of the hierarchy are created by merging clusters at the next lower level. At the lowest level, each cluster contains a single observation. At the highest level there is only one cluster containing all of the data.

Strategies for hierarchical clustering divide into two basic paradigms: *agglomerative* (bottom-up) and *divisive* (top-down). Agglomerative strategies start at the bottom and at each level recursively merge a selected pair of clusters into a single cluster. This produces a grouping at the next higher level with one less cluster. The pair chosen for merging consist of the two groups with the smallest intergroup dissimilarity. Divisive methods start at the top and at each level recursively split one of the existing clusters at

that level into two new clusters. The split is chosen to produce two new groups with the largest between-group dissimilarity. With both paradigms there are $N-1$ levels in the hierarchy.

Each level of the hierarchy represents a particular grouping of the data into disjoint clusters of observations. The entire hierarchy represents an ordered sequence of such groupings. It is up to the user to decide which level (if any) actually represents a "natural" clustering in the sense that observations within each of its groups are sufficiently more similar to each other than to observations assigned to different groups at that level. The Gap statistic described earlier can be used for this purpose.

Recursive binary splitting/agglomeration can be represented by a rooted binary tree. The nodes of the trees represent groups. The root node represents the entire data set. The N terminal nodes each represent one of the individual observations (singleton clusters). Each nonterminal node ("parent") has two daughter nodes. For divisive clustering the two daughters represent the two groups resulting from the split of the parent; for agglomerative clustering the daughters represent the two groups that were merged to form the parent.

Most agglomerative and some divisive methods (when viewed bottom-up) possess a monotonicity property. That is, the dissimilarity between merged clusters is monotone increasing with the level of the merger. Thus the binary tree can be plotted so that the height of each node is proportional to the value of the intergroup dissimilarity between its two daughters. The terminal nodes representing individual observations are all plotted at zero height. This type of graphical display is called a *dendrogram*.

A dendrogram provides a highly interpretable complete description of the hierarchical clustering in a graphical format. This is one of the main reasons for the popularity of hierarchical clustering methods.

For the microarray data, Figure 14.12 shows the dendrogram resulting from agglomerative clustering with average linkage; agglomerative clustering and this example are discussed in more detail later in this chapter. Cutting the dendrogram horizontally at a particular height partitions the data into disjoint clusters represented by the vertical lines that intersect it. These are the clusters that would be produced by terminating the procedure when the optimal intergroup dissimilarity exceeds that threshold cut value. Groups that merge at high values, relative to the merger values of the subgroups contained within them lower in the tree, are candidates for natural clusters. Note that this may occur at several different levels, indicating a clustering hierarchy: that is, clusters nested within clusters.

Such a dendrogram is often viewed as a graphical summary of the data itself, rather than a description of the results of the algorithm. However, such interpretations should be treated with caution. First, different hierarchical methods (see below), as well as small changes in the data, can lead to quite different dendrograms. Also, such a summary will be valid only to the extent that the pairwise *observation* dissimilarities possess the hierar-

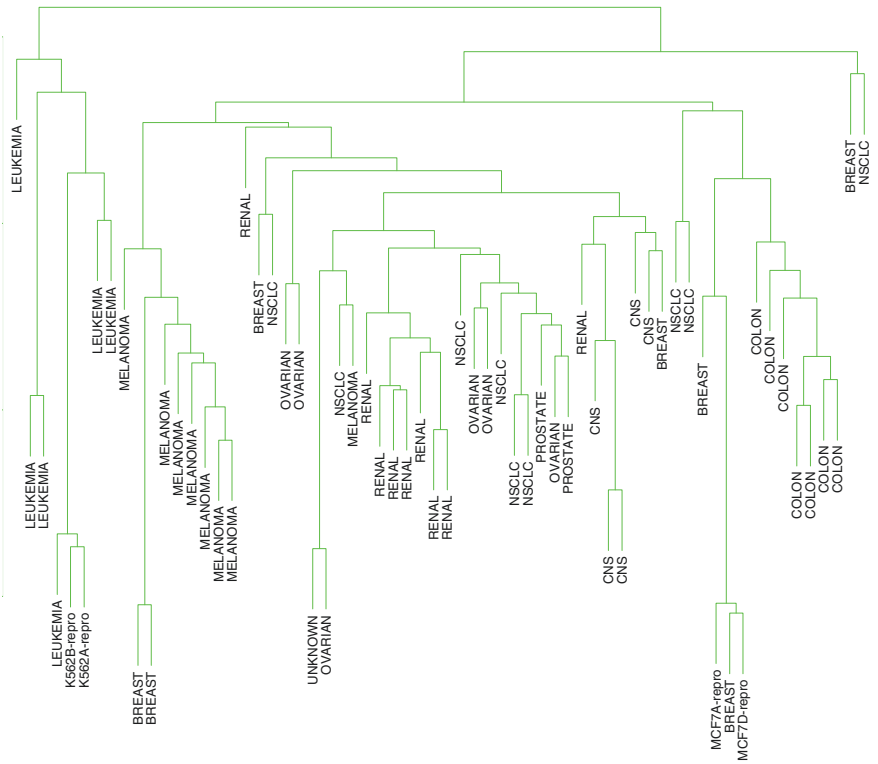

FIGURE 14.12. *Dendrogram from agglomerative hierarchical clustering with average linkage to the human tumor microarray data.*

chical structure produced by the algorithm. Hierarchical methods impose hierarchical structure whether or not such structure actually exists in the data.

The extent to which the hierarchical structure produced by a dendrogram actually represents the data itself can be judged by the *cophenetic correlation coefficient*. This is the correlation between the $N(N-1)/2$ pairwise observation dissimilarities $d_{ii'}$ input to the algorithm and their corresponding *cophenetic* dissimilarities $C_{ii'}$ derived from the dendrogram. The cophenetic dissimilarity $C_{ii'}$ between two observations (i, i') is the intergroup dissimilarity at which observations i and i' are first joined together in the same cluster.

The cophenetic dissimilarity is a very restrictive dissimilarity measure. First, the $C_{ii'}$ over the observations must contain many ties, since only $N-1$ of the total $N(N-1)/2$ values can be distinct. Also these dissimilarities obey the *ultrametric inequality*

$$C_{ii'} \leq \max\{C_{ik}, C_{i'k}\} \tag{14.40}$$

for any three observations (i, i', k). As a geometric example, suppose the data were represented as points in a Euclidean coordinate system. In order for the set of interpoint distances over the data to conform to (14.40), the triangles formed by all triples of points must be isosceles triangles with the unequal length no longer than the length of the two equal sides (Jain and Dubes, 1988). Therefore it is unrealistic to expect general dissimilarities over arbitrary data sets to closely resemble their corresponding cophenetic dissimilarities as calculated from a dendrogram, especially if there are not many tied values. Thus the dendrogram should be viewed mainly as a description of the *clustering* structure of the data as imposed by the particular algorithm employed.

Agglomerative Clustering

Agglomerative clustering algorithms begin with every observation representing a singleton cluster. At each of the $N-1$ steps the closest two (least dissimilar) clusters are merged into a single cluster, producing one less cluster at the next higher level. Therefore, a measure of dissimilarity between two clusters (groups of observations) must be defined.

Let G and H represent two such groups. The dissimilarity $d(G, H)$ between G and H is computed from the set of pairwise observation dissimilarities $d_{ii'}$ where one member of the pair i is in G and the other i' is in H. *Single linkage* (SL) agglomerative clustering takes the intergroup dissimilarity to be that of the closest (least dissimilar) pair

$$d_{SL}(G, H) = \min_{\substack{i \in G \\ i' \in H}} d_{ii'}. \tag{14.41}$$

This is also often called the *nearest-neighbor* technique. *Complete linkage* (CL) agglomerative clustering (*furthest-neighbor* technique) takes the intergroup dissimilarity to be that of the furthest (most dissimilar) pair

$$d_{CL}(G, H) = \max_{\substack{i \in G \\ i' \in H}} d_{ii'}. \tag{14.42}$$

Group average (GA) clustering uses the average dissimilarity between the groups

$$d_{GA}(G, H) = \frac{1}{N_G N_H} \sum_{i \in G} \sum_{i' \in H} d_{ii'} \tag{14.43}$$

where N_G and N_H are the respective number of observations in each group. Although there have been many other proposals for defining intergroup dissimilarity in the context of agglomerative clustering, the above three are the ones most commonly used. Figure 14.13 shows examples of all three.

If the data dissimilarities $\{d_{ii'}\}$ exhibit a strong clustering tendency, with each of the clusters being compact and well separated from others, then all three methods produce similar results. Clusters are compact if all of the

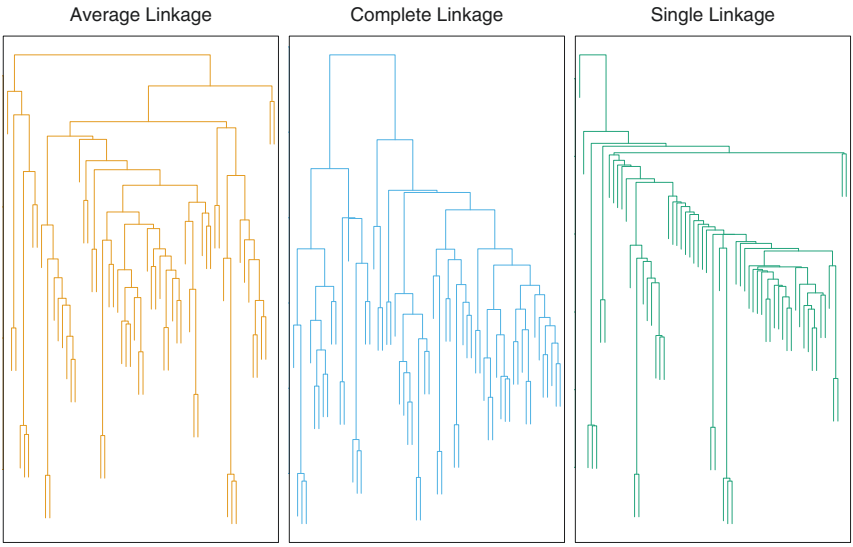

FIGURE 14.13. *Dendrograms from agglomerative hierarchical clustering of human tumor microarray data.*

observations within them are relatively close together (small dissimilarities) as compared with observations in different clusters. To the extent this is not the case, results will differ.

Single linkage (14.41) only requires that a single dissimilarity $d_{ii'}$, $i \in G$ and $i' \in H$, be small for two groups G and H to be considered close together, irrespective of the other observation dissimilarities between the groups. It will therefore have a tendency to combine, at relatively low thresholds, observations linked by a series of close intermediate observations. This phenomenon, referred to as *chaining*, is often considered a defect of the method. The clusters produced by single linkage can violate the "compactness" property that all observations within each cluster tend to be similar to one another, based on the supplied observation dissimilarities $\{d_{ii'}\}$. If we define the *diameter* D_G of a group of observations as the largest dissimilarity among its members

$$D_G = \max_{\substack{i \in G \\ i' \in G}} d_{ii'}, \tag{14.44}$$

then single linkage can produce clusters with very large diameters.

Complete linkage (14.42) represents the opposite extreme. Two groups G and H are considered close only if all of the observations in their union are relatively similar. It will tend to produce compact clusters with small diameters (14.44). However, it can produce clusters that violate the "closeness" property. That is, observations assigned to a cluster can be much

closer to members of other clusters than they are to some members of their own cluster.

Group average clustering (14.43) represents a compromise between the two extremes of single and complete linkage. It attempts to produce *relatively* compact clusters that are *relatively* far apart. However, its results depend on the numerical scale on which the observation dissimilarities $d_{ii'}$ are measured. Applying a monotone strictly increasing transformation $h(\cdot)$ to the $d_{ii'}$, $h_{ii'} = h(d_{ii'})$, can change the result produced by (14.43). In contrast, (14.41) and (14.42) depend only on the ordering of the $d_{ii'}$ and are thus invariant to such monotone transformations. This invariance is often used as an argument in favor of single or complete linkage over group average methods.

One can argue that group average clustering has a statistical consistency property violated by single and complete linkage. Assume we have attribute-value data $X^T = (X_1, \ldots, X_p)$ and that each cluster k is a random sample from some population joint density $p_k(x)$. The complete data set is a random sample from a mixture of K such densities. The group average dissimilarity $d_{GA}(G, H)$ (14.43) is an estimate of

$$\int \int d(x, x')\, p_G(x)\, p_H(x')\, dx\, dx', \tag{14.45}$$

where $d(x, x')$ is the dissimilarity between points x and x' in the space of attribute values. As the sample size N approaches infinity $d_{GA}(G, H)$ (14.43) approaches (14.45), which is a characteristic of the relationship between the two densities $p_G(x)$ and $p_H(x)$. For single linkage, $d_{SL}(G, H)$ (14.41) approaches zero as $N \to \infty$ independent of $p_G(x)$ and $p_H(x)$. For complete linkage, $d_{CL}(G, H)$ (14.42) becomes infinite as $N \to \infty$, again independent of the two densities. Thus, it is not clear what aspects of the population distribution are being estimated by $d_{SL}(G, H)$ and $d_{CL}(G, H)$.

Example: Human Cancer Microarray Data (Continued)

The left panel of Figure 14.13 shows the dendrogram resulting from average linkage agglomerative clustering of the samples (columns) of the microarray data. The middle and right panels show the result using complete and single linkage. Average and complete linkage gave similar results, while single linkage produced unbalanced groups with long thin clusters. We focus on the average linkage clustering.

Like K-means clustering, hierarchical clustering is successful at clustering simple cancers together. However it has other nice features. By cutting off the dendrogram at various heights, different numbers of clusters emerge, and the sets of clusters are nested within one another. Secondly, it gives some partial ordering information about the samples. In Figure 14.14, we have arranged the genes (rows) and samples (columns) of the expression matrix in orderings derived from hierarchical clustering.

Note that if we flip the orientation of the branches of a dendrogram at any merge, the resulting dendrogram is still consistent with the series of hierarchical clustering operations. Hence to determine an ordering of the leaves, we must add a constraint. To produce the row ordering of Figure 14.14, we have used the default rule in S-PLUS: at each merge, the subtree with the tighter cluster is placed to the left (toward the bottom in the rotated dendrogram in the figure.) Individual genes are the tightest clusters possible, and merges involving two individual genes place them in order by their observation number. The same rule was used for the columns. Many other rules are possible—for example, ordering by a multidimensional scaling of the genes; see Section 14.8.

The two-way rearrangement of Figure 14.14 produces an informative picture of the genes and samples. This picture is more informative than the randomly ordered rows and columns of Figure 1.3 of Chapter 1. Furthermore, the dendrograms themselves are useful, as biologists can, for example, interpret the gene clusters in terms of biological processes.

Divisive Clustering

Divisive clustering algorithms begin with the entire data set as a single cluster, and recursively divide one of the existing clusters into two daughter clusters at each iteration in a top-down fashion. This approach has not been studied nearly as extensively as agglomerative methods in the clustering literature. It has been explored somewhat in the engineering literature (Gersho and Gray, 1992) in the context of compression. In the clustering setting, a potential advantage of divisive over agglomerative methods can occur when interest is focused on partitioning the data into a relatively *small* number of clusters.

The divisive paradigm can be employed by recursively applying any of the combinatorial methods such as K-means (Section 14.3.6) or K-medoids (Section 14.3.10), with $K = 2$, to perform the splits at each iteration. However, such an approach would depend on the starting configuration specified at each step. In addition, it would not necessarily produce a splitting sequence that possesses the monotonicity property required for dendrogram representation.

A divisive algorithm that avoids these problems was proposed by Macnaughton Smith et al. (1965). It begins by placing all observations in a single cluster G. It then chooses that observation whose average dissimilarity from all the other observations is largest. This observation forms the first member of a second cluster H. At each successive step that observation in G whose average distance from those in H, minus that for the remaining observations in G is largest, is transferred to H. This continues until the corresponding difference in averages becomes negative. That is, there are no longer any observations in G that are, on average, closer to those in H. The result is a split of the original cluster into two daughter clusters,

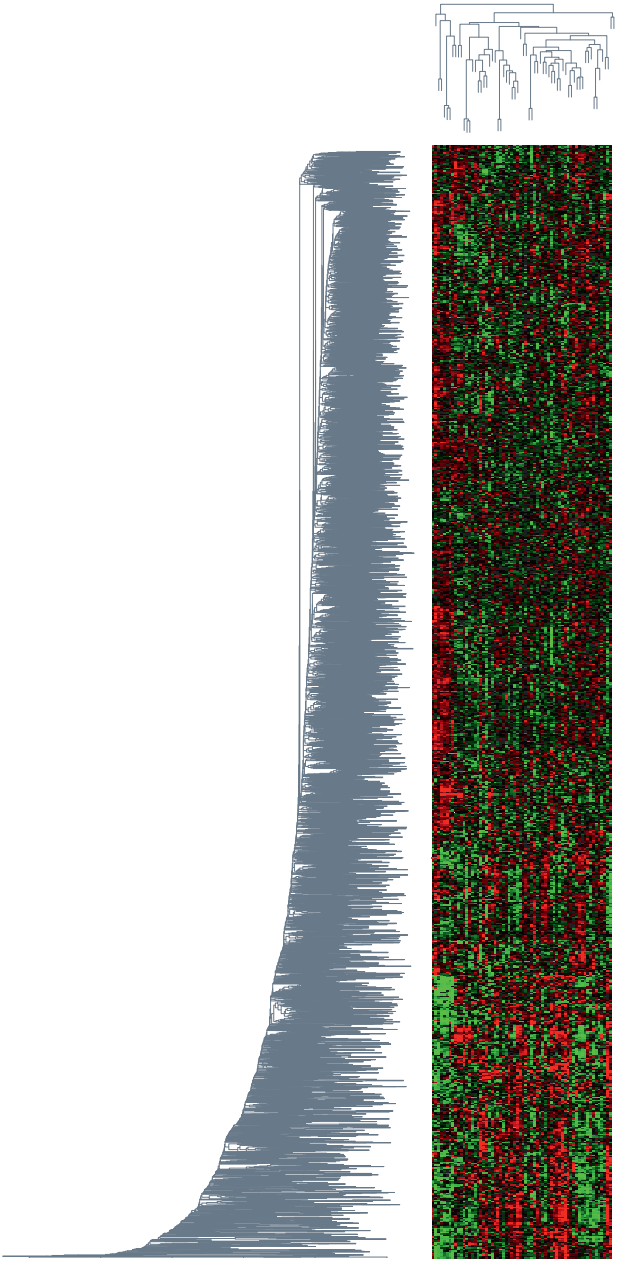

FIGURE 14.14. *DNA microarray data: average linkage hierarchical clustering has been applied independently to the rows (genes) and columns (samples), determining the ordering of the rows and columns (see text). The colors range from bright green (negative, under-expressed) to bright red (positive, over-expressed).*

the observations transferred to H, and those remaining in G. These two clusters represent the second level of the hierarchy. Each successive level is produced by applying this splitting procedure to one of the clusters at the previous level. Kaufman and Rousseeuw (1990) suggest choosing the cluster at each level with the largest diameter (14.44) for splitting. An alternative would be to choose the one with the largest average dissimilarity among its members

$$\bar{d}_G = \frac{1}{{N_G}^2} \sum_{i \in G} \sum_{i' \in G} d_{ii'}.$$

The recursive splitting continues until all clusters either become singletons or all members of each one have zero dissimilarity from one another.

14.4 Self-Organizing Maps

This method can be viewed as a constrained version of K-means clustering, in which the prototypes are encouraged to lie in a one- or two-dimensional manifold in the feature space. The resulting manifold is also referred to as a *constrained topological map*, since the original high-dimensional observations can be mapped down onto the two-dimensional coordinate system. The original SOM algorithm was online—observations are processed one at a time—and later a batch version was proposed. The technique also bears a close relationship to *principal curves and surfaces*, which are discussed in the next section.

We consider a SOM with a two-dimensional rectangular grid of K prototypes $m_j \in \mathbb{R}^p$ (other choices, such as hexagonal grids, can also be used). Each of the K prototypes are parametrized with respect to an integer coordinate pair $\ell_j \in \mathcal{Q}_1 \times \mathcal{Q}_2$. Here $\mathcal{Q}_1 = \{1, 2, \ldots, q_1\}$, similarly $\mathcal{Q}_2$, and $K = q_1 \cdot q_2$. The m_j are initialized, for example, to lie in the two-dimensional principal component plane of the data (next section). We can think of the prototypes as "buttons," "sewn" on the principal component plane in a regular pattern. The SOM procedure tries to bend the plane so that the buttons approximate the data points as well as possible. Once the model is fit, the observations can be mapped down onto the two-dimensional grid.

The observations x_i are processed one at a time. We find the closest prototype m_j to x_i in Euclidean distance in $\mathbb{R}^p$, and then for all neighbors m_k of m_j, move m_k toward x_i via the update

$$m_k \leftarrow m_k + \alpha(x_i - m_k). \tag{14.46}$$

The "neighbors" of m_j are defined to be all m_k such that the distance between ℓ_j and ℓ_k is small. The simplest approach uses Euclidean distance, and "small" is determined by a threshold r. This neighborhood always includes the closest prototype m_j itself.