9 *Descriptive Modeling*

9.1 Introduction

In earlier chapters we explained what is meant, in the context of data mining, by the terms *model* and *pattern*. A model is a high-level description, summarizing a large collection of data and describing its important features. Often a model is *global* in the sense that it applies to all points in the measurement space. In contrast, a pattern is a local description, applying to some subset of the measurement space, perhaps showing how just a few data points behave or characterizing some persistent but unusual structure within the data. Examples would be a mode (peak) in a density function or a small set of outliers in a scatter plot.

Earlier chapters distinguished between models and patterns, and also between descriptive and predictive models. A descriptive model presents, in convenient form, the main features of the data. It is essentially a summary of the data, permitting us to study the most important aspects of the data without their being obscured by the sheer size of the data set. In contrast, a predictive model has the specific objective of allowing us to predict the value of some target characteristic of an object on the basis of observed values of other characteristics of the object.

This chapter is concerned with descriptive models, presenting outlines of several algorithms for finding descriptive models that are important in data mining contexts. Chapters 10 and 11 will describe predictive models, and chapter 13 will describe descriptive patterns.

We have already noted that data mining is usually concerned with building empirical models—models that are not based on some underlying theory about the mechanism through which the data arose, but that are simply a description of the observed data. The fundamental objective is to produce insight and understanding about the structure of the data, and to enable us to see its important features. Beyond this, of course, we hope to discover unsuspected structure as well as structure that is interesting and valuable in some sense. A good model can also be thought of as *generative* in the sense that data generated according to the model will have the same characteristics as the real data from which the model was produced. If such synthetically generated data have features not possessed by the original data, or do not possess features of the original data (such as, for example, correlations between variables), then the model is a poor one: it is failing to summarize the data adequately.

This chapter focuses on specific techniques and algorithms for fitting descriptive models to data. It builds on many of the ideas introduced in earlier chapters: the principles of uncertainty (chapter 4), decomposing data mining algorithms into basic components (chapter 5), and the general principles underlying model structures, score functions, and parameter and model search (chapters 6, 7, and 8, respectively).

There are, in fact, many different types of model, each related to the others in various ways (special cases, generalizations, different ways of looking at the same structure, and so on). We cannot hope to examine all possible models types in detail in a single chapter. Instead we will look at just some of the more important types, focusing on methods for density estimation and cluster analysis in particular. The reader is alerted to the fact that are other descriptive techniques in the literature (techniques such as structural equation modeling or factor analysis for example) that we do not discuss here.

One point is worth making at the start. Since we are concerned here with global models, with structures that are representative of a mass of objects in some sense, then we do not need to worry about failing to detect just a handful of objects possessing some property; that is, in this chapter we are not concerned with patterns. This is good news from the point of view of scalability: as we discussed in chapter 4, we can, for example, take a (random) sample from the data set and still hope to obtain good results.

9.2 Describing Data by Probability Distributions and Densities

9.2.1 Introduction

For data that are drawn from a larger population of values, or data that can be regarded as being drawn from such a larger population (for example, because the measurements have associated measurement error), describing data in terms of their underlying distribution or density function is a fundamental *descriptive* strategy. Adopting our usual notation of a p-dimensional data matrix, with variables $X_1, \ldots, X_p$, our goal is to model the joint distribution or density $f(X_1, \ldots, X_p)$ as first encountered in chapter 4. For convenience, we will refer to "densities" in this discussion, but the ideas apply to discrete as well as to continuous X variables.

The joint density in a certain sense provides us with complete information about the variables $X_1, \ldots, X_p$. Given the joint density, we can answer any question about the relationships among any subset of variable; for example, are X_3 and X_7 independent? Thus, we can answer questions about the conditional density of some variables given others; for example, what is the probability distribution of X_3 given the value of X_7, $f(x_3 \mid x_7)$?.

There are many practical situations in which knowing the joint density is useful and desirable. For example, we may be interested in the *modes* of the density (for real-valued Xs). Say we are looking at the variables income and credit-card spending for a data set of n customers at a particular bank. For large n, in a scatterplot we will just see a mass of points, many overlaid on top of each other. If instead we estimate the joint density f(income, spending) (where we have yet to describe how this would be done), we get a density function of the two dimensions that could be plotted as a contour map or as a three-dimensional display with the density function being plotted in the third dimension. The estimated joint density would in principle impart useful information about the underlying structure and patterns present in the data. For example, the presence of peaks (modes) in the density function could indicate the presence of subgroups of customers. Conversely, gaps, holes, or valleys might indicate regions where (for one reason or another) this particular bank had no customers. And the overall shape of the density would provide an indication of how income and spending are related, for this population of customers.

A quite different example is given by the problem of generating approximate answers to queries for large databases (also known as *query selectivity estimation*). The task is the following: given a query (that is, a condition that the observations must satisfy), estimate the fraction of rows that satisfy this condition (the *selectivity* of the query). Such estimates are needed in query optimization in database systems, and a single query optimization task might need hundreds of such estimates. If we have a good approximation for the joint distribution of the data in the database, we can use it to obtain approximate selectivities in a computationally efficient manner.

Thus, the joint density is fundamental and we will need to find ways to estimate and conveniently summarize it (or its main features).

9.2.2 Score Functions for Estimating Probability Distributions and Densities

As we have noted in earlier chapters, the most common score function for estimating the parameters of probability functions is the *likelihood* (or, equivalently by virtue of the monotonicity of the log transform, the *log-likelihood*). As a reminder, if the probability function of random variables $\mathbf{X}$ is $f(\mathbf{x};\theta)$, where θ are the parameters that need to be estimated, then the log-likelihood is $\log f(D|\theta)$ where $D = \{\mathbf{x}(1), \ldots, \mathbf{x}(n)\}$ is the observed data. Making the common assumption that that the separate rows of the data matrix have arisen independently, this becomes

$$S_L(\theta) = -\sum_{i=1}^{n} \log f(\mathbf{x}(i);\theta). \tag{9.1}$$

If f has a simple functional form (for example, if it has the form of the single univariate distributions outlined in the appendix) then this score function can usually be minimized explicitly, producing a closed form estimator for the parameters θ. However, if f is more complex, iterative optimization methods may be required.

Despite its importance, the likelihood may not always be an adequate or appropriate measure for comparing models. In particular, when models of different complexity (for example, Normal densities with covariance structures parameterized in terms of different numbers of parameters) are compared then difficulties may arise. For example, with a nested series of models in which higher-level models include lower-level ones as special cases, the more flexible higher level models will always have a greater likelihood. This will come as no surprise. The likelihood score function is a measure of how well the model fits the data, and more flexible models necessarily fit the data no worse (and usually better) than a nested less flexible model. This means that likelihood will be appropriate in situations in which we are using it as a score function to summarize a complete body of data (since then our aim is simply closeness of fit between the simplifying description and the raw data) but not if we are using it to select a single model (from a set of candidate model structures) to apply it to a sample of data from a larger population (with the implicit aim being to generalize beyond the data actually observed). In the latter case, we can solve the problem by modifying the likelihood to take the complexity of the model into account. We discussed this in detail in chapter 7, where we outlined several score functions based on adding an extra term to the likelihood that penalizes model complexity. For example, the BIC (Bayesian Information Criterion) score function was defined as:

$$S_{BIC}(M_k) = 2S_L(\hat{\theta}_k; M_k) + d_k \log n, \qquad 1 \leq k \leq K, \tag{9.2}$$

where d_k is the number of parameters in model M_k and $S_L(\hat{\theta}_k; M_k)$ is the minimizing value of the negative log-likelihood (achieved at $\hat{\theta}_k$).

Alternatively, also as discussed in chapter 7, we can calculate the score using an independent sample of data, producing an "out-of-sample" evaluation. Thus the *validation log-likelihood* (or "holdout log-likelihood") is defined as

$$S_{vl}(M_k) = \sum_{\mathbf{x} \in D_v} \log f_{M_k}(\mathbf{x}|\hat{\theta}), 1 \leq k \leq K, \tag{9.3}$$

where the points $\mathbf{x}$ are from the validation data set D_v, the parameters $\hat{\theta}$ were estimated (for example, via maximum likelihood) on the disjoint training data $D_t = D \setminus D_v$, and there are K models under consideration.

9.2.3 Parametric Density Models

We pointed out, in chapter 6, that there are two general classes of density function model structures: parametric and nonparametric. *Parametric models* assume a particular functional form (usually relatively simple) for the density function, such as a uniform distribution, a Normal distribution, an exponential distribution, a Poisson distribution, and so on (see Appendix A for more details on some of these common densities and distributions). These distribution functions are often motivated by underlying causal models of generic data-generating mechanisms. Choice of what might be an appropriate density function should be based on knowledge of the variable being measured (for example, the knowledge that a variable such as income can only be positive should be reflected in the choice of the distribution adopted to model it). Parametric models can often be characterized by a relatively small number of parameters. For example, the p-dimensional Normal distribution is defined as

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{p}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)}, \tag{9.4}$$

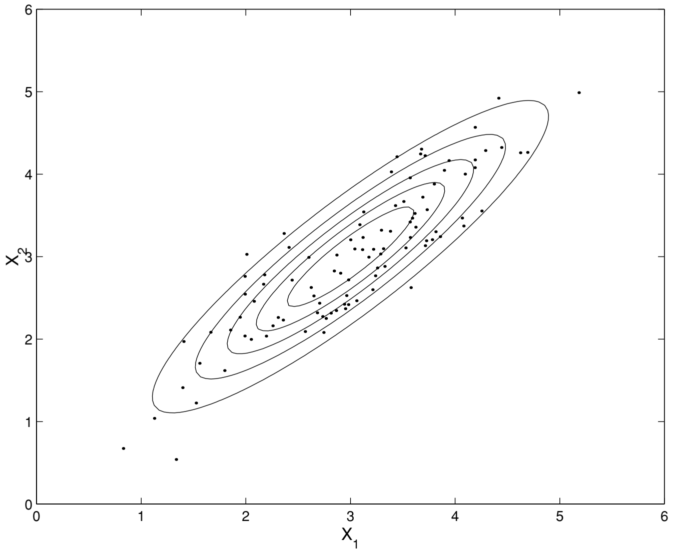


Figure 9.1 Illustration of the density contours for a two-dimensional Normal density function, with mean $[3, 3]$ and covariance matrix $\Sigma = \begin{pmatrix} 1.0 & 0.9 \\ 0.9 & 1.0 \end{pmatrix}$. Also shown are 100 data points simulated from this density.

where Σ is the $p \times p$ covariance matrix of the X variables, $|\Sigma|$ is the determinant of this matrix, and μ is the p-dimensional vector mean of the Xs. The *parameters* of the model are the mean vector and the covariance matrix (thus, $p + p(p+1)/2$ parameters in all).

The multivariate Normal (or Gaussian) distribution is particularly important in data analysis. For example, because of the central limit theorem, under fairly broad assumptions the mean of N independent random variables (each from any distribution) tends to have a Normal distribution. Although the result is asymptotic in nature, even for relatively small values of N (e.g., $N = 10$) the sample mean will typically be quite Normal. Thus, if a measurement can be thought of as being made up of the sum of multiple relatively independent causes, the Normal model is often a reasonable model to adopt.

The functional form of the multivariate Normal model in equation 9.4 is less formidable than it looks. The exponent, $(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)$, is a scalar value (a quadratic form) known as the *Mahalanobis distance* between the data point $\mathbf{x}$ and the mean μ, denoted as $r^2_\Sigma(\mathbf{x}, \mu)$. This is a generalization of standard Euclidean distance that takes into account (through the covariance matrix Σ) correlations in p-space when distance is calculated. The denominator in equation 9.4 is simply a normalizing constant (call it C) to ensure that the function integrates to 1 (that is, to ensure it is a true probability density function). Thus, we can write our Normal model in significantly simplified form as

$$f(\mathbf{x}) = \frac{1}{C} e^{-r_\Sigma^2(\mathbf{x},\mu)/2} . \tag{9.5}$$

If we were to plot (say for $p = 2$) all of the points $\mathbf{x}$ that have the same fixed values of $r_\Sigma^2(\mathbf{x}, \mu)$, (or equivalently, all of the points $\mathbf{x}$ that like on iso-density contours $f(\mathbf{x}) = c$ for some constant c), we would find that they trace out an ellipse in 2-space (more generally, a hyperellipsoid in p-space), where the ellipse is centered at μ. That is, the contours describing the multivariate Normal distribution are ellipsoidal, with height falling exponentially from the center as a function of $r_\Sigma^2(\mathbf{x}, \mu)$. Figure 9.1 provides a simple illustration in two dimensions. The eccentricity and orientation of the elliptical contours is determined by the form of Σ. If Σ is a multiple of the identity matrix (all variables have the same variance and are uncorrelated) then the contours are circles. If Σ is a diagonal matrix, but with different variance terms on the diagonals, then the axes of the elliptical contours are parallel to the variable axes and the contours are elongated along the variable axes with greater variance. Finally, if some of the variables are highly correlated, the (hyper) elliptical contours will tend to be elongated along vectors defined as linear combinations of these variables. In figure 9.1, for example, the two variables X_1 and X_2 are highly correlated, and the data are spread out along the line defined by the linear combination $X_1 + X_2$.

For high-dimensional data (large p) the number of parameters in the Normal model will be dominated by the $O(p^2)$ covariance terms in the covariance matrix. In practice we may not want to model all of these covariance terms explicitly, since for large p and finite n (the number of data points available) we may not get very reliable estimates of many of the covariance terms. We could, for example, instead assume that the variables are independent, which is equivalent in the Normal case to assuming that the covariance matrix has a diagonal structure (and, hence, has only p parameters). (Note that if we assume that Σ is diagonal it is easy to show that the p-dimensional multivariate Normal density factors into a product of p univariate Normal distributions, a necessary and sufficient condition for independence of the p variables.) An even more extreme assumption would be to assume that

$\Sigma = \sigma^2 I$, where I is the identity matrix—that is, that the data has the same variance for all p variables as well as being independent.

Independence is a highly restrictive assumption. A less restrictive assumption would be that the covariance matrix had a block diagonal structure: we assume that there are groups of variables (the "blocks") that are dependent, but that variables are independent across the groups. In general, all sorts of assumptions may be possible, and it is important, in practice, to test the assumptions. In this regard, the multivariate Normal distribution has the attractive property that two variables are conditionally independent, given the other variables, if and only if the corresponding element of the inverse of the covariance matrix is zero. This means that the inverse covariance matrix Σ^{-1} reveals the pattern of relationships between the variables. (Or, at least, it does in principle: in fact, of course, it will be necessary to decide whether a small value in the inverse covariance matrix is sufficiently small to be regarded as zero.) It also means that we can hypothesize a graphical model in which there are no edges linking the nodes corresponding to variables that have a small value in this inverse matrix (we discussed graphical models in chapter 6).

It is important to test the assumptions made in a model. Specific statistical goodness-of-fit tests are often available, but even simple eyeballing can be revealing. The simple histogram, or one of its more sophisticated cousins outlined in chapter 3, can immediately reveal constraints on permissible ranges (for example, the non-negativity of income noted above), lack of symmetry, and so on. If the assumptions are not justified, then analysis of some transformation of the raw scores may be appropriate. Unfortunately, there are no hard-and-fast rules about whether or not an assumption is justified. Slight departures may well be unimportant—but it will depend on the problem. This is part of the art of data mining. In many situations in which the distributional assumptions break down we can obtain perfectly legitimate estimates of parameters, but statistical tests are invalid. For example, we can physically fit a regression model using the least squares score function, whether or not the errors are Normally distributed, but hypothesis tests on the estimated parameters may well not be accurate. This might matter during the model building process—in helping to decide whether or not to include a variable—but it may not matter for the final model. If the final model is good for its purpose (for example, predictive accuracy in regression) that is sufficient justification for it to be adopted.

Fitting a p-dimensional Normal model is quite easy. Maximum likelihood (or indeed Bayesian) estimation of each of the means and the covariance terms can be defined in closed form (as discussed in chapter 4), and takes only $O(n)$ steps for each parameter, so $O(np^2)$ in total. Other well-known parametric models (such as those defined in the appendix) also usually possess closed-form parameter solutions that can be calculated by a single pass through the data.

The Normal model structure is a relatively simple and constrained model. It is unimodal and symmetric about the axes of the ellipse. It is parametrized completely in terms of its mean vector and covariance matrix. However, it follows from this that nonlinear relationships cannot be captured, nor can any form of multimodality or grouping. The mixture models of the next section provide a flexible framework for modeling such structures. The reader should also note that although the Normal model is probably the most widely-used parametric model in practice, there are many other density functions with different "shapes" that are very useful for certain applications (e.g., the exponential model, the log-normal, the Poisson, the Gamma: the interested reader is referred to the appendix). The multivariate t-distribution is similar in form to the multivariate Normal but allows for longer tails, and is found useful in practical problems where more data can often occur in the tails than a Normal model would predict.

9.2.4 Mixture Distributions and Densities

In chapter 6 we saw how simple parametric models could be generalized to allow *mixtures* of components—that is, linear combinations of simpler distributions. This can be viewed as the next natural step in complexity in our discussion of density modeling: namely, the generalization from parametric distributions to weighted linear combinations of such functions, providing a general framework for generating more complex density and distribution models as combinations of simpler ones. Mixture models are quite useful in practice for modeling data when we are not sure what specific parametric form is appropriate (later in this chapter we will see how such mixture models can also be used for the task of *clustering*).

It is quite common in practice that a data set is *heterogeneous* in the sense that it represents multiple different subpopulations or groups, rather than one single homogeneous group. Heterogeneity is particularly prevalent in very large data sets, where the data may represent different underlying phenomena that have been collected to form one large data set. To illustrate this point, consider figure 3.1 in chapter 3. This is a histogram of the number of weeks owners of a particular credit card used that card to make supermarket purchases in 1996. As we pointed out there, the histogram appears to be bimodal, with a large and obvious mode to the left and a smaller, but nevertheless possibly important mode to the right. An initial stab at a model for such data might be that it follows a Poisson distribution (despite being bounded above by 52), but this would not have a sufficiently heavy tail and would fail to pick up the right-hand mode. Likewise, a binomial model would also fail to follow the right-hand mode. Something more sophisticated and flexible is needed. An obvious suggestion here is that the empirical distribution should be modeled by a theoretical distribution that has two components. Perhaps there are two kinds of people: those who are unlikely to use their credit card in a supermarket and those who do so most weeks. The first set of people could be modeled by a Poisson distribution with a small probability. The second set could be modeled by a reversed Poisson distribution with its mode around 45 or 46 weeks (the position of the mode would be a parameter to be estimated in fitting the model to the data). This leads us to an overall distribution of the form

$$f(x) = \pi \frac{(\lambda_1)^x e^{-\lambda_1}}{x!} + (1-\pi) \frac{(\lambda_2)^{52-x} e^{-\lambda_2}}{(52-x)!}, \tag{9.6}$$

where x is the value of the random variable X taking values between 0 and 52 (indicating how many weeks a year a person uses their card in a supermarket), and $\lambda_1 > 0, \lambda_2 > 0$ are parameters of the two component Poisson models. Here π is the probability that a person belongs to the first group, and, given this, the expression $\lambda_1^x e^{-\lambda_1}/x!$ gives the probability that this person will use their card x times in the year. Likewise, $1-\pi$ is the probability that this person belong to the second group and $\lambda_2^{52-x} e^{-\lambda_2}/(52-x)!$ is the conditional probability that such a person will use their card x times in the year.

One way to think about this sort of model is as a two-stage generative process for a particular individual. In the first step there is a probability π (and $1-\pi$) that the individual comes from one group or the other. In the second step, an observation x is generated for that person according to the component distribution he or she was assigned to in the first step.

Equation 9.6 is an example of a *finite mixture distribution,* where the overall model $f(x)$ is a weighted linear combination of a finite number of component distributions (in this case just two). Clearly it leads to a much more flexible model than a simple single Poisson distribution—at the very least, it involves three parameters instead of just one. However, by virtue of the argument that led to it, it may also be a more realistic description of what is underlying the data. These two aspects—the extra flexibility of the models consequent on the larger number of parameters and arguments based on suspicion of a heterogeneous underlying population—mean that mixture models are widely used for modeling distributions that are more complicated than simple standard forms.

The general form of a mixture distribution (for multivariate $\mathbf{x}$) is

$$f(\mathbf{x}) = \sum_{k=1}^{K} \pi_k f_k(\mathbf{x}; \theta_k), \tag{9.7}$$

where π_k is the probability that an observation will come from the kth component (the so-called kth *mixing proportion* or *weight*), K is the number of components, $f_k(x; \theta_k)$ is the distribution of the kth component, and θ_k is the vector of parameters describing the kth component (in the Poisson mixture example above, each θ_k consisted of a single parameter λ_k). In most applications the component distributions f_k have the same form, but there are situations where this is not the case. The most widely used form of mixture distribution has Normal components. Note that the mixing proportions π_k must lie between 0 and 1 and sum to 1.

Some examples of the many practical situations in which mixture distributions might be expected on theoretical grounds are the length distribution of fish (since they hatch at a specific time of the year), failure data (where there may be different causes of failure, and each cause results in a distribution of failure times), time to death, and the distribution of characteristics of heterogeneous populations of people (e.g., heights of males and females).

9.2.5 The EM Algorithm for Mixture Models

Unlike the simple parametric models discussed earlier in this chapter, there is generally no direct closed-form technique for maximizing the likelihood score function when the underlying model is a mixture model, given a data set $D = \{\mathbf{x}(1), \ldots, \mathbf{x}(n)\}$. This is easy to see by writing out the log-likelihood for a mixture model—we get a sum of terms such as $\log(\sum_k \pi_k f_k(\mathbf{x}; \theta_k))$, leading to a nonlinear optimization problem (unlike, for example, the closed form solutions for the multivariate Normal model).

Over the years, many different methods have been applied in estimating the parameters of mixture distributions given a particular mixture form. One of the more widely used modern methods in this context is the EM approach. As discussed in chapter 8, this can be viewed as a general iterative optimization algorithm for maximizing a likelihood score function given a probabilistic model with missing data. In the present case, the mixture model can be regarded as a distribution in which the class labels are missing. If we knew these labels, we could get closed-form estimates for the parameters of each component by partitioning the data points into their respective groups. However, since we do not know the origin of each data point, we must simultaneously try to learn which component a data point originated from and the parameters of these components. This "chicken-and-egg" problem is neatly solved by the EM algorithm; it starts with some guesses at the parameter values for each component, then calculates the probability that each data point came from one of the K components (this is known as the E-step), calculates new parameters for each component given these probabilistic memberships (this is the M-step, and can typically be carried out in closed form), recalculates the probabilistic memberships, and continues on in this manner until the likelihood converges. As discussed in chapter 8, despite the seemingly heuristic nature of the algorithm, it can be shown that for each EM-step the likelihood can only increase, thus guaranteeing (under fairly broad conditions) convergence of the method to at least a local maximum of the likelihood as a function of the parameter space.

The complexity of the EM algorithm depends on the complexity of the E and M steps at each iteration. For multivariate normal mixtures with K components the computation will be dominated by the calculation of the K covariance matrices during the M-step at each iteration. In p dimensions, with K clusters, there are $O(Kp^2)$ covariance parameters to be estimated, and each of these requires summing over n data points and membership weights, leading to a $O(Kp^2n)$ time-complexity per step. For univariate mixtures (such as the Poisson above) we get $O(Kn)$. The space-complexity is typically $O(Kn)$ to store the K membership probability vectors for each of the n data points $\mathbf{x}(i)$. However, for large n, we often need not store the $n \times K$ membership probability matrix explicitly, since we may be able to calculate the parameter estimates during each M-step *incrementally* via a single pass through the n data points.

EM often provides a large increase in likelihood over the first few iterations and then can slowly converge to its final value; however the likelihood function as a function of iterations need not be concave. For example, figure 9.2 illustrates the convegence of the log-likelihood as a function of the EM iteration number, for a problem involving fitting Gaussian mixtures to a two-dimensional medical data set (that we will later discuss in more detail in section 9.6). For many data sets and models we can often find a reasonable

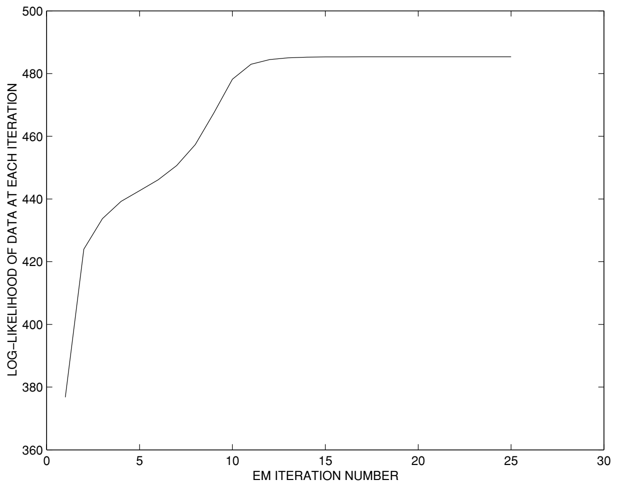


Figure 9.2 The log-likelihood of the red-blood cell data under a two-component Normal mixture model (see figure 9.11) as a function of iteration number.

solution in only 5 to 20 iterations of the algorithm. Each solution provided by EM is of course a function of where one started the search (since it is a local search algorithm), and thus, multiple restarts from randomly chosen starting points are a good idea to try to avoid poor local maxima. Note that as either (or both) K and p increase, the number of local maxima of the likelihood can increase greatly as the dimensionality of the parameter space scales accordingly.

Sometimes caution has to be exercised with maximum likelihood estimates of mixture distributions. For example, in a normal mixture, if we put the mean of one component equal to one of the sample points and let its standard deviation tend to zero, the likelihood will increase without limit. The maximum likelihood solution in this case is likely to be of limited value. There are various ways around this. The largest finite value of the likelihood might be chosen to give the estimated parameter values. Alternatively, if the standard deviations are constrained to be equal, the problem does not arise. A more general solution is to set up the problem in a Bayesian context, with priors on the parameters, and maximize the MAP score function (for example) instead of the likelihood. Here the priors provide a framework for "biasing" the score function (the MAP score function) away from problematic regions in parameter space in a principled manner. Note that the EM algorithm generalizes easily from the case of maximizing likelihood to maximizing MAP (for example, we replace the M-step with an MAP-step, and so forth).

Another problem that can arise is due to lack of identifiability. A family of mixture distributions is said to be *identifiable* if and only if the fact that two members of the family are equal,

$$\sum_{k=1}^{c} \pi_k f(x; \theta_k) = \sum_{j=1}^{c'} \pi'_j f(x; \theta'_j), \tag{9.8}$$

implies that $c = c'$, and that for all k there is some j such that $\pi_k = \pi'_j$ and $\theta_k = \theta'_j$. If a family is not identifiable, then two different members of it may be indistinguishable, which can lead to problems in estimation.

Nonidentifiability is more of a problem with discrete distributions than continuous ones because, with m categories, only $m - 1$ independent equations can be set up. For example, in the case of a mixture of several Bernoulli components, there is effectively only a single piece of information available in the data, namely, the proportion of 1s that occur in the data. Thus, there is no way of estimating the proportions that are separately due to each component Bernoulli, or the parameters of those components.

9.2.6 Nonparametric Density Estimation

In chapter 3 we briefly discussed the idea of estimating a density function by taking a local data-driven weighted average of x measurements about the point of interest (the so-called "kernel density" method). For example, a *histogram* is a relatively primitive version of this idea, in which we simply count the number of points that fall in certain bins. Our estimate for the density is the number of points in a given bin, appropriately scaled. The histogram is problematic as a model structure for densities for a number of reasons. It provides a nonsmooth estimate of what is often presumed to be truly a smooth function, and it is not obvious how the number of bins, bin locations, and widths should be chosen. Furthermore, these problems are exacerbated when we move beyond the one-dimensional histogram to a p-dimensional histogram. Nonetheless, for very large data sets and small p (particularly $p = 1$), the bin widths can be made quite small, and the resulting density estimate may still be relatively smooth and insensitive to the exact location or width of the bins. With large data sets it always a good idea to look at the histograms (with a large number of bins) for each variable, since the histogram can provide a wealth of information on outliers, multimodality, skewness, tail behavior, and so forth (recall the example of the Pima Indians blood pressure data in chapter 3, where the histogram clearly indicated the presence of some rather suspicious values at zero).

A more general model structure for local densities is to define the density at any point $\mathbf{x}$ as being proportional to a weighted sum of all points in the training data set, where the weights are defined by an appropriately chosen *kernel function*. For the one-dimensional case we have (as defined in chapter 3)

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} w_i, \quad w_i = K\left(\frac{x - x(i)}{h}\right), \tag{9.9}$$

where $f(x)$ is the kernel density estimate at a query point x, $K(t)$ is the kernel function (for example, $K(t) = 1 - |t|, t \leq 1; K(t) = 0$ otherwise) and h is the *bandwidth* of the kernel. Intuitively, the density at x is proportional to the sum of weights evaluated at x, which in turn depend on the proximity of the n points in the training data to x. As with nonparametric regression (discussed in chapter 6), the model is not defined explicitly, but is determined implicitly by the data and the kernel function. The approach is "memory-based" in the sense that all of the data points are retained in the model; that is, no summarization occurs. For very large data sets of course this may be impractical from a computational and storage viewpoint.

In one dimension, the kernel function K is usually chosen as a smooth unimodal function (such as a Normal or triangular distribution) that integrates to 1; the precise shape is typically not critical. As in regression, the bandwidth h plays the role of determining how smooth the model is. If h is relatively large, then the kernel is relatively wide so that many points receive significant weight in the sum and the estimate of the density is very smooth. If h is relatively small, the kernel estimate is determined by the small number of points that are close to x, and the estimate of the density is more sensitive locally to the data (more "spiky" in appearance). Estimating a good value of h in practice can be somewhat problematic. There is no single objective methodology for finding the bandwidth h that has wide acceptance. Techniques based on cross-validation can be useful but are typically computationally complex and not always reliable. Simple "eyeballing" of the resulting density along specific dimensions is always recommended to check whether or not the chosen values for h appear reasonable.

Under appropriate assumptions these kernel models are flexible enough to approximate *any* smooth density function, if h is chosen appropriately, which

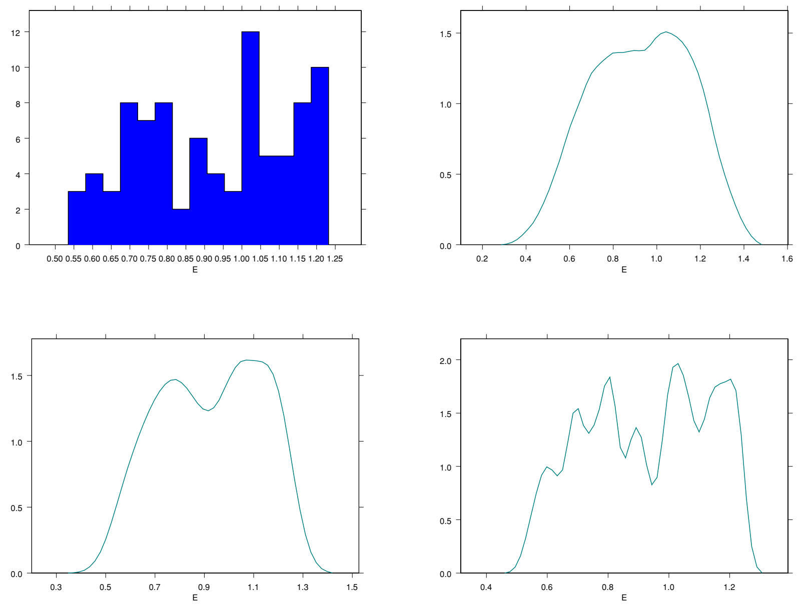


Figure 9.3 Density estimates for the variable ethanol (E) using a histogram (top left) and Gaussian kernel estimates with three different bandwidths: $h = 0.5$ (top right), $h = 0.25$ (lower left), and $h = 0.1$ (lower right).

adds to their appeal. However, this approximation result holds in the limit as we get an infinite number of data points, making it somewhat less relevant for the finite data sets we see in practice. Nonetheless, kernel models can be very valuable for low-dimensional problems as a way to determine structure in the data (such as local peaks or gaps) in a manner that might not otherwise be visible.

Example 9.1 Figure 9.3 shows an example of different density estimates for measurements of ethanol (E) from a data set involving air pollution measurements at different geographic locations. The histogram (top left) is quite "rough" and noisy, at least for this particular choice of bin widths and bin locations. The Normal kernel with bandwidth $h = 0.5$ is probably too smooth (top right). Conversely, the estimate based on a bandwidth of $h = 0.1$ (lower right) is probably too noisy, and introduces modes in the density that are likely to be spurious. The $h = 0.25$ estimate (lower left) is quite plausible and appears to have a better trade-off between over-and undersmoothing than the other estimates; it would suggest that the ethanol measurements have a definite bimodal characteristic. While visual inspection can be useful technique for determining bandwidths interactively, once again, it is largely limited to one-dimensional or two-dimensional problems.

Density estimation with kernel models becomes much more difficult as p increases. To begin with, we now need to define a p-dimensional kernel function. A popular choice is to define the p-dimensional kernel as a product of one-dimensional kernels, each with its own bandwidth, which keeps the number of parameters (the bandwidths $h_1, \ldots, h_p$ for each dimension) linear in the number of dimensions. A less obvious problem is the fact that in high dimensions it is natural for points to be farther away from each other than we might expect intuitively (the "curse of dimensionality" again). In fact, if we want to keep our approximation error constant as p increases, the number of data points we need grows exponentially with p. (Recall the example in chapter 6 where we would need 842,000 data points to get a reliable estimate of the density value at the mean of a 10-dimensional Normal distribution.) This is rather unfortunate and means in practice that kernel models are really practical only for relatively low-dimensional problems.

Kernel methods are often complex to implement for large data sets. Unless the kernel function $K(t)$ has compact support (that is, unless it is zero outside some finite range on t) then calculating the kernel estimate $f(x)$ at some point x potentially involves summing over contributions from all n data points in the database. In practice of course since most of these contributions will be negligible (that is, will be in the tails of the kernel) there are various ways to speed up this calculation. Nonetheless, this "memory-based" representation can be a relatively complex one to store and compute with (it can be $O(n)$ to compute the density at just one query data point).

9.2.7 Joint Distributions for Categorical Data

In chapter 6 we discussed the problem of constructing joint distributions for multivariate categorical data. Say we have p variables each taking m values. The joint distribution requires the specification of $O(m^p)$ different probabilities. This exponential growth is problematic for a number of reasons.

First there is the problem of how to estimate such a large number of probabilities. As an example, let $\{p_1, \ldots, p_{m^p}\}$ represent a list of all the joint probability terms in the unknown distribution we are trying to estimate from a data set with n p-dimensional observations. Hence, we can think of m^p different "cells," $\{c_1, \ldots, c_{m^p}\}$ each containing n_i observations, $1 \leq i \leq m^p$. The expected number of data points in cell$i$, given a random sample from $p(\mathbf{x})$ of size n, can be written as $E_{p(\mathbf{x})}[n_i] = np_i$. Assuming (for example) that $p(\mathbf{x})$ is approximately uniform (that is, $p_i \approx 1/m^p$) we get that

$$E_{p(\mathbf{x})}[n_i] \approx \frac{n}{m^p}. \tag{9.10}$$

Thus, for example, if $n < 0.5m^p$, the expected number of data points falling in any given cell is closer to 0 than to 1. Furthermore, if we use straightforward frequency counts (the maximum likelihood estimate—see chapter 4) as our method for estimating probabilities, we will estimate $\hat{p}_i = 0$ for each empty cell, whether or not $p_i = 0$ in truth. Note that if $p(\mathbf{x})$ is nonuniform the problem is actually worse since there will be more cells with smaller p_i (that is, less chance of any data falling in them). The fundamental problem here is the m^p exponential growth in the number of cells. With $p = 20$ binary variables ($m = 2$) we get $m^p \approx 10^6$. By doubling the number of variables to $p = 40$ we now get $m^p \approx 10^{12}$. Say that we had n data points for the case of $p = 20$ and that we wanted to add some new variables to the analysis while still keeping the expected number of data points per cell to be constant (that is, the same as it was with n data points). If we added extra 20 variables to the problem we would need to increase the data set from n to $n' = 10^6 n$, an increase by a factor of a million.

A second practical problem is that even if we can reliably estimate a full joint distribution from data, it is exponential in both space and time to work with directly. A full joint distribution will have a $O(m^p)$ memory requirement; for example, $O(10^{12})$ real-valued probabilities would need to be stored for a full distribution on 40 binary variables. Furthermore, many computations using this distribution will also scale exponentially. Let the variables be $\{X_1, \ldots, X_p\}$, each taking m values. If we wanted to determine the marginal distribution on any single variable X_j (say), we could calculate it as

$$p(x_j) = \sum_{X_1,\ldots,X_{j-1},X_{j+1},\ldots,X_p} p(X_1, \ldots, X_{j-1}, x_j, X_{j+1} \ldots, X_p), \tag{9.11}$$

that is, by summing over all the other variables in the distribution. The sum on the right involves $O(m^{p-1})$ summations—for example, $O(10^{39})$ summations when $p = 40$ and $m = 2$. Clearly this sort of exercise is intractable except for relatively small values of m and p.

The practical consequence is that we can only reliably estimate and work with *full* joint distributions for relatively low-dimensional problems. Although our examples were for categorical data, essentially the same problems also arise of course for ordered or real-valued data.

As we have seen in chapter 6, one of the key ideas for addressing this curse of dimensionality is to impose *structure* on the underlying distribution $p(\mathbf{x})$—for example, by *assuming* independence:

$$p(\mathbf{x}) = p(x_1, \ldots, x_p) = \prod_{j=1}^{p} p_j(x_j). \tag{9.12}$$

Instead of requiring $O(m^p)$ separate probabilities here we now only need p "marginal" distributions $p_1(x_1), \ldots, p_p(x_p)$, each of which can be specified by m numbers, for a total of mp probabilities. Of course, as discussed earlier, the independence assumption is just that, an *assumption,* and typically it is far too strong an assumption for most real-world data mining problems.

As described earlier in chapter 6, a somewhat weaker assumption is to presume that there exists a hidden ("latent") variable C, taking K values, and that the measurements $\mathbf{x}$ are *conditionally independent* given C. This is equivalent to the mixture distributions discussed earlier, with an additional assumption of conditional independence within each component; that is,

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k p_k(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \left(\prod_{j=1}^{p} p_{k,j}(x_j) \right). \tag{9.13}$$

This model requires mp probabilities per component, times K components, in addition to the K component weights $\pi_1, \ldots, \pi_K$. Thus, it scales linearly in K, m, and p, rather than exponentially. The EM algorithm can again be used to estimate the parameters for each component $p_k(\mathbf{x})$ (and the weights π_k), where the conditional independence assumption is enforced during estimation. One way to think about this "mixture of independence models" is that we are trying to find K different groups in the data such that for each group the variables are at least approximately conditionally independent. In fact, given a fixed K value, EM will try to find K component distributions (each of conditional independence form) that maximize the overall likelihood of the data. This model can be quite useful for modeling large sparse transactional data sets or sets of text documents represented as binary vectors. Finding a suitable value of K depends on our goal: from a descriptive viewpoint we can vary K in accordance with how complex we wish our fitted model to be. Note also that this form of model is equivalent to the first-order "naive"

Bayes model discussed in chapter 6 (and again in chapter 10 in the context of classification), whereas here the class variable C is unobserved and must be learned from the data. We will see later in this chapter that this also forms a useful basis for clustering the data, where we interpret each component $p_k(\mathbf{x})$ as a cluster.

A somewhat different way to structure a probability distribution parsimoniously is to model conditional independence in a general manner. We have described one such general framework (known as *belief networks*, or equivalently, *acyclic directed graphical models*) back in chapter 6. Recall that the basic equation for such models can be written as

$$p(\mathbf{x}) = \prod_{j=1}^{p} p_j(x_j | pa(x_j)), \tag{9.14}$$

which is a *factorization* of the overall joint distribution function into a product of conditional distributions. In fact, such a factorization can always be defined by the chain rule, but this model gains its power when the dependencies can be assumed to be relatively sparse. Recall that the graphical formalism associates each variable X_j with a single node in a graph. A directed edge from X_i to X_j indicates that X_j depends directly on X_i. $pa(x_j)$ indicates values taken from the *parent set* $pa(X_j)$ of variables for variable X_j. The connectivity structure of a graph implies a set of conditional independence relationships for $p(\mathbf{x})$. These independence relationships can be summarized by the fact that, given the values of the parents of X_j, $pa(X_j)$, a node X_j is independent of all other variables in the graph that are non-descendants of X_j.

If the sizes of the parent sets in the graph are relatively small compared to p, then we will have a much simpler representation for the joint distribution (compared to the full model). In this context, the independence model corresponds to a graph with no edges at all, and the complete graph corresponds to the full joint distribution with no independence structure being assumed. Another well-known graph structure is the Markov chain model, in which the variables are ordered in some manner (for example, temporally) and each variable X_j depends only on X_{j-1}. Here each variable is linked to just two others, so that the overall graph is a single line of connected nodes (see figure 6.7 in chapter 6).

A primary attraction of the graphical formalism is that it provides a systematic and mathematically precise language for describing and communicating the structure of independence relationships in probability distributions.

Perhaps more importantly it also provides a systematic framework for *computational methods* in handling probability calculations with the associated joint distribution. For example, if the underlying graph is singly-connected (that is, when directionality of the edges is ignored the graph has no loops), one can show that the time to compute any marginal or conditional probability of interest is upper bounded by pm^{d+1}, where p is the number of variables, m is the number of values for each variable (assumed the same for all variables for simplicity), and d is the number of variables in the largest parent set in the graph. For example, for a Markov chain model we have $d = 1$ leading to the well-known $O(pm^2)$ complexity for such models. For graphs that have loops, there is an equivalent complexity bound of $pm^{d'+1}$, where d' is the size of the largest parent set in an equivalent singly connected graph (obtained from the original graph in a systematic manner).

From a data mining perspective there are two aspects to learning graphical models from data: learning the parameters given a fixed graphical structure, and the more difficult problem of learning parameters and structure together. Note that in the categorical case the parameters of the model are simply the conditional probability tables for each variable, $p(x_j|pa(X_j)), 1 \leq j \leq p$.

Given a fixed structure, there is no need to perform structure-search, and the simple maximum likelihood or MAP score functions work fine. If there are no hidden variables, the problem of learning reduces to estimating the conditional probability tables for each variable X_j given its parents $pa(X_j)$: in either the maximum likelihood or MAP case this reduces to simple counting (see chapter 4). With hidden variables, and assuming that the connectivity of these hidden variables in the graph is known, the EM algorithm (chapter 8) is again directly applicable under fairly broad conditions. The estimation of the conditional probability tables is now iterative (rather than closed-form as in the nonhidden case), and as usual care must be taken with initial conditions and detection of convergence. The mixture models discussed earlier can be viewed as graphical models with a single hidden variable. Hidden Markov models (as used in speech) can be viewed as graphical models with a discrete hidden time-dependent variable that is assumed to be Markov.

It is worth emphasizing that if we have strong prior belief that a particular graphical model structure is appropriate for our data mining problem, then it is usually worth taking advantage of this knowledge (assuming it is reliable) either as a fixed model or as a starting point for the structure learning methods described below.

Learning the structure of a directed graphical model from data has been a topic of research interest recently, and there now exist numerous algorithms for this purpose. Consider, first, the problem of learning structure with no hidden variables. The score function is typically some form of penalized likelihood: the BIC score function (see section 9.2.2), for example, is fairly widely used because it is easy to compute. Given a score function, the problem reduces to searching in graph space for the graph structure (with estimated parameters) that produces the maximum score. The general problem of finding the maximum score has been shown to be NP-hard (as seems to be the case with most nontrivial structure-finding problems in data mining). Thus, iterative local search methods are used: starting with some "prior" structure such as the empty graph and then adding and deleting edges until no further local improvement in the score function is possible. One useful feature from a computational viewpoint is that because the distribution can be expressed in *factored form* (equation 9.14), the likelihood and penalty terms can also be factored into expressions that are local in terms of the graph structure—for example, terms that only involve X_j and its parents. Thus, we can calculate the effect of local changes to the model (such as adding or deleting an edge) with local computations (since the impact of the change affects only one factor in the score function).

Learning structure with hidden variables is still considered to be something of a research problem. Clearly it is more difficult than learning structure with no hidden variables (which is itself NP-hard). The EM algorithm is again applicable, but the search problem is typically quite complex since there are so many different ways that one can introduce hidden variables into a multivariate model.

The family of *log-linear models* is a further generalization of acyclic directed graphical models, which characterize dependence relations in a more general form. Discussion of this class of models is beyond the scope of this text (references are provided in the section on further reading). Markov random fields are another class of graphical models, where an *undirected* graph is used to represent dependence, e.g., to represent correlational effects between pixels in an image. These random field models have seen wide application in image analysis and spatial statistics, where they are used to define a joint distribution over measurements on a grid or image.

9.3 Background on Cluster Analysis

We now move beyond probability density and distribution models to focus on the related descriptive data mining task of *cluster analysis*—that is, decomposing or partitioning a (usually multivariate) data set into groups so that the points in one group are similar to each other and are as different as possible from the points in other groups. Although the same techniques may often be applied, we should distinguish between two different objectives. In one, which we might call *segmentation* or *dissection,* the aim is simply to partition the data in a way that is convenient. "Convenient" here might refer to administrative convenience, practical convenience, or any other kind. For example, a manufacturer of shirts might want to choose just a few sizes and shapes so as to maximize coverage of the male population. He or she will have to choose those sizes in terms of collar size, chest size, arm length, and so on, so that no man has a shape too different from that of a well-fitting shirt. To do this, he or she will partition the population of men into a few groups in terms of the variables collar, chest, and arm length. Shirts of one size will then be made for each group.

In contrast to this, we might want to see whether a sample of data is composed of natural subclasses. For example, whiskies can be characterized in terms of color, nose, body, palate, and finish, and we might want to see whether they fall into distinct classes in terms of these variables. Here we are not partitioning the data for practical convenience, but rather are hoping to discover something about the nature of the sample or the population from which it arose—to discover whether the overall population is, in fact, heterogeneous.

Technically, this second exercise is what *cluster analysis* seeks to do—to see whether the data fall into distinct groups, with members within each group being similar to other members in that group but different from members of other groups. However, the term "cluster analysis" is often used in general to describe both segmentation and cluster analysis problems (and we shall also be a little lax in this regard). In each case the aim is to split the data into classes, so perhaps this is not too serious a misuse. It is resolved, as we shall see below, by the fact that there is a huge number of different algorithms for partitioning data in this way. The important thing is to match our method with our objective. This way, mistakes will not arise, whatever we call the activity.

Example 9.2 Owners of credit cards can be split into subgroups according to how they use their card—what kind of purchases they make, how much money they spend, how often they use the card, where they use the card, and so on. It can be very useful for marketing purposes to identify the group to which a card owner belongs, since he or she can then be targeted with promotional material that might be of interest (this clearly benefits the owner of the card, as well as the card company). Market segmentation in general is, in fact, a heavy user of the kinds of techniques discussed in this section. The segmentation may be in terms of lifestyle, past purchasing behavior, demographic characteristics, or other features.

A chain store might want to study whether outlets that are similar, in terms of social neighborhood, size, staff numbers, vicinity to other shops, and so on, have similar turnovers and yield similar profits. A starting point here would be to partition the outlets, in terms of these variables, and then to examine the distributions of turnover within each group.

Cluster analysis has been heavily used in some areas of medicine, such as psychiatry, to try to identify whether there are different subtypes of diseases lumped together under a single diagnosis.

Cluster analysis methods are used in biology to see whether superficially identical plants or creatures in fact belong to different species. Likewise, geographical locations can be split into subgroups on the basis of the species of plants or animals that live there.

As an example of where the difference between dissection and clustering analysis might matter, consider partitioning the houses in a town. If we are organizing a delivery service, we might want to split them in terms of their geographical location. We would want to dissect the population of houses so that those within each group are as close as possible to each other. Delivery vans could then be packed with packages to go to just one group. On the other hand, a company marketing home improvement products might want to split the houses into naturally occurring groups of similar houses. One group might consist of small starter homes, another of three-and four-bedroom family homes, and another (presumably smaller) of executive mansions.

It will be obvious from this that such methods (cluster and dissection techniques) hinge on the notion of *distance*. In order to decide whether a set of points can be split into subgroups, with members of a group being closer to other members of their group than to members of other groups, we need to say what we mean by "closer to." The notion of "distance," and different measures of it, has already been discussed in chapter 2. Any of the measures described there, or indeed any other distance measure, can be used as the basis for a cluster or dissection analysis. As far as these techniques are concerned, the concept of distance is more fundamental than the coordinates of the points. In principle, to carry out a cluster analysis all we need to know is the set of interpoint distances, and not the values on any variables. However, some methods make use of "central points" of clusters, and so require that the raw coordinates be available.

Cluster analysis has been the focus of a huge amount of research effort, going back for several decades, so that the literature is now vast. It is also scattered. Considerable portions of it exist in the statistical and machine learning literatures, but other many other publications on cluster analysis may be found elsewhere. One of the problems is that new methods are constantly being developed, sometimes without an awareness of what has already been developed. More seriously, a proper understanding of their properties and the way they behave with different kinds of data is available for very few of the methods. One of the reasons for this is that it is difficult to tell whether a cluster analysis has been successful. Contrast this with predictive modeling, in which we can take a test data set and see how accurately the value of the target variable is predicted in this set. For a clustering problem, unfortunately, there is no direct notion of *generalization* to a test data set, although, as we will see in our discussion of probabilistic clustering (later in this chapter), it is possible in some situations to pose the question of whether or not the cluster structure discovered in the training data is genuinely present in the underlying population. Generally speaking, however, the validity of a clustering is often in the eye of the beholder; for example, if a cluster produces an interesting scientific insight, we can judge it to be useful. Quantifying this precisely is difficult, if not impossible, since the interpretation of how interesting a clustering is will inevitably be application-dependent and subjective to some degree.

As we shall see in the next few sections, different methods of cluster analysis are effective at detecting different *kinds* of clusters, and we should consider this when we choose a particular algorithm. That is, we should consider what we mean or intend to mean by a "cluster." In effect, different clustering algorithms will be biased toward finding different types of cluster structures (or "shapes") in the data, and it is not always easy to ascertain precisely what this bias is from the description of the clustering algorithm.

To illustrate, we might take a "cluster" as being a collection of points such that the maximum distance between all pairs of points in the cluster is as small as possible. Then each point will be similar to each other point in the cluster. An algorithm will be chosen that seeks to partition the data so as to minimize this maximum interpoint distance (more on this below). We would clearly expect such a method to produce compact, roughly spherical, clusters. On the other hand, we might take a "cluster" as being a collection of points such that each point is as close as possible to some other member of the cluster—although not necessarily to all other members. Clusters discovered by this approach need not be compact or roughly spherical, but could have long (and not necessarily straight) sausage shapes. The first approach would simply fail to pick up such clusters. The first approach would be appropriate in a segmentation situation, while the second would be appropriate if the objects within each hypothesized group were measured at different stages of some evolutionary process. For example, in a cluster analysis of people suffering from some illness, to see whether there were different subtypes, we might want to allow for the possibility that the patients had been measured at different stages of the disease, so that they had different symptom patterns even though they belonged to the same subtype.

The important lesson to be learned from this is that we must match the method to the objectives. In particular, we must adopt a cluster analytic tool that is effective at detecting clusters that conform to the definition of what is meant by "cluster" in the problem at hand. It is perhaps worth adding that we should not be too rigid about it. Data mining, after all, is about discovering the *unexpected,* so we must not be too determined in imposing our preconceptions on the analysis. Perhaps a search for a different kind of cluster structure will throw up things we have not previously thought of.

Broadly speaking, we can identify three different general types of cluster analysis algorithms: those based on an attempt to find the optimal partition into a specified number of clusters, those based on a hierarchical attempt to discover cluster structure, and those based on a probabilistic model for the underlying clusters. We discuss each of these in turn in the next three sections.

9.4 Partition-Based Clustering Algorithms

In chapter 5 we described how data mining algorithms can often be conveniently thought of in five parts: the *task,* the *model,* the *score function,* the *search* method, and the *data management* technique. In partition-based clustering the *task* is to partition a data set into k disjoint sets of points such that the points within each set are as homogeneous as possible, that is, given the set of n data points $D = \{\mathbf{x}(1), \ldots, \mathbf{x}(n)\}$, our task is to find K clusters $\mathcal{C} = \{C_1, \ldots, C_K\}$ such that each data point $\mathbf{x}(i)$ is assigned to a unique cluster C_k.

Homogeneity is captured by an appropriate *score function* (as discussed below), such as minimizing the distance between each point and the centroid of the cluster to which it is assigned. Partition-based clustering typically places more emphasis on the score function than on any formal notion of a model. Often the *centroid* or *average* of the points belonging to a cluster is considered to be a representative point for that cluster, and there is no explicit statement of what sort of shape of cluster is being sought. For cluster representations based on the notion of a single "center" for each cluster, however, the boundaries between clusters will be implicitly defined. For example, if a point $\mathbf{x}$ is assigned to a cluster according to which cluster center is closest in a Euclidean-distance sense, then we will get linear boundaries between the clusters in $\mathbf{x}$ space.

We will see that maximizing (or minimizing) the score function is typically a computationally intractable *search* problem, and thus, iterative improvement heuristic search methods, such as those described in chapter 8, are often used to optimize the score function given a data set.

9.4.1 Score Functions for Partition-Based Clustering

A large number of different score functions can be used to measure the quality of clustering and a wide range of algorithms has been developed to search for an optimal (or at least a good) partition.

In order to define the clustering score function we need to have a notion of distance between input points. Denote by $d(\mathbf{x}, \mathbf{y})$ the distance between points $\mathbf{x}, \mathbf{y} \in D$, and assume for simplicity that the function d defines a metric on D. Most score functions for clustering stress two aspects: clusters should be compact, and clusters should be as far from each other as possible. A straightforward formulation of these intuitive notions is to look at *within cluster variation* $wc(\mathcal{C})$ and *between cluster variation* $bc(\mathcal{C})$ of a clustering $\mathcal{C}$. The within cluster variation measures how compact or tight the clusters are, while the between cluster variation looks at the distances between different clusters.

Suppose that we have selected *cluster centers* $\mathbf{r}_k$ from each cluster. This can be a designated representative data point $\mathbf{x}(i) \in C_k$ that is defined to be "central" in some manner. If the input points belong to a space where taking means makes sense, we can use the centroid of the points in the cluster C_k as the cluster center, where $\mathbf{r}_k$ will then be defined as

$$\mathbf{r}_k = \frac{1}{n_k} \sum_{\mathbf{x} \in C_k} \mathbf{x}, \qquad (9.15)$$

with n_k the number of points in the kth cluster. A simple measure of *within-cluster variation* is to look at the sum of squares of distances from each point to the center of the cluster it belongs to:

$$wc(\mathcal{C}) = \sum_{k=1}^{K} wc(C_k) = \sum_{k=1}^{K} \sum_{\mathbf{x}(i) \in C_k} d\left(\mathbf{x}, \mathbf{r}_k\right)^2 . \tag{9.16}$$

For the case in which $d(\mathbf{x}, \mathbf{r}_k)$ is defined as Euclidean distance, $wc(\mathcal{C})$ is referred to as the *within-cluster sum-of-squares*.

Between-cluster variation can be measured by the distance between cluster centers:

$$bc(\mathcal{C}) = \sum_{1 \le j < k \le K} d\left(\mathbf{r}_j, \mathbf{r}_k\right)^2 . \tag{9.17}$$

The overall quality (or score function) of a clustering $\mathcal{C}$ can then be defined as a monotone combination of the factors $wc(\mathcal{C})$ and $bc(\mathcal{C})$, such as the ratio $bc(\mathcal{C})/wc(\mathcal{C})$.

The within cluster measure above is in a sense global: for the cluster C_k to make a small contribution to it, all points of C_k have to be relatively close to the cluster center. Thus the use of this measure of cluster tightness leads to spherical clusters. The well-known K-means algorithm, discussed in the next section, uses the means within each group as cluster centers and Euclidean distance for d to search for the clustering $\mathcal{C}$ that minimizes the within cluster variation of equation 9.16, for measurements $\mathbf{x}$ in a Euclidean space $\mathcal{R}^p$.

If we are given a candidate clustering, how difficult is it to evaluate $wc(\mathcal{C})$ and $bc(\mathcal{C})$? Computing $wc(\mathcal{C})$ takes $O(\sum_i |C_i|) = O(n)$ operations, while $bc(\mathcal{C})$ can be computed in $O(k^2)$ operations. Thus computing a score function for a single clustering requires (at least in principle) a pass through the whole data.

A different notion of within cluster variation is to consider for each point in the cluster the distance to the nearest point in the same cluster, and take the maximum of these distances:

$$wc(C_k) = \max_i \min_{\mathbf{y}(j) \in C_k} \{d\left(\mathbf{x}(i), \mathbf{y}(j)\right) \mid x(i) \in C_k, x \neq y\}. \tag{9.18}$$

This *minimum distance* or *single-link* criterion for cluster distance leads to elongated clusters. We will return to this score function in the context of hierarchical agglomerative clustering algorithms in section 9.5.

We can use the notion of covariance to develop more general score functions for clusterings $\mathcal{C}$ in a Euclidean space. For points within a particular cluster C_k, we can define a $p \times p$ matrix

$$\mathbf{W}_k = \sum_{\mathbf{x} \in C_k} (\mathbf{x} - \mathbf{r}_k)(\mathbf{x} - \mathbf{r}_k)^T \tag{9.19}$$

that is an (unnormalized) *covariance matrix* for the points in cluster C_k. The within-cluster sum-of-squares for a particular cluster is then the trace (sum of diagonal elements) of this matrix, $tr(\mathbf{W}_k)$, and thus the total within-cluster sum-of-squares of equation 9.16 can be expressed as

$$wc(\mathcal{C}) = \sum_k tr(\mathbf{W}_k). \tag{9.20}$$

In this context, letting $\mathbf{W} = \sum_k \mathbf{W}_k$, we can see that a score function that tries to make $\mathbf{W}$ "smaller" (for example, minimize the trace or the determinant of $\mathbf{W}$) will tend to encourage a more compact clustering of the data.

We can define a matrix $\mathbf{B}$ that summarizes the squared differences between the cluster centers as

$$\mathbf{B} = \sum_{k=1}^{c} n_k (\mathbf{r}_k - \hat{\mu})(\mathbf{r}_k - \hat{\mu})^T, \tag{9.21}$$

where $\hat{\mu}$ is the estimated global mean of all data points in D. This is a $p \times p$ matrix that characterizes the covariance of the cluster means (weighted by n_k) with respect to each other. For example, $tr(\mathbf{B})$ is the weighted sum of squared distances of the cluster means relative to the estimated global mean of the data. Thus, having a score function that emphasizes a "larger" $\mathbf{B}$ will tend to encourage the cluster means to be more separated.

We stress again the important, but often overlooked, point that the nature of the score function has a very important influence on what types of clusters will be found in the data. Different score functions (for example, different combinations of $\mathbf{W}$ and $\mathbf{B}$) can express significantly different preferences in terms of cluster structure.

Traditional score functions based on $\mathbf{W}$ and $\mathbf{B}$ are $tr(\mathbf{W})$, the trace of $\mathbf{W}$, the determinant $\mid \mathbf{W} \mid$ of $\mathbf{W}$, and $tr(\mathbf{B}\mathbf{W}^{-1})$. A disadvantage of $tr(\mathbf{W})$ is that it depends on the scaling adopted for the separate variables. Alter the units of one of them and a different cluster structure may result. Of course, this can be overcome by standardizing the variables prior to analysis, but this is often just as arbitrary as any other choice. The $tr(\mathbf{W})$ criterion tends to yield compact spherical clusters, and it also has a tendency to produce roughly equal groups. Both of these properties may make this score function useful in a segmentation context, but they are less attractive for discovering natural clusters (for example, in astronomy the discovery of a distinct very small cluster may represent a major advance).

The $\mid \mathbf{W} \mid$ score function does not have the same scale dependence as $tr(\mathbf{W})$, so it also detects elliptic structures as clusters, but it also favors equal-sized clusters. Adjustments that take cluster size into account have been suggested (for example, dividing by $\prod n_k^{2n_k}$), so that the equal-sized cluster tendency is counteracted, but it might be better to go for a different criterion altogether than adjust an imperfect one. Note also that the original score function, $\mid \mathbf{W} \mid$, has optimality properties if the data are thought to arise from a mixture of multivariate normal distributions, and this is sacrificed by the modification. (Of course, if our data are thought to be generated in that way, we might contemplate fitting a formal mixture model, as outlined in section 9.2.4.)

The $tr(\mathbf{BW}^{-1})$ score function also has a tendency to yield equal-sized clusters, and this time of roughly equal shape. Note that since this score function is equivalent to summing the eigenvalues of $\mathbf{BW}^{-1}$ it will place most emphasis on the largest eigenvalue and hence will tend to yield collinear clusters.

The property that the clusters obtained from using these score functions tend to have similar shape is not attractive in all situations (indeed, it is probably rarely attractive). Score functions based on other ways of combining the separate within-cluster matrices $\mathbf{W}_k$ can relax this—for example, $\prod \mid \mathbf{W}_k \mid^{n_k}$ and $\prod \mid \mathbf{W}_k \mid^{1/p}$, where p is the number of variables. Even these score functions, however, have a tendency to favor similarly-sized clusters. (A modification to the $\prod \mid \mathbf{W}_k \mid^{n_k}$ score functions, analogous to that of the $\mid \mathbf{W} \mid$ score function, that can help to overcome this property, is to divide each $\mid \mathbf{W}_k \mid$ by $\prod n_k^{2n_k}$. This is equivalent to letting the distance vary between different clusters.)

A variant of these methods uses the sum of squared distances not from the cluster means, but from particular members of the cluster. The search (see below) then includes a search over cluster members to find the one that minimizes the score function. In general, of course, measures other than the sum of squared distances from the cluster "center" can be used. In particular, the influence of the outlying points of a cluster can be reduced by replacing the sum of squared distances by robust estimates of distance. The L_1 norm has also been proposed as a measure of distance. Typically this will be used with the vector of medians as the cluster "center."

Methods based on minimizing a within cluster matrix of sums of squares can be regarded as minimizing deviations from the centroids of the groups. A technique known as *maximal predictive classification* (developed for use with binary variables in taxonomy but more widely applicable) can also be regarded as minimizing deviations from group "centers," though with a different definition of centers. Suppose that each component of the measurement vector is binary—that is, each object has given rise to a binary vector—and suppose we have a proposed grouping into clusters. For each group we can define a binary vector that consists of the most common value, within the group, of each variable. This vector of modes (instead of means) will serve as the "center" of the group. Distance of a group member from this center is then measured in terms of the number of variables that have values that differ from those in this central vector. The score function optimized is then the total number of differences between the objects and the centers of the groups they belong to. The "best" grouping is the one that minimizes the overall number of such differences.

Hierarchical methods of cluster analysis, described in the next section, do not construct a single partition of the data, but rather construct a hierarchy of (typically) nested clusters. We can then decide where to cut the hierarchy so as to partition the data in such a way as to obtain the most convincing partition. For partition-based methods, however, it is necessary to decide at the start how many clusters we want. Of course, we can rerun the analysis several times, with different numbers of clusters, but this still requires us to be able to choose between competing numbers. There is no "best" solution to this problem. We can, of course, examine how the clustering score function changes as we increase the number of clusters, but this may not be comparable across different numbers; for example, perhaps the score shows apparent improvement as the number increases, regardless of whether there is really a better cluster structure (for example, the sum of within cluster squared distances is guaranteed to not increase with K). For a multivariate uniform distribution divided optimally into K clusters, the score function $K^2 \mid \mathbf{W} \mid$ asymptotically takes the same value for all K; results such as this can be used as the basis for comparing partitions with different K values.

It is apparent that cluster analysis is very much a data-driven tool, with relatively little formal model-building underlying it. However, some researchers have attempted to put it on a sounder model-based footing. For example, we can supplement the procedures by assuming that there is also a random process generating sparsely distributed points uniformly across the whole space, in addition to whatever mechanism generates clusters of points.

This makes the methods less susceptible to outliers. A further assumption is to model the distribution of the data parametrically within each cluster using specific distributional assumptions—we will return to this in our discussion of probabilistic model-based clustering in section 9.6.

9.4.2 Basic Algorithms for Partition-Based Clustering

We saw in the previous section that a large variety of score functions can be used to determine the quality of clustering. Now what about the algorithms to optimize those score functions? In principle, at least, the problem is straightforward. We simply search through the space of possible assignments $\mathcal{C}$ of points to clusters to find the one that minimizes the score (or maximizes it, depending on the chosen score function).

The nature of the search problem can be thought of as a form of combinatorial optimization, since we are searching for the allocation of n objects into K classes that maximizes (or minimizes) our chosen score function. The number of possible allocations (different clusterings of the data) is approximately K^n. For example, there are some $2^{100} \approx 10^{10}$ possible allocations of 100 objects into two classes. Thus, as we have seen with other data mining problems, direct exhaustive search methods are certainly not applicable unless we are dealing with tiny data sets. Nonetheless, for some clustering score functions, methods have been developed that permit exhaustive coverage of all possible clusterings without actually carrying out an exhaustive search. These include branch and bound methods, which eliminate potential clusterings on the grounds that they have poorer scores than alternatives already found, without actually evaluating the scores for the potential clusterings. Such methods, while extending the range over which exhaustive evaluation can be made, still break down for even moderately-sized data sets. For this reason, we do not examine them further here.

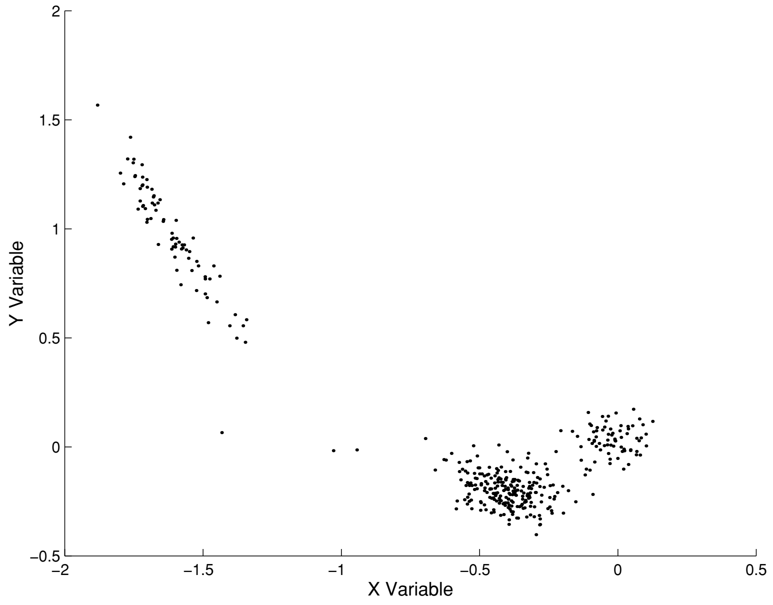
Unfortunately, neither do there exist closed-form solutions for any score function of interest; that is, there is usually no direct method for finding a specific clustering $\mathcal{C}$ that optimizes the score function. Thus, since closed form solutions do not exist and exhaustive search is infeasible, we must resort to some form of systematic *search* of the space of possible clusters (such search methods were discussed in chapter 8). It is important to emphasize that given a particular score function, the problem of clustering has been reduced to an optimization problem, and thus there are a large variety of choices available in the optimization literature that are potentially applicable.

Iterative-improvement algorithms based on local search are particularly popular for cluster analysis. The general idea is to start with a randomly chosen clustering of the points, then to reassign points so as to give the greatest increase (or decrease) in the score function, then to recalculate the updated cluster centers, to reassign points again, and so forth until there is no change in the score function or in the cluster memberships. This greedy approach has the virtue of being simple and guaranteeing at least a local maximum (minimum) of the score function. Of course it suffers the usual drawback of greedy search algorithms in that we do not know how good the clustering $\mathcal{C}$ that it converges to is relative to the best possible clustering of the data (the global optimum for the score function being used).

Here we describe one well-known example of this general approach, namely, the K-means algorithm (which has close connection to the EM algorithm discussed in chapter 8 and was mentioned in section 9.2.4). The number K of clusters is fixed before the algorithm is run (this is typical of many clustering algorithms). There are several variants of the K-means algorithm. The basic version begins by randomly picking K cluster centers, assigning each point to the cluster whose mean is closest in a Euclidean distance sense, then computing the mean vectors of the points assigned to each cluster, and using these as new centers in an iterative approach. As an algorithm, the method is as follows: assuming we have n data points $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, our task is to find K clusters $\{C_1 \ldots, C_K\}$:

```
for $k = 1, \ldots, K$ let $\mathbf{r}(k)$ be a randomly chosen point from $D$;
while changes in clusters $C_k$ happen do
    form clusters:
    for $k = 1, \ldots, K$ do
        $C_k = \{\mathbf{x} \in D \mid d(\mathbf{r}_k, \mathbf{x}) \leq d(\mathbf{r}_j, \mathbf{x})$ for all $j = 1, \ldots, K, j \neq k\}$;
    end;
    compute new cluster centers:
    for $k = 1, \ldots, K$ do
        $\mathbf{r}_k$ = the vector mean of the points in $C_k$
    end;
end;
```

Example 9.3 Electromechanical control systems for large 34m and 70m antennas are an important component in NASA's Deep Space Network for tracking and communicating with deep-space spacecraft. The motor-currents of the an-



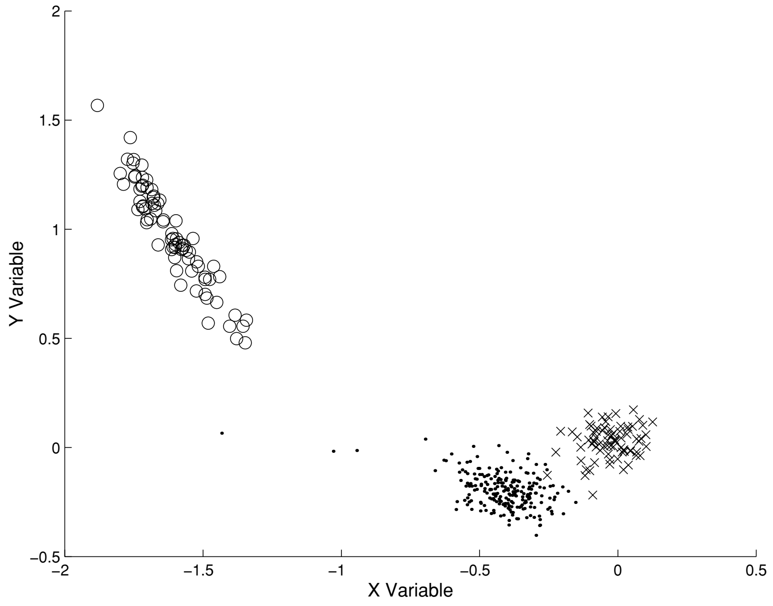


Figure 9.4 Antenna data. On top the data points are shown without class labels, and on the bottom different symbols are used for the three known classes (dots are normal, circles are tachometer noise, and x's are short circuit.)

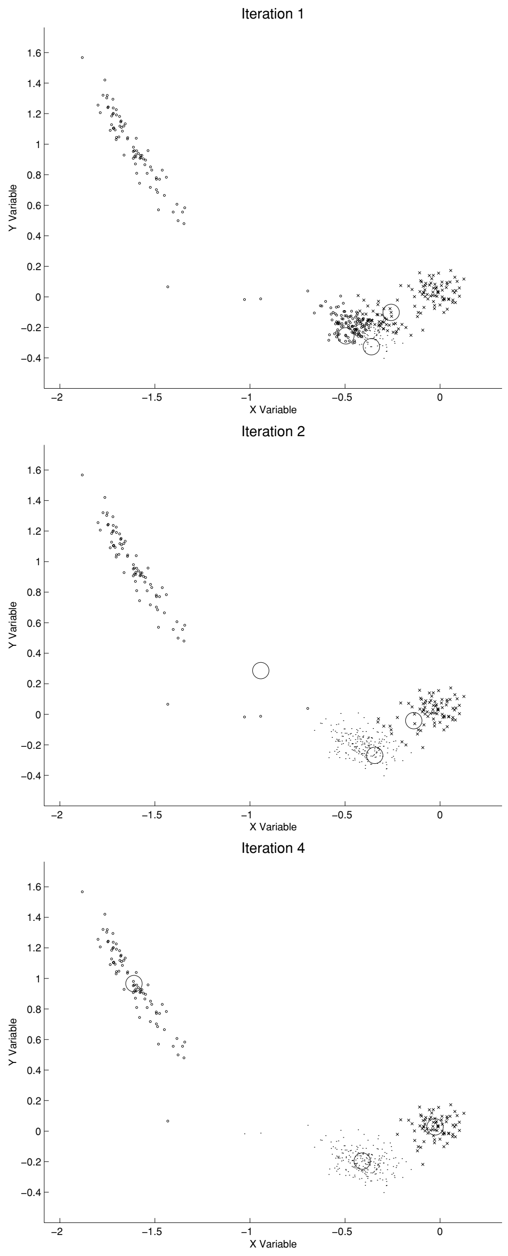


Figure 9.5 Example of running the K-means algorithm on the two-dimensional antenna data. The plots show the locations of the means of the clusters (large circles) at various iterations of the K-means algorithm, as well as the classification of the data points at each iteration according to the closest mean (dots, circles, and xs for each of the three clusters).

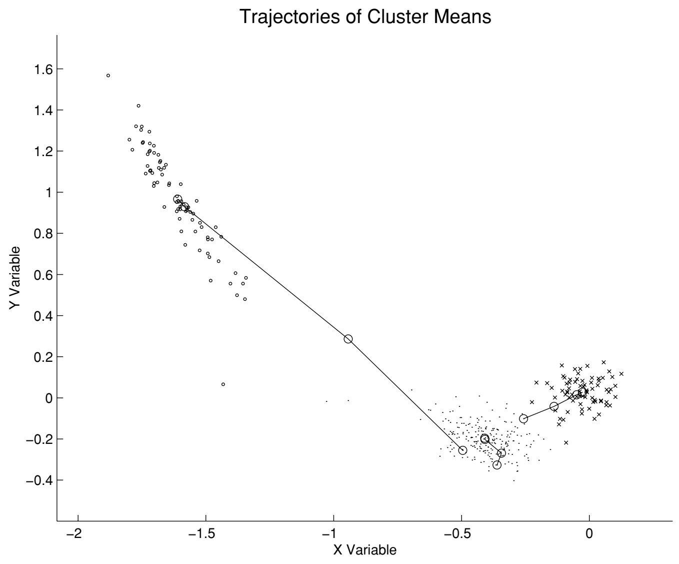


Figure 9.6 A summary of the trajectories of the three cluster means during the K-means iterations of figure 9.5.

tenna control systems are quite sensitive to subtle changes in operating behavior and can be used for online health monitoring and fault detection. Figure 9.4 shows sample data from a 34m Deep Space Network antenna. Each bivariate data point corresponds to a two-second window of motor-current measurements, that have been modeled by a simple autoregressive (linear) time-series model, and where the two dimensions correspond to the first two estimated coefficients of the autoregressive model for a particular window. The model is fit in real time to the data every two seconds, and changes in coefficients reflect changes in the spectral signature of the motor current measurements.

The data in the lower plot of figure 9.4 show which data points belong to which condition (three groups, one normal and two fault conditions). Figure 9.5 is an illustrative example of the results of applying the K-means algorithm to clustering this data, using $K = 3$, and having removed the class labels (that is, using the data in the upper plot of figure 9.4 as input to the K-means algorithm). All three initial starting points for the algorithm are located in the center (normal) cloud, but after only four iterations (figure 9.5) the algorithm quickly converges to a clustering (the trajectory of the cluster means are plotted in figure 9.6). The final clustering after the fourth iteration (lower plot of figure

9.5) produces three groups that very closely match the known grouping shown in figure 9.4. For this data the grouping is relatively obvious, of course, in that the various fault conditions can be seen to be separated from the normal cloud (particularly the tachometer noise condition on the left). Nonetheless it is reassuring to see that the K-means algorithm quickly and accurately converges to a clustering that is very close to the true groups.

The complexity of the K-means algorithm is $O(KnI)$, where I is the number of iterations. Namely, given the current cluster centers $\mathbf{r}_k$, we can in one pass through the data compute all the Kn distances $d(\mathbf{r}_k, \mathbf{x})$ and for each $\mathbf{x}$ select the minimal one; then computing the new cluster centers can also be done in time $O(n)$.

A variation of this algorithm is to examine each point in turn and update the cluster centers whenever a point is reassigned, repeatedly cycling through the points until the solution does not change. If the data set is very large, we can simply add in each data point, without the recycling. Further extensions (for example, the ISODATA algorithm) include splitting and/or merging clusters. Note that there are a large number of different partition-based clustering algorithms, many of which hinge around adding or removing one point at a time from a cluster. Efficient updating formula been developed in the context of evaluating the change incurred in a score function by moving one data point in or out of a cluster—in particular, for all of the score functions involving $\mathbf{W}$ discussed in the last section.

The search in the K-means algorithm is restricted to a small part of the space of possible partitions. It is possible that a good cluster solution will be missed due to the algorithm converging to a local rather than global minimum of the score function. One way to alleviate (if not solve) this problem is to carry out multiple searches from different randomly chosen starting points for the cluster centers. We can even take this further and adopt a simulated annealing strategy (as discussed in chapter 8) to try to avoid getting trapped in local minima of the score function.

Since cluster analysis is essentially a problem of searching over a huge space of potential solutions to find whatever optimizes a specified score function, it is no surprise that various kinds of mathematical programming methods have been applied to this problem. These include linear programming, dynamic programming, and linear and nonlinear integer programming.

Clustering methods are often applied on large data sets. If the number of observations is so large that standard algorithms are not tractable, we can try to compress the data set by replacing groups of objects by succinct representations. For example, if 100 observations are very close to each other in a metric space, we can replace them with a weighted observation located at the centroid of those observations and having an additional feature (the radius of the group of points that is represented). It is relatively straightforward to modify some of the clustering algorithms to operate on such "condensed" representations.

9.5 Hierarchical Clustering

Whereas partition-based methods of cluster analysis begin with a specified number of clusters and search through possible allocations of points to clusters to find an allocation that optimizes some clustering score function, hierarchical methods gradually merge points or divide superclusters. In fact, on this basis we can identify two distinct types of hierarchical methods: the *agglomerative* (which merge) and the *divisive* (which divide). The agglomerative are the more important and widely used of the two. Note that hierarchical methods can be viewed as a specific (and particularly straightforward) way to reduce the size of the search. They are analogous to stepwise methods used for model building in other parts of this book.

A notable feature of hierarchical clustering is that it is difficult to separate the model from the score function and the search method used to determine the best clustering. Because of this, in this section we will focus on clustering algorithms directly. We can consider the final hierarchy to be a model, as a hierarchical mapping of data points to clusters; however, the nature of this model (that is, the cluster "shape") is implicit in the algorithm rather than being explicitly represented. Similarly for the score function, there is no notion of an explicit global score function. Instead, various local scores are calculated for pairs of leaves in the tree (that is, pairs of clusters for a particular hierarchical clustering of the data) to determine which pair of clusters are the best candidates for agglomeration (merging) or dividing (splitting). Note that as with the global score functions used for partition-based clustering, different local score functions can lead to very different final clusterings of the data.

Hierarchical methods of cluster analysis permit a convenient graphical display, in which the entire sequence of merging (or splitting) of clusters is shown. Because of its tree-like nature, such a display is called a *dendrogram*. We illustrate in an example below.

Cluster analysis is particularly useful when there are more than two variables: if there are only two, then we can eyeball a scatterplot to look for

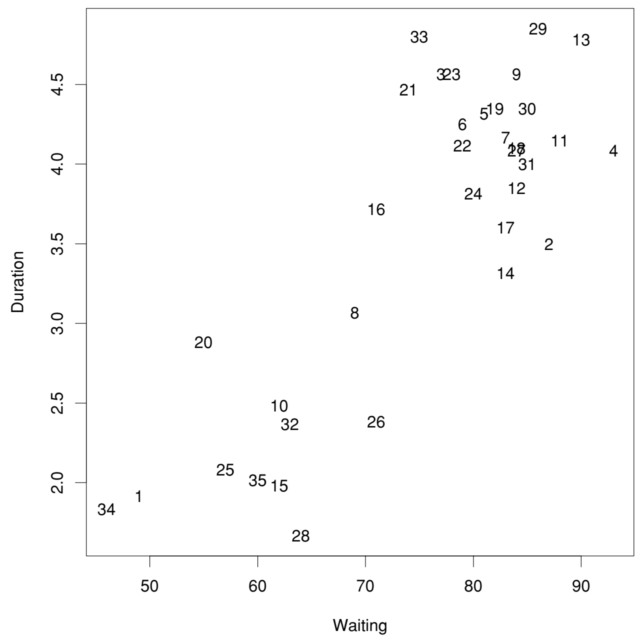


Figure 9.7 Duration of eruptions versus waiting time between eruptions (in minutes) for the Old Faithful geyser in Yellowstone Park.

structure. However, to illustrate the ideas on a data set where we can see what is going on, we will apply a hierarchical method to some two dimensional data. The data are extracted from a larger data set given in Azzalini and Bowman (1990). Figure 9.7 shows a scatterplot of the two-dimensional data. The vertical axis is the time between eruptions and the horizontal axis is the length of the following eruption, both measured in minutes. The points are given numbers in this plot merely so that we can relate them to the den-

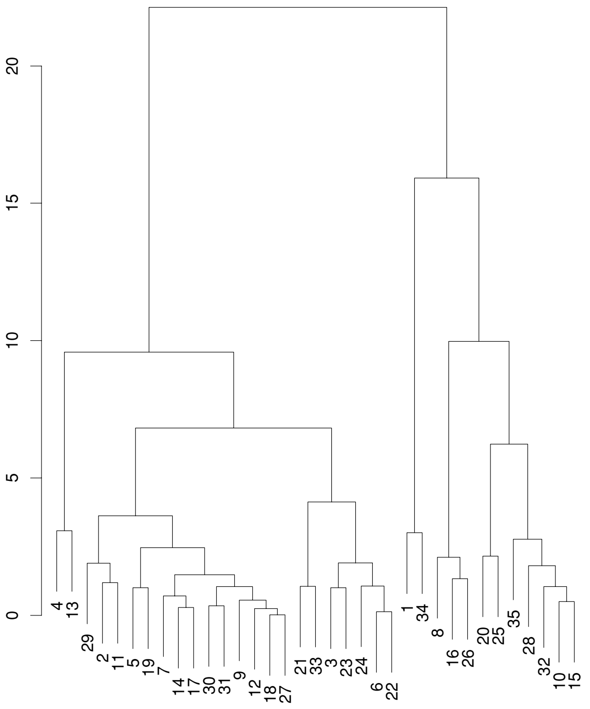


Figure 9.8 Dendrogram resulting from clustering of data in figure 9.7 using the criterion of merging clusters that leads to the smallest increase in the total sum of squared errors.

drogram in this exposition, and have no other substantive significance.

As an example, figure 9.8 shows the dendrogram that results from agglomerative merging the two clusters that leads to the smallest increase in within-cluster sum of squares. The height of the crossbars in the dendrogram (where branches merge) shows values of this score function. Thus, initially, the smallest increase is obtained by merging points 18 and 27, and from figure 9.7 we can see that these are indeed very close (in fact, the closest). Note that closeness from a visual perspective is distorted because of the fact that the x-scale is in fact compressed on the page relative to the y-scale. The next merger comes from merging points 6 and 22. After a few more mergers of individual pairs of neighboring points, point 12 is merged with the cluster consisting of the two points 18 and 27, this being the merger that leads to least increase in the clustering criterion. This procedure continues until the final merger, which is of two large clusters of points. This structure is evident from the dendrogram. (It need not always be like this. Sometimes the final merger is of a large cluster with one single outlying point—as we shall see below.) The hierarchical structure displayed in the dendrogram also makes it clear that we could terminate the process at other points. This would be equivalent to making a horizontal cut through the dendrogram at some other level, and would yield a different number of clusters.

9.5.1 Agglomerative Methods

Agglomerative methods are based on measures of distance between *clusters*. Essentially, given an initial clustering, they merge those two clusters that are nearest, to form a reduced number of clusters. This is repeated, each time merging the two closest clusters, until just one cluster, of all the data points, exists. Usually the starting point for the process is the initial clustering in which each cluster consists of a single data point, so that the procedure begins with the n points to be clustered.

Assume we are given n data points $D = \{\mathbf{x}(1), \dots, \mathbf{x}(n)\}$, and a function $\mathcal{D}(C_i, C_j)$ for measuring the distance between two clusters C_i and C_j. Then an agglomerative algorithm for clustering can be described as follows:

```
for $i = 1, \dots, n$ let $C_i = \{\mathbf{x}(i)\}$;
while there is more than one cluster left do
    let $C_i$ and $C_j$ be the clusters
        minimizing the distance $\mathcal{D}(C_k, C_h)$ between any two clusters;
    $C_i = C_i \cup C_j$;
    remove cluster $C_j$;
end;
```

What is the time complexity of this method? In the beginning there are n clusters, and in the end 1; thus there are n iterations of the main loop. In iteration i we have to find the closest pair of clusters among $n - i + 1$

clusters. We will see shortly that there are a variety of methods for defining the intercluster distance $\mathcal{D}(C_i, C_j)$. All of them, however, require in the first iteration that we locate the closest pair of objects. This takes $O(n^2)$ time, unless we have special knowledge about the distance between objects and so, in most cases, the algorithm requires $O(n^2)$ time, and frequently much more. Note also that the space complexity of the method is also $O(n^2)$, since all pairwise distances between objects must be available at the start of the algorithm. Thus, the method is typically not feasible for large values of n. Furthermore, interpreting a large dendrogram can be quite difficult (just as interpreting a large classification tree can be difficult).

Note that in agglomerative clustering we need distances between individual data objects to begin the clustering, and during clustering we need to be able to compute distances between groups of data points (that is, distances between clusters). Thus, one advantage of this approach (over partition-based clustering, for example) is the fact that we do not need to have a vector representation for each object as long as we can compute distances between objects or between sets of objects. Thus, for example, agglomerative clustering provides a natural framework for clustering objects that are not easily summarized as vector measurements. A good example would be clustering of protein sequences where there exist several well-defined notions of distance such as the *edit-distance* between two sequences (that is, a measure of how many basic edit operations are required to transform one sequence into another).

In terms of the general case of distances between sets of objects (that is, clusters) many measures of distance have been proposed. If the objects are vectors then any of the global score functions described in section 9.4 can be used, using the difference between the score before merger and that after merging two clusters.

However, local pairwise distance measures (that is, between pairs of clusters) are especially suited to hierarchical methods since they can be computed directly from pairwise distances of the members of each cluster. One of the earliest and most important of these is the *nearest neighbor* or *single link* method. This defines the distance between two clusters as the distance between the two closest points, one from each cluster;

$$\mathcal{D}_{sl}(C_i, C_j) = \min_{\mathbf{x},\mathbf{y}}\{d(\mathbf{x},\mathbf{y}) \mid \mathbf{x} \in C_i, \mathbf{y} \in C_j\}, \qquad (9.22)$$

where $d(\mathbf{x},\mathbf{y})$ is the distance between objects $\mathbf{x}$ and $\mathbf{y}$. The single link method is susceptible (which may be a good or bad thing, depending upon our objectives) to the phenomenon of "chaining," in which long strings of points are assigned to the same cluster (contrast this with the production of compact spherical clusters). This means that the single link method is of limited value for segmentation. It also means that the method is sensitive to small perturbations of the data and to outlying points (which, again, may be good or bad, depending upon what we are trying to do). The single link method also has the property (for which it is unique—no other measure of distance between clusters possesses it) that if two pairs of clusters are equidistant it does not matter which is merged first. The overall result will be the same, regardless of the order of merger.

The dendrogram from the single link method applied to the data in figure 9.7 is shown in figure 9.9. Although on this particular data set the results of single link clustering and that of figure 9.8 are quite similar, the two methods can in general produce quite different results.

At the other extreme from single link, *furthest neighbor,* or *complete link,* takes as the distance between two clusters the distance between the two most distant points, one from each cluster:

$$\mathcal{D}_{fl}(C_i, C_j) = \max_{\mathbf{x},\mathbf{y}}\{d(\mathbf{x},\mathbf{y}) \mid \mathbf{x} \in C_i, \mathbf{y} \in C_j\}, \tag{9.23}$$

where $d(\mathbf{x},\mathbf{y})$ is again the distance between objects $\mathbf{x}$ and $\mathbf{y}$. For vector objects this imposes a tendency for the groups to be of equal size in terms of the volume of space occupied (and not in terms of numbers of points), making this measure particularly appropriate for segmentation problems.

Other important measures, intermediate between single link and complete link, include (for vector objects) the centroid measure (the distance between two clusters is the distance between their centroids), the group average measure (the distance between two clusters is the average of all the distances between pairs of points, one from each cluster), and Ward's measure for vector data (the distance between two clusters is the difference between the total within cluster sum of squares for the two clusters separately, and the within cluster sum of squares resulting from merging the two clusters discussed above). Each such measure has slightly different properties, and other variants also exist; for example, the median measure for vector data ignores the size of clusters, taking the "center" of a combination of two clusters to be the midpoint of the line joining the centers of the two components. Since we are seeking the novel in data mining, it may well be worthwhile to experiment with several measures, in case we throw up something unusual and interesting.

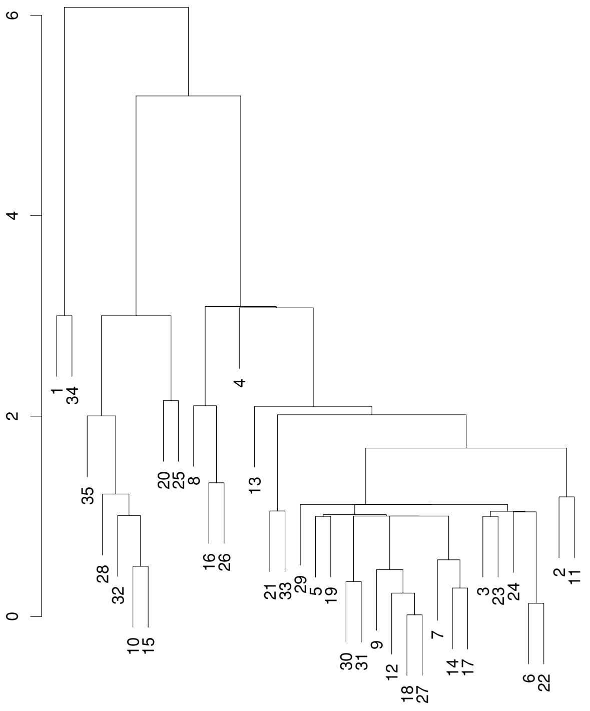


Figure 9.9 Dendrogram of the single link method applied to the data in figure 9.7.

9.5.2 Divisive Methods

Just as stepwise methods of variable selection can start with no variables and gradually add variables according to which lead to most improvement (analogous to agglomerative cluster analysis methods), so they can also start with all the variables and gradually remove those whose removal leads to least deterioration in the model. This second approach is analogous to divisive methods of cluster analysis. Divisive methods begin with a single cluster composed of all of the data points, and seek to split this into components. These further components are then split, and the process is taken as far as necessary. Ultimately, of course, the process will end with a partition in which each cluster consists of a single point.

Monothetic divisive methods split clusters using one variable at a time (so they are analogous to the basic form of tree classification methods discussed in chapter 5). This is a convenient (though restrictive) way to limit the number of possible partitions that must be examined. It has the attraction that the result is easily described by the dendrogram—the split at each node is defined in terms of just a single variable. The term *association analysis* is sometimes uses to describe monothetic divisive procedures applied to multivariate binary data. (This is not the same use as the term "association rules" described in chapter 5.)

Polythetic divisive methods make splits on the basis of all of the variables together. Any intercluster distance measure can be used. The difficulty comes in deciding how to choose potential allocations to clusters—that is, how to restrict the search through the space of possible partitions. In one approach, objects are examined one at a time, and that one is selected for transfer from a main cluster to a subcluster that leads to the greatest improvement in the clustering score.

In general, divisive methods are more computationally intensive and tend to be less widely used than agglomerative methods.

9.6 Probabilistic Model-Based Clustering Using Mixture Models

The mixture models of section 9.2.4 can also be used to provide a general framework for clustering in a probabilistic context. This is often referred to as *probabilistic model-based clustering* since there is an assumed probability model for each component cluster. In this framework it is assumed that the data come from a multivariate finite mixture model of the general form

$$f(\mathbf{x}) = \sum_{k=1}^{K} \pi_k f_k(\mathbf{x}; \theta_k), \tag{9.24}$$

where f_k are the component distributions. Roughly speaking, the general procedure is as follows: given a data set $D = \{\mathbf{x}(1), \ldots, \mathbf{x}(n)\}$, determine how many clusters K we want to fit to the data, choose parametric models for each of these K clusters (for example, multivariate Normal distributions

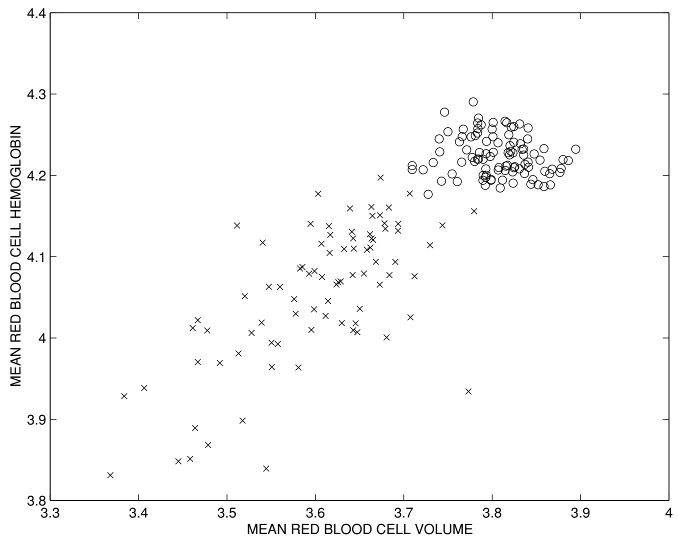


Figure 9.10 Red blood cell measurements (mean volume and mean hemoglobin concentration) from 182 individuals showing the separation of the individuals into two groups: healthy (circles) and iron deficient anemia (crosses).

are a common choice), and then use the EM algorithm of section 9.2.4 (and described in more detail in chapter 8) to determine the component parameters θ_k and component probabilities π_k from the data. (We can of course also try to determine a good value of K from the data, we will return to this question later in this section.) Typically the likelihood of the data (given the mixture model) is used as the score function, although other criteria (such as the so-called classification likelihood) can also be used. Once a mixture decomposition has been found, the data can then be assigned to clusters—for example, by assigning each point to the cluster from which it is most likely to have come.

To illustrate the idea, we apply the method to a data set where the true class labels are in fact known but are removed and then "discovered" by the algorithm.

Example 9.4 Individuals with chronic iron deficiency anemia tend to produce red blood cells of lower volume and lower hemoglobin concentration than nor-

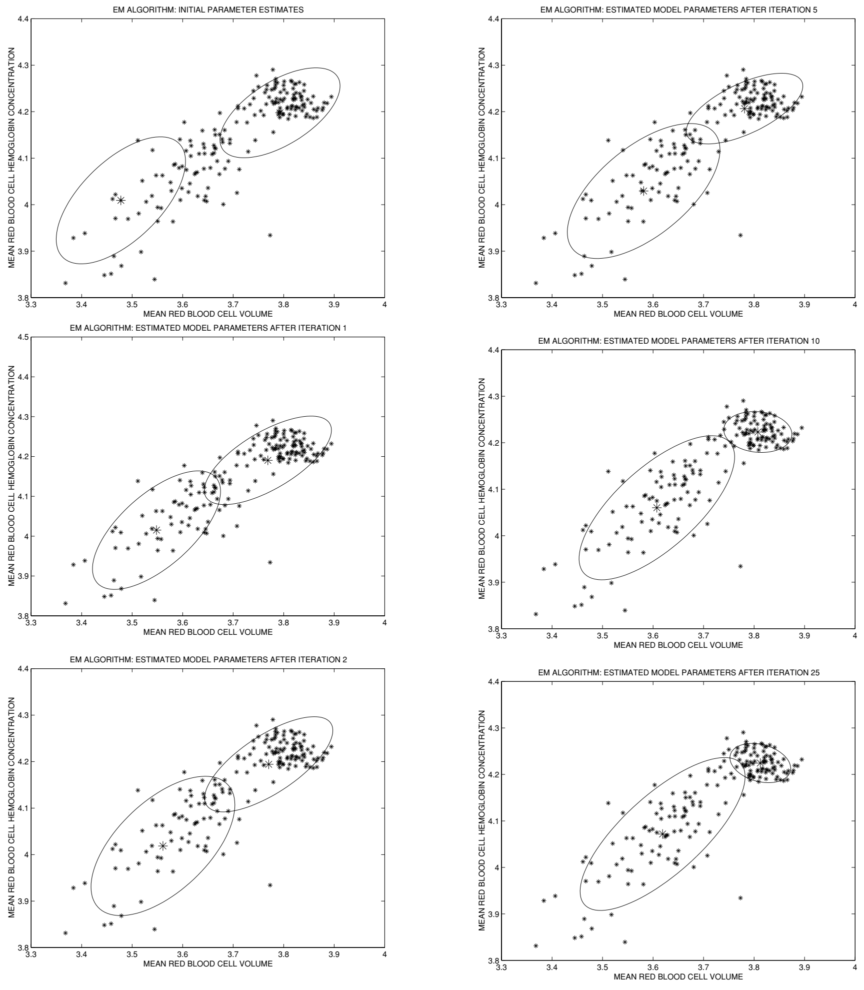


Figure 9.11 Example of running the EM algorithm on the red blood cell measurements of figure 9.10. The plots (running top to bottom, left first, then right) show the 3σ covariance ellipses and means of the fitted components at various stages of the EM algorithm.

mal. A blood sample can be taken to determine a person's mean red blood cell volume and hemoglobin concentration. Figure 9.10 shows a scatter plot of the bivariate mean volume and hemoglobin concentrations for 182 individuals with labels determined by a diagnostic lab test. A normal mixture model with $K = 2$ was fit to these individuals, with the labels removed. The results are shown in figure 9.11, illustrating that a two-component normal mixture appears to capture the main features of the data and would provide an excellent clustering if the group labels were unknown (that is, if the lab test had not been performed). Figure 9.2 verifies that the likelihood (or equivalently, log-likelihood) is nondecreasing as a function of iteration number. Note, however, that the rate of convergence is nonmonotonic; that is, between iterations 5 and 8 the rate of increase in log-likelihood slows down, and then increases again from iterations 8 to 12.

The red blood cell example of figure 9.11 illustrates several features of the probabilistic approach:

- The probabilistic model provides a full distributional description for each component. Note, for example, the difference between the two fitted clusters in the red blood cell example. The normal component is relatively compact, indicating that variability across individuals under normal circumstances is rather low. The iron deficient anemia cluster, on the other hand, has a much greater spread, indicating more variability. This certainly agrees with our common-sense intuition, and it is the type of information that can be very useful to a scientist investigating fundamental mechanisms at work in the data-generating process.

- Given the model, each individual (each data point) has an associated K-component vector of the probabilities that it arose from each group, and that can be calculated in a simple manner using Bayes rule. These probabilities are used as the basis for partitioning the data, and hence defining the clustering. For the red blood cell data, most individuals lie in one group or the other with probability near 1. However, there are certain individuals (close to the intersection of the two clouds) whose probability memberships will be closer to 0.5—that is, there is uncertainty about which group they belong to. Again, from the viewpoint of exploring the data, such data points may be valuable and worthy of detection and closer study (for example, individuals who may be just at the onset of iron deficient anemia).

- The score function and optimization procedures are quite natural in a probabilistic context, namely likelihood and EM respectively. Thus, there is a well-defined theory on how to fit parameters to such models as well as a large library of algorithms that can be leveraged. Extensions to MAP and Bayesian estimation (allowing incorporation of prior knowledge) are relatively straightforward.

- The basic finite mixture model provides a principled framework for a variety of extensions. One useful idea, for example, is to add a $(K+1)$th noise component (for example, a uniform density) to pick up outliers and background points that do not appear to belong to any of the other K components; the relative weight π_{K+1} of this background component can be learned by EM directly from the data.

- The method can be extended to data that are not in p-dimensional vector form. For example, we can cluster sequences using mixtures of probabilistic sequence models (for example, mixtures of Markov models), cluster curves using mixtures of regression models, and so forth, all within the same general EM framework.

These advantages come at a certain cost. The main "cost" is the assumption of a parametric model for each component; for many problems it may be difficult a priori to know what distributional forms to assume. Thus, model-based probabilistic clustering is really only useful when we have reason to believe that the distributional forms are appropriate. For our red blood cell data, we can see by visual inspection that the normal assumptions are quite reasonable. Furthermore, since the two measurements consist of estimated means from large samples of blood cells, basic statistical theory also suggests that a normal distribution is likely to be quite appropriate.

The other main disadvantage of the probabilistic approach is the complexity of the associated estimation algorithm. Consider the difference between EM and K-means. We can think of K-means as a stepwise approximation to the EM algorithm applied to a mixture model with Normal mixture components (where the covariance matrices for each cluster are all assumed to be the identity matrix). However, rather than waiting until convergence is complete before assigning the points to the clusters, the K-means algorithm reassigns them at each step.

Example 9.5 Suppose that we have a data set where each variable X_j is 0/1 valued—for example, a large transaction data set where $x_j = 1$ (or 0) represents whether a person purchased item j (or not). We can apply the mixture modeling framework as follows: assume that given the cluster k, the variables

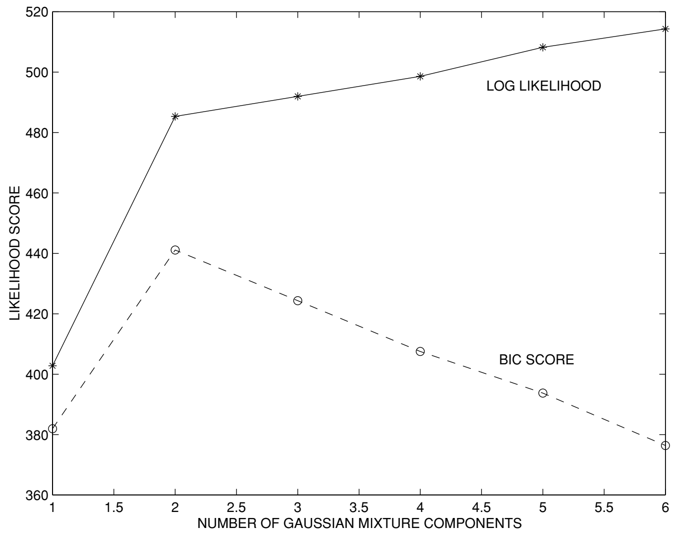


Figure 9.12 Log-likelihood and BIC score as a function of the number of Normal components fitted to the red blood cell data of figure 9.11.

are conditionally independent (as discussed in section 9.2.7); that is, that we can write

$$p_k(\mathbf{x};\theta_k) = \prod_{j=1}^{p} p_k(x_j;\theta_{kj}).$$

To specify a model for the data, we just need to specify the probability of observing value 1 for the jth variable in the kth component. Denoting this probability by θ_{kj}, we can write the component density for the kth component as

$$p_k(x_j;\theta_{kj}) = \theta_{kj}^{x_j}(1-\theta_{kj})^{1-x_j}, \quad 1 \le k \le K,$$

which is a convenient way of writing the probability of observing value x_j in component k of the mixture model. The full mixture equation for observation $\mathbf{x}(i)$ is the weighted sum of these component densities:

$$\begin{aligned} p(\mathbf{x}(i)) &= \sum_{k=1}^{K} \pi_k p_k(\mathbf{x}(i);\theta_k) \\ &= \sum_{k=1}^{K} \pi_k \prod_j \theta_{kj}^{x_j(i)}(1-\theta_{kj})^{1-x_j(i)} \qquad (9.25) \\ & \qquad (9.26) \end{aligned}$$

where $x_j(i)$ indicates whether person i bought product j or not.

The EM equations for this model are quite simple. Let $p(k|i)$ be the probability that person i belongs to cluster k. By Bayes rule, and given a fixed set of parameters θ, this can be written as:

$$p(k|i) = \frac{\pi_k \prod_j \theta_{kj}^{x_j(i)} (1 - \theta_{kj})^{1-x_j(i)}}{p(\mathbf{x}(i))}, \tag{9.27}$$

where $p(\mathbf{x}(i))$ is as defined in equation 9.26. Calculation of $p(k|i)$ takes $O(nK)$ steps since it must be carried out for each individual i and each cluster k. Calculation of these "membership probabilities" is in effect the E-step for this problem.

The M-step is simply a weighted estimate of the probability of a person buying item j given that they belong to cluster k:

$$\theta_{kj}^{\text{new}} = \frac{\sum_{i=1}^{n} p(k|i) x_j(i)}{\sum_{i=1}^{n} p(k|i)}, \tag{9.28}$$

where in this equation, observation $x_j(i)$ is weighted by the probability $p(k|i)$, namely the probability that individual i was generated by cluster k (according to the model). A particular product j purchased by individual i is in effect assigned fractionally (via the weights $p(k|i)$, $1 \leq k \leq K$) to the K cluster models. This M-step requires $O(nKp)$ operations since the weighted sum in the numerator must be performed over all n individuals, for each cluster k, and for each of the p parameters (one for each variable in the independence model). If we have I iterations of the EM algorithm in total we get $O(IKnp)$ as the basic complexity, which can be thought of as I times K times the size of the data matrix.

For really large data sets that reside on disk, however, doing I passes through the data set will not be computationally tractable. Techniques have been developed for summarizing cluster representations so that the data set can in effect be compressed during the clustering method. For example, in mixture modeling many data points "gravitate" to one component relatively early in the computation; that is, their membership probability for this component approaches 1. Updating the membership of such points could be omitted in future iterations. Similarly, if a point belongs to a group of points that always share cluster membership, then the points can be represented by using a short description.

To conclude our discussion on probabilistic clustering, consider the problem of finding the best value for K from the data. Note that as K (the number of clusters) is increased, the value of the likelihood at its maximum cannot decrease as a function of K. Thus, likelihood alone cannot tell us directly about which of the models, as a function of K, is closest to the true data generating process. Moreover, the usual approach of hypothesis testing (for example, testing the hypothesis of one component versus two, two versus three, and so forth) does not work for technical reasons related to the mixture likelihood. However, a variety of other ingenious schemes have been developed based to a large extent on approximations of theoretical analyses. We can identify three general classes of techniques in relatively widespread use:

Penalized Likelihood: Subtract a term from the maximizing value of the likelihood. The BIC (Bayesian Information Criterion) is widely used. Here

$$S_{BIC}(M_K) = 2S_L(\hat{\theta}_K\,; M_K) + d_K \log n \tag{9.29}$$

where $S_L(\theta_K\,; M_K)$ is the minimizing value of the negative log-likelihood and d_K is the number of parameters, both for a mixture model with K components. This is evaluated from $K = 1$ up to some $K_{\max}$ and the minimum taken as the most likely value of K. The original derivation of BIC was based on asymptotic arguments in a different (regression) context, arguments that do not strictly hold for mixture modeling. Nonetheless, the technique has been found to work quite well in practice and has the merit of being relatively cheap to compute relative to the other methods listed below. In figure 9.12 the negative of the BIC score function is plotted for the red blood cell data and points to $K = 2$ as the best model (recall that there is independent medical knowledge that the data belong to two groups here, so this result is quite satisfying). There are a variety of other proposals for penalty terms (see chapter 7), but BIC appears to be the most widely used in the clustering context.

Resampling Techniques: We can use either bootstrap methods or cross-validated likelihood using resampling ideas as another approach to generate "honest" estimates of which K value is best. These techniques have the drawback of requiring significantly more computation than BIC—for example, ten times more for the application of ten-fold cross-validation. However, they do provide a more direct assessment of the quality of the models, avoiding the need for the assumptions associated with methods such as BIC.

Bayesian Approximations: The fully Bayesian solution to the problem is to estimate a distribution $p(K|D)$,—that is, the probability of each K value given the data, where all uncertainty about the parameters is integrated out in the usual fashion. In practice, of course, this integration is intractable (recall that we are integrating in a d_K-dimensional space) so various approximations are sought. Both analytic approximations (for example, the Laplace approximation about the mode of the posterior distribution) and sampling techniques (such as Markov chain Monte Carlo) are used. For large data sets with many parameters in the model, sampling techniques may be computationally impractical, so analytic approximation methods tend to be more widely used. For example, the AUTOCLASS algorithm of Cheeseman and Stutz (1996) for clustering with mixture models uses a specific analytic approximation of the posterior distribution for model selection. The BIC penalty-based score function can also be viewed as an approximation to the full Bayesian approach.

In a sense, the formal probabilistic modeling implicit in mixture decomposition is more general than cluster analysis. Cluster analysis aims to produce merely a partition of the available data, whereas mixture decomposition produces a description of the distribution underlying the data (that this distribution is composed of a number of components). Once these component probability distributions have been identified, points in the data set can be assigned to clusters on the basis of the component that is most likely to have generated them. We can also look at this another way: the aim of cluster analysis is to divide the data into naturally occurring regions in which the points are closely or densely clustered, so that there are relatively sparse regions between the clusters. From a probability density perspective, this will correspond to regions of high density separated by valleys of low density, so that the probability density function is fundamentally multimodal. However, mixture distributions, even though they are composed of several components, can well be unimodal.

Consider the case of a two-component univariate normal mixture. Clearly, if the means are equal, then this will be unimodal. In fact, a sufficient condition for the mixture to be unimodal (for all values of the mixing proportions) when the means are different is $\mid \mu_1 - \mu_2 \mid \leq 2\min(\sigma_1, \sigma_2)$. Furthermore, for every choice of values of the means and standard deviations in a two-component normal mixture there exist values of the mixing proportions for which the mixture is unimodal. This means that if the means are close enough there will be just one cluster, even though there are two components. We can still use the mixture decomposition to induce a clustering, by assigning each data point to the cluster from which it is most likely to have come, but this is unlikely to be a useful clustering.

9.7 Further Reading

A general introduction to parametric probability modeling is Ross (1997) and an introduction to general concepts in multivariate data analysis is provided by Everitt and Dunn (1991). General texts on mixture distributions include Everitt and Hand (1981), Titterington, Smith, and Makov (1985), McLachlan and Basford (1988), Böhning (1998), and McLachlan and Peel (2000). Diebolt and Robert (1994) provide an example of the general Bayesian approach to mixture modeling. Statistical treatments of graphical models include those by Whittaker (1990), Edwards (1995), Cox and Wermuth (1996), and Lauritzen (1996). Pearl (1988) and Jensen (1996) emphasize representational and computational aspects of such models, and the edited collection by Jordan (1999) contains recent research articles on learning graphical models from data. Friedman and Goldszmidt (1996) and Chickering, Heckerman, and Meek (1997) provide details on specific algorithms for learning graphical models from data. Della Pietra, Della Pietra, and Lafferty (1997) describe the application of Markov random fields to text modeling, and Heckerman et al. (2000) describe use a form of Markov random fields for model-based collaborative filtering. Bishop, Fienberg, and Holland (1975) is a standard reference on log-linear models.

There are now many books on cluster analysis. Recommended ones include Anderberg (1973), Späth (1985), Jain and Dubes (1988), and Kaufman and Rousseeuw (1990). The distinction between dissection and finding natural partitions is not always appreciated, and yet it can be important and should not be ignored. Examples of authors who have made the distinction include Kendall (1980), Gordon (1981), and Späth (1985). Marriott (1971) showed that the criterion $K^2 tr(\mathbf{W})$ was asymptotically constant for optimal partitions of a multivariate uniform distribution. Krzanowski and Marriott (1995), table 10.6, give a list of updating formula for clustering criteria based on $\mathbf{W}$. Maximal predictive classification was developed by Gower (1974). The use of branch and bound to extend the range of exhaustive evaluation of all possible clusterings is described in Koontz, Narendra, and Fukunaga (1975) and Hand (1981). The K-means algorithm is described in MacQueen (1967), and the ISODATA algorithm is described in Hall and Ball (1965). Kaufman and Rousseeuw (1990) describe a variant in which the "central point" of each cluster is an element of that cluster, rather than the centroid of the elements. A review of early work on mathematical programming methods applied in cluster analysis is given by Rao (1971) with a more recent review provided by Mangasarian (1996).

One of the earliest references to the single link method of cluster analysis was Florek et al. (1951), and Sibson (1973) was important in promoting the idea. Lance and Williams (1967) presented a general formula, useful for computational purposes, that included single link and complete link as special cases. The median method of cluster analysis is due to Gower (1967). Lambert and Williams (1966) describe the "association analysis" method of monothetic divisive partitioning. The polythetic divisive method of clustering is due to MacNaughton-Smith et al. (1964). The overlapping cluster methods are due to Shepard and Arabie (1979).

Other formalisms for clustering also exist. For example, Karypis and Kumar (1998) discuss graph-based clustering algorithms. Zhang, Ramakrishnan, and Livny (1997) describe a framework for clustering that is scalable to very large databases. There are also countless applications of clustering. For a cluster analytic study of whiskies, see Lapointe and Legendre (1994). Eisen et al. (1998) illustrate the application of hierarchical agglomerative clustering to gene expression data. Zamir and Etzioni (1998) describe a clustering algorithm specifically for clustering Web documents.

Probabilistic clustering is discussed in the context of mixture models in Titterington, Smith, and Makov (1985) and in McLachlan and Basford (1987). Banfield and Raftery (1993) proposed the idea (in a mixture model context) of adding a "cluster" that pervades the whole space by superimposing a separate Poisson process that generated a low level of random points throughout the entire space, so easing the problem of clusters being distorted due to a handful of outlying points. More recent work on model-based probabilistic clustering is described in Celeux and Govaert (1995), Fraley and Raftery (1998), and McLachlan and Peel (1998). The application of mixture models to clustering sequences is described in Poulsen (1990), Smyth (1997), Ridgeway (1997), and Smyth (1999). Mixture-based clustering of curves for parametric models was originally described by Quandt and Ramsey (1978) and Späth (1979) and later generalized to the non-parametric case by Gaffney and Smyth (1999). Jordan and Jacobs (1994) provide a generalization of standard mixtures to a mixture-based architecture called "mixtures of experts" that provides a general mixture-based framework for function approximation.

Studies of tests for numbers of components of mixture models are described in Everitt (1981), McLachlan (1987), and Mendell, Finch, and Thode (1993). An early derivation of the BIC criterion is provided by Shibata (1978). Kass and Raftery (1995) provide a more recent overview including a justification for the application of BIC to a broad range of model selection tasks. The bootstrap method for determining the number of mixture model components was introduced by McLachlan (1987) and later refinements can be found in Feng and McCulloch (1996) and McLachlan and Peel (1997). Smyth (2000) describes a cross-validation approach to the same problem. Cheeseman and Stutz (1996) outline a general Bayesian framework to the problem of model-based clustering, and Chickering and Heckerman (1998) discuss an empirical study comparing different Bayesian approximation methods for finding the number of components K.

Different techniques for speeding up the basic EM algorithm for large data sets are described in Neal and Hinton (1998), Bradley, Fayyad, and Reina (1998), and Moore (1999).

Cheng and Wallace (1993) describe an interesting application of hierarchical agglomerative clustering to the problem of clustering spatial atmospheric measurements from the Earth's upper atmosphere. Smyth, Ide, and Ghil (1999) provide an alternative analysis of the same data using Normal mixture models, and use cross-validated likelihood to provide a quantitative confirmation of the earlier Cheng and Wallace clusters. Mixture models in haematology are described in McLaren (1996). Wedel and Kamakura (1998) provide an extensive review of the development and application of mixture models in consumer modeling and marketing applications. Cadez et al. (2000) describe the application of mixtures of Markov models to the problem of clustering individuals based on sequences of page-requests from massive Web logs.

The antenna data of figure 9.4 are described in more detail in Smyth (1994) and the red blood cell data of figure 9.10 are described in Cadez et al. (1999).