Principles of Data Mining

David Hand
Imperial College

Heikki Mannila
Microsoft Research

Padhraic Smyth
University of California, Irvine

MIT Press, to appear
Draft — November 15, 1999

Chapter 9

Descriptive Modeling

9.1 Introduction

In Chapter 1 we defined 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. Typically 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. We further described this distinction in more detail in other chapters–for example, in Chapter 6. As well as distinguishing between models and patterns, earlier chapters also noted the distinction 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 one to study the most important aspects of the data without these being obscured by the sheer size of the data set. In contrast, a predictive model has the specific objective of allowing one to predict the value of some target characteristic of an object on the basis of observed values of other characteristics.

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

In Chapter 6 we noted the distinction between mechanistic and empirical models—the former being based on some underlying theory about the mechanism through which the data arose and the latter being simply a description of the observed data. Data mining is usually concerned with the latter situation. The fundamental objective is to produce insight and understanding about the structure of the data, and to enable one to see what are its important features. Beyond this, of course, one hopes one might discover unsuspected structure and structure which is interesting and valuable in some sense. A good model can also be thought of as *generative* in the sense that data randomly generated according to the model will have the same characteristics as the real data from which the model was produced. If such randomly generated data has features not possessed by the original data, or does not possess features which the original data does (such as, for example, correlations between variables), then the model is a poor one: it is failing in its job of adequately summarizing the data.

This chapter focuses on specific techniques and algorithms for fitting descriptive models to data. As such, it builds on many of the ideas introduced in earlier chapters; the principles of uncertainty in Chapter 4, decomposing data mining algorithms into basic components in 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). In a single chapter we cannot hope to examine all possible models types in detail. What we have done is look at just some of the more important types.

One point is worth making at the start. Since we are concerned here with global models, with structures which 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 (we are not concerned with patterns). This is good news from the point of view of scalability: 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

Describing data in terms of its underlying distribution or density function is a fundamental *descriptive* technique in data analysis. We will revert to our notation of a p-dimensional data matrix, with variables $X_1, \ldots, X_p$. Our goal is to model the joint distribution or density $p(X_1, \ldots, X_p)$ as first encountered in Chapter 4. We will assume for now that the X's are real-valued, and thus refer to densities rather than distributions in this section: however, the discussion carries over directly to categorical data and distribution functions.

The joint density in a certain sense provides us with "complete" information about the variables $X_1, \ldots, X_p$. Given the joint density, one can answer any question about the relationships among any subset of variables (e.g., what is $p(x_3|x_7)$? are X_3 and X_7 independent?) as well as being able to predict the value(s) of any variable(s) given the values of any subset of the remaining variables (e.g., calculate $p(x_1, x_2|x_3, x_4)$). For example, our intuition tells us that a joint probability density model should contain much more information than (say) a simple conditional density model (such as $p(x_1|x_2, \ldots, x_p)$) which might be used for prediction. One way to see this is that any conditional density can always be obtained from the appropriate joint density, i.e., it is implicitly specified by it.

Thus, estimating a full density may involve a more difficult modeling and estimation problem than estimating the "simpler" conditional density. Why model the full joint density at all then? There are a number of situations where knowing the joint density is useful and desirable. For example, we may be interested in the *modes* of the density (for real-valued

X's). 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 scatter-plot we will just see a mass of points, many overlaid on top of each other. If instead we estimate the joint density p(income, spending) (where we have yet to describe how this would be done), we would get a density function of the two dimensions which could be plotted as a contour map or as a 3d 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 may indicate regions where (for one reason or another) this particular bank has 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. Thus, the density estimate (once constructed) can be used in a variety of ways to characterize structure and patterns in data.

Another example of the usefulness of knowing the full density models stems from the task of generating approximate answers to queries for large databases (also known as *query selectivity estimation*). The task in selectivity estimation is the following: given a query, i.e., a condition on the observations, estimate the selectivity of the query, that is, the fraction of rows that satisfy the condition. Such estimates are needed for example 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 in principle use it to obtain approximate selectivities in a computationally efficient manner.

9.2.2 Score Functions for Learning Probability Distributions and Densities

In the discussions below on distribution and density estimation we will implicitly assume that we are using the log-likelihood score function unless otherwise stated. Let $p(\mathbf{x};\theta)$ be a density (or distribution) function for $\mathbf{x}$ with model parameters θ. We have already discussed in Chapters 4 and 7 that the "canonical" score function for evaluating the quality of a particular set of parameters θ (given that p is fixed) is the log-likelihood $S_l = \log p(D|\theta)$ where $D = \{\mathbf{x}(1), \ldots, \mathbf{x}(n)\}$ is the observed data. Assuming (again) that the data are generated independently conditioned on the model p, we have

$$S_l(\theta) = l(\theta) = \sum_{i=1}^{n} \log \hat{p}(\mathbf{x}(i);\theta). \tag{9.1}$$

Typically we can maximize this score function in closed form if p is a simple parametric form (e.g., a Gaussian or binomial or any of the density or distributional forms in Appendix A). If p is more complex then iterative optimization methods may be required to search for the θ which maximize $S_l(\theta)$ (for example the use of EM when p is a mixture model).

If we have models of different complexity under consideration (e.g., Gaussians densities with different covariance structures) then as discussed in Chapter 7 the log-likelihood is insufficient as a method for indicating which model is closest to "truth" (the unknown density which gave rise to the observed data) since it can always be increased by increasing the representational power (or complexity) of the model. Thus, if we want to select one model among many, we can use score functions which try to estimate how well each model will perform on out-of-sample data, i.e., how well they will generalize For example, under the usual assumption of independent sampling, the *validation likelihood* is defined as

$$S_{vl}(M_k) = \sum_{\mathbf{x} \in D_v} \log p_{\hat{M_k}}(\mathbf{x}|\theta), \quad 1 \leq k \leq K, \tag{9.2}$$

where the points $\mathbf{x}$ are from the validation data set D_v, the parameters θ were estimated (e.g., via maximum likelihood) on the disjoint training data $D_t = D \setminus D_v$, and there are K models under consideration. Good models will assign high probability to new data points and poor models will assign low probability. The validation likelihood can be particularly useful if data are plentiful but the cost of computing it (say over a million validation data points) may be somewhat slow if we are repeatedly calculating it during the middle of a search. If data are less plentiful we can use cross-validation instead, but now of course there is a further computational penalty. A relatively cheap score function (in that the time to compute the penalty term is independent of how many data points one has) is the BIC (Bayesian Information Criterion) score function, which we will recall from Chapter 7 is defined as:

$$S_{BIC}(M_k) = S_l(\theta_k; M_k) - \frac{d_k}{2} \log n \quad 1 \leq k \leq K \tag{9.3}$$

where d_k is the number of parameters in model M_k and $S_l(\theta_k; M_k)$ is the maximizing value of the log-likelihood (achieved at θ_k).

9.2.3 Parametric Density Models

There are two general classes of density function model structures, parametric and non-parametric models. *Parametric models* assume a particular functional form (usually relatively simple) for the density function, such as a uniform, Normal (Gaussian) distribution, exponential, Poisson, and so forth (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. Important distinctions between these model structures include the notions of location (mean), scale (variance) and symmetry, all of which should be constrained by underlying prior knowledge about the variable being measured (for example, knowing that a variable such as income can only be positive should be reflected in the choice of density function). Parametric density models can usually be characterized by a relatively small number of parameters. For example, the p-dimensional Gaussian distribution is defined as

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

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 X's. The *parameters* of the model are the mean vector and the covariance matrix (thus, $p + p(p+1)/2$ parameters in all).

The functional form above 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 which takes into account (through the covariance matrix Σ) correlations in p-space when calculating distance. The denominator in Equation 9.4 is simply a normalizing constant (call it C) to ensure that the function integrates to 1 (i.e., is a true probability density). Thus, we can write our Gaussian model in significantly simplified form as

$$p(\mathbf{x}) = \frac{1}{C} e^{-r^2_\Sigma(\mathbf{x},\mu)/2} \tag{9.5}$$

If we were to plot (say for $p = 2$) all of the points $\mathbf{x}$ which have the same fixed values of $r^2_\Sigma(\mathbf{x},\mu)$, we would find that they trace out an ellipse in 2-space (more generally, a hyperellipsoid in p-space), where the ellipse is centred at μ. Thus, the Gaussian density function if plotted in contour form (say in 2d) consists of elliptical contours centred about the mean. The height of the contours fall off exponentially from the center as a function of $r^2_\Sigma(\mathbf{x},\mu)$. The shape 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.

For high-dimensional data (large p) the number of parameters in the Gaussian 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 Gaussian case to assuming that the covariance matrix has a diagonal structure (and, hence, only has p parameters). (Note that if we assume that Σ is diagonal it is easy to show that the p-dimensional multivariate Gaussian density factors into a product of p univariate Gaussians, 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, i.e., that the data has the same variance for all p variables as well as being independent.

A somewhat less restrictive assumption would be to impose a block diagonal structure on Σ if we suspect that there are groups of variables (the "blocks") which may be dependent, but that there is no reason to model dependence across the groups. Such assumptions of course may be quite inaccurate in practice and if data are relatively plentiful it is always a good idea to check one's assumptions The multivariate normal distribution has the useful 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 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 draw a graph in which there are no edges linking the nodes corresponding to variables which have a small value in this inverse matrix (recall that we discussed graphical models in Chapter 6 and we will return to this topic in more detail later in this Chapter).

We can also use the notion of cross-validation to help select an appropriate covariance structure for Σ. Remember that the log-likelihood cannot increase as we remove degrees of freedom from the model, e.g., as we place constraints on our covariance model. So instead we can use cross-validated estimate of the log-likelihood (or an estimate from a large validation data set) to select which among several structures for Σ has the best fit "out of sample." We can further generalize this idea and estimate a covariance matrix of the form

$$\Sigma_\lambda = \lambda \Sigma_{\mathrm{diag}} + (1-\lambda)\Sigma_{\mathrm{unconstrained}} \tag{9.6}$$

where λ is a mixing weight between 0 and 1, Σ_{diag} is the estimated covariance matrix when Σ is constrained to be diagonal and $\Sigma_{\mathrm{unconstrained}}$ is the unconstrained estimate of Σ. The weight λ can be chosen so as to maximize the cross-validated (or directly validated) log-likelihood score function. This is just one example of a form of data-driven *regularization* where we "smooth" the full unconstrained model with a much more constrained model in a data-driven manner. There are numerous other ways to achieve the same general effect including (for example) fully Bayesian techniques for parameter estimation as discussed in Chapter 4.

The use of diagnostics is always recommended when fitting a parametric density function or distribution. Specific statistical goodness-of-fit tests are available for example for the multivariate Gaussian model, but even simple "eye-balling" can be revealing. For example, if we were modeling financial data, then we would know that a variable such as salary can only be positive. Hence, it may not make much sense to model this with a Gaussian density function since one of the tails will extend below zero. Furthermore, even a cursory examination of the data will reveal that it has a non-symmetric distribution, again invalidating the Gaussian assumption. (Note that if we are using a multivariate Gaussian model then we are effectively assuming that each individual marginal density is also Gaussian in form). For the income data it might be worth examining (for example) a transformation of the measurements to see if the transformed data look "more Gaussian". For example, a log-transformation can be helpful with data such as this (using a Gaussian model for $\log(x)$ is known as a lognormal model).

The Gaussian 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 linear correlation (as captured by the covariance matrix Σ). Thus, non-linear relationships can not be captured, nor can any form of bimodality or grouping. The Gaussian model is widely used in practice, but some care and attention must be taken to ensure that it is reasonable. In general, all of the well-known parametric models have similar constraints. Their power lies in their generality and the fact that they often do model well-known natural phenomena. However, it is worth noting that many of these models were originally developed with univariate measurements in mind, and that we increase p, the chances that all of the p variables obey the same simple multivariate parametric form becomes less and less likely in practice.

9.2.4 Mixture Distributions and Densities

In Chapter 6 we saw how the parametric model can be generalized to allow *mixtures* of parametric components, i.e., linear combinations of simpler unimodal density models. This can be viewed as the next natural step in complexity in our discussion of density modeling: the generalization from parametric unimodal densities to weighted linear combinations of 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 where we are not sure ahead of time what (if any) 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 which have been collected together to form one large data set. To illustrate this point, consider Figure XXX2 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 which 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

$$f(x) = p\frac{(\lambda_1)^x e^{-\lambda_1}}{x!} + (1-p)\frac{(\lambda_2)^{52-x} e^{-\lambda_2}}{(52-x)!}. \tag{9.7}$$

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 parameter of the two component Poisson models.

Here p is the probability that a person belongs to the first group. Then, given that they do, the expression $\lambda_1^x e^{-\lambda_1}/x!$ gives the probability that they will use their card x times in the year. Likewise, $1-p$ is the probability that they belong to the second group and $\lambda_2^{52-x} e^{-\lambda_2}/(52-x)!$ gives the 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 2-stage generative process for a particular individual. In the first step there is a probability p (and $1-p$) that the individual has been "assigned" to one group or the other. In the second step, an observation x is generated for that person according to the component distribution they were assigned to in the first step. Note that a key idea here is that λ_1 and λ_2 are different, i.e., the two component distributions are different so that the overall model ends up being a weighted combination of two different underlying models.

Equation 9.7 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. The overall distribution consists of a mixture of two Poisson components. 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 which led to it, it may also be a more realistic description of what is underlying the data. (In fact, as it happens even this model is not a very good fit, and deeper exploration is required.) These two aspects—the extra flexibility of the models consequent on the larger number of parameters and arguments based on suspicion of a heterogeneous— mean that mixture models are widely used for modeling distributions which are more complicated than simple standard forms.

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

$$f(\mathbf{x}) = \sum_{k=1}^{K} p_k f_k(\mathbf{x}; \theta_k), \tag{9.8}$$

where p_k is the probability that an observation will come from the kth component (the so-called kth *mixing proportion*), 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 term 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 p_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.

9.2.5 The EM Algorithm for Mixture Models

Over the years, many methods have been applied in estimating the parameters of mixture distributions given a data set $D = \{\mathbf{x}(1), \ldots, \mathbf{x}(n)\}$ and a particular mixture form. One of the more widely used modern methods in this context is the EM approach. As discussed in Chapter 8, EM can be viewed as a general iterative optimization algorithm for maximizing a likelihood score function given a probabilistic model with missing data. Here the model is a mixture model and the missing data can be thought of as missing labels telling us which component each data point $\mathbf{x}(i)$ originated from. If we knew these component 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 don't 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) one gets $O(Kn)$. The space-complexity is typically $O(Kn)$ to store the K membership probability vectors for each of the n data points $\mathbf{x}(i)$. EM often provides a large increase in likelihood over the first few iterations and then can slowly asymptote to its final value; however the likelihood function as a function of iterations need not be concave (e.g., see Figure 9.9 below). Nonetheless, for many data sets and models one can often find a reasonable 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 widely used to avoid poor local maxima. Note that as either (or both) K and p increase, the chances of finding local maxima 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 then the likelihood will increase without limit. The maximum likelihood solution in this case is likely to be of limited value. There are various ways round 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 maximizing the MAP score function (for example) instead of the likelihood. Here the priors can serve to provide 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 (e.g., one replaces the M-step with an MAP=step, and so forth).

Another problem which 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,

$$\sum_{k=1}^{c} p_k f(x; \theta_k) = \sum_{j=1}^{c'} p'_j f(x; \theta'_j), \tag{9.9}$$

implies that $c = c'$, and that for all k there is some j such that $p_k = p'_j$ 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.

Non-identifiability 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 which occur in the data. Thus, there is no way of estimating the proportions which are separately due to each component Bernoulli, or the parameters of those components

9.2.6 Non-Parametric Density Estimation

In Section 6.XXX we briefly discussed the idea of modeling a regression function by taking a weighted average of y measurements about the point of interest. We can use a very similar idea to estimate the density in a local data-driven fashion instead of using a global parametric model. For example, a *histogram* is a relatively primitive version of this idea, where we simply count the number of points that fall in certain bins. Our estimate for the density is simply 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 non-smooth 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 problem are exacerbated when we move beyond the one-dimensional histogram to a p-dimensional histogram. Nonetheless, it should be pointed out that 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.

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 in Equation **??**)

$$p(x) = \sum_{i=1}^{N} w_i, \qquad w_i = K\left(\frac{x - x(i)}{h}\right) \tag{9.10}$$

where $p(x)$ is the kernel density estimate at a query point x, where $K(t)$ is the kernel function (e.g., $K(t) = 1 - |t|, t \leq 1; K(t) = 0$ otherwise) and h is the *bandwidth* of the kernel. Intuitively, the density at x is proportion to the sum of weights about x, which in turn depend on the proximity of the N points in the training data. As with regression, 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, i.e., no summarization occurs.

The kernel function K is usually chosen as a smooth unimodal function (such as a Gaussian or triangular kernel); the precise shape is 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 are receiving significant weight in the sum and the estimate of the density is very smooth. If h is relatively small, the kernel estimate is being determined by the small number of points which are close to x, and the estimate of the density is more sensitive locally to the data (more "spiky" in appearance). In practice, choosing h determines what kind of density function the method will produce, and various techniques (such as cross-validation) exist for finding a good bandwidth from the data itself. Under appropriate assumptions these kernel models are flexible enough to approximate *any* smooth density function, if h is chosen appropriately, which 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 which might not otherwise be visible.

Display 9.1 *Figure 9.1 shows an example of different density estimates for the variable* ethanol *(E). The histogram (top left) is quite "rough" and noisy, at least for this particular choice of bin widths and bin locations. The Gaussian 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 which 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. More automated techniques, such as cross-validation, are discussed in Chapter xxx.*

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 as a product of 1-dimensional kernels, each with its own bandwidth, which keeps the number of parameters (the bandwidths 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 further away from each other than we might expect intuitively. In fact, if we wanted to keep our approximation error constant as p increases, the number of data points we need grows exponentially with p. This is rather unfortunate and means in practice that kernel models are really only practical for relatively low-dimensional problems.

Display 9.2 *The following example is taken from Silverman (1986) and illustrates emphatically the difficulties of density estimation in high dimensions. Consider data which is simulated from a multivariate Gaussian density with unit covariance matrix and mean (0,0,...,0). Assume that the bandwidth h is always chosen to minimize the mean square error at the mean. Silverman calculated the number of data points required to ensure that relative mean square error at zero is less than 0.1, i.e., that* $E[(\hat{p}(\mathbf{x}) - p(\mathbf{x}))^2]/p(\mathbf{x})^2 < 0.1$, *at* $\mathbf{x} = 0$ *and where* $p(\mathbf{x})$ *is the true Gaussian density,*

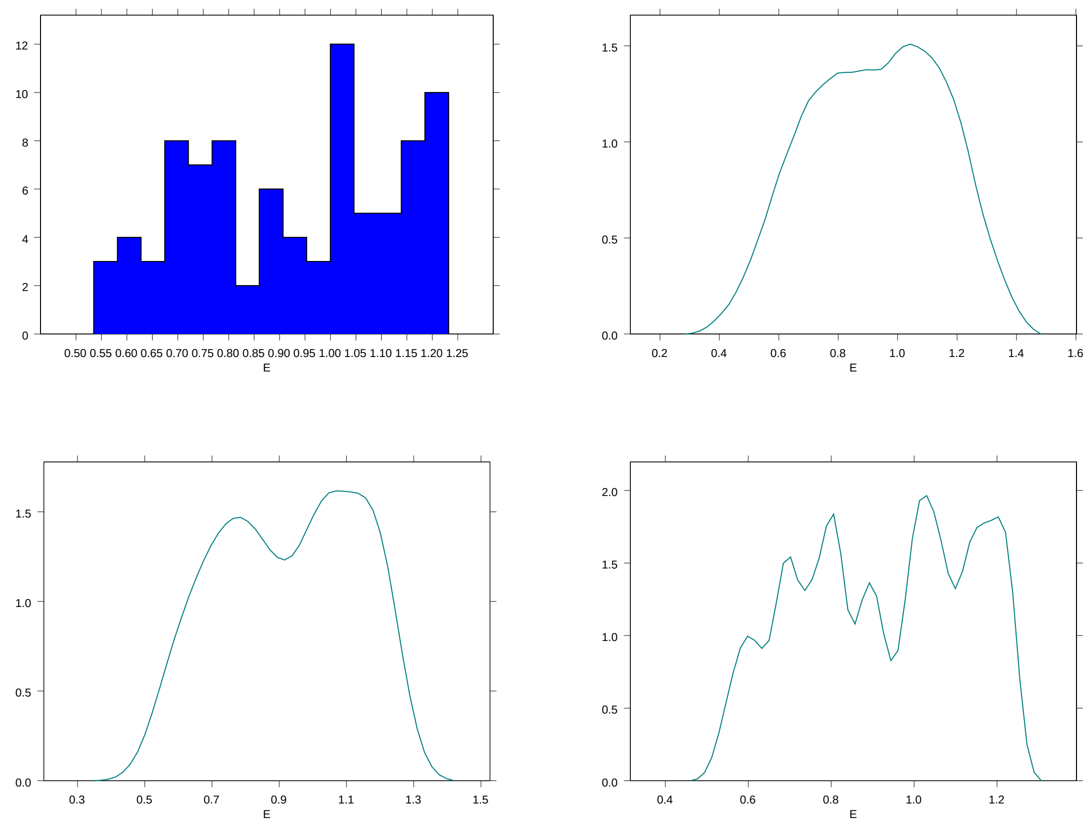

Figure 9.1: Density estimates for the variable ethanol (E), with a histogram (top left) and Guassian kernel estimates with three different bandwidths ($h = 0.5$ (top right), $h = 0.25$ (lower left), and $h = 0.1$ (lower right)).

and $\hat{p}(\mathbf{x})$ is a kernel estimate using a Gaussian kernel with the optimal bandwidth parameter. Thus, we are looking at the relatively "easy" problem of estimating (within 10% relative accuracy) a Gaussian density at the mode of this density (where the points will be most dense on average) by using a Gaussian kernel: what could be easier? Silverman showed (analytically) that the number of data points grows exponentially. In 1 dimension we need 4 points, in 2 we need 19, 3 we need 67, in 6 we need 2790, and by 10 dimensions we need about 842,000. This is an inordinate number of data points for such a simple problem! The lesson to be learned is that density estimation rapidly becomes very difficult as dimensionality increases.

Thus, for practical reasons, kernel density estimation is typically only practical in relatively low-dimensional spaces. In addition, the method is potentially complex to implement for large data sets. Unless the kernel function $K(t)$ has compact support (i.e., unless it is zero outside some finite range on t) then calculating the kernel estimate $p(x)$ at some point $p(x)$ 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 (i.e., 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.

Estimating a good value of h in practice can also be somewhat problematic. It is fair to say that there is no single objective methodology for finding the bandwidth h which 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.

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.

Firstly 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 (i.e., $p_i \approx 1/m^p$) we get that

$$E_{p(\mathbf{x})}[n_i] \approx \frac{n}{m^p}. \tag{9.11}$$

Thus, for example, if $n < 0.5m^p$, the expected number of data points falling in any given cell is closer to 0 than 1. If we use frequency counts (maximum likelihood, 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 or not. Note that if $p(\mathbf{x})$ is non-uniform the problem is actually worse since there will be more cells with smaller p_i (i.e., less chance of any data falling in them). The fundamental problem here is the m^p exponential growth in the number of cells. Note that m^p can grow pretty quickly, e.g., 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 (i.e., the same as it was with n data points). By adding an 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 could 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, e.g., $O(10^12)$ 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

$$p(x_j) = \sum_{X_1,\ldots,X_{j-1},X_{j+1},\ldots,X_p} p(X_1, \ldots, X_{j-1}, x_j, X_{j+1} \ldots, X_p), \tag{9.12}$$

i.e., by "summing out" over all the other variables in the distribution. The sum on the right involves $O(m^{p-1})$ summations, e.g., $O(10^39)$ summations when $p = 40$ and $m = 2$. Clearly this 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. Indeed although our examples above were for categorical data, essentially the same problems arise for ordered or real-valued data. The problem is sometimes referred to as the "curse of dimensionality."

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 $p(\mathbf{x})$, e.g., by *assuming* independence:

$$p(\mathbf{x}) = p(x_1, \ldots, x_p) = \prod_{j=1}^{p} p_j(x_j). \tag{9.13}$$

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.

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, i.e.,

$$p(\mathbf{x}) = \sum_{k=1}^{K} p_k p_k(\mathbf{x}) == \sum_{k=1}^{K} p_k \Big(\prod_{j=1}^{p} p_{k,j}(x_j) \Big). \tag{9.14}$$

This model requires mp probabilities per component, times K components, in addition to the K component weights $p_1, \ldots, p_K$. 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 p_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) which maximize the overall likelihood of the data. This model can be quite useful for modeling large sparse transactional data sets. 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, where 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.

The "mixture of independence models" is also valid for density functions. For example, if each component is Gaussian, this model corresponds to "covering" the points in the space $\mathbf{x}$ with ellipses whose major axes are parallel to the measurement axes (since an independence assumption in a Gaussian model results such ellipses).

A somewhat different way to parsimoniously structure a probability distribution 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

$$p(\mathbf{x}) = \prod_{j=1}^{p} p_j(x_j | pa(X_j)) \tag{9.15}$$

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 the *parent set* for 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 a node X_j is independent of all other variables in the graph which are non-descendants of X_j, given the values of the parents of X_j, $pa(X_j)$.

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, the conditional independence model has a graph where all variables X_j have a single parent C, 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, where the variables are ordered in some manner (e.g., temporally) and each variable X_j depends only on X_{j-1}.

We note in passing that the directed graph formalism cannot capture the structure of all probability distributions: there are other formalisms such as undirected graphical models, log-linear models, and so forth, each with their own independence semantics. Here we have just focused on the directed framework given that it is relatively easy to described and in relatively wide use.

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 (i.e., when directionality of the edges is ignored the graph has no loops) then 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 which 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 here that the parameters of the model are simply the conditional probability tables for variable, $p(x_j|pa(X_j)), 1 \leq j \leq p$, or the parameters for the associated density functions if we are dealing with conditional density functions.

Historically, the structure of a directed graphical model for a given problem was fixed *a priori* based on prior knowledge about the problem. 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 non-hidden case), and as usual care must be taken with initial conditions and detection of convergence.

It is worth emphasizing that if one has strong prior belief that a particular graphical model structure is appropriate for one's 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 recent research interest 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 log-likelihood of the observed data given the model from which is subtracted a penalty term which penalizes the model depending on how many parameters it has (which roughly speaking is proportional to the connectivity of the graph). The BIC score function (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) which 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 non-trivial structure-finding problems in data mining). Thus, iterative local search methods are used: starting with a "prior" structure, adding and deleting edges until no further local improvement is possible. One nice feature from a computational viewpoint is that because the distribution can be expressed in *factored form* (Equation 9.15), the likelihood and penalty terms can also be factored into expressions which are local in terms of the graph structure, e.g., terms which only involve X_j and its parents. Thus, one can calculate the effect of local changes to the model (such as adding or delating an edge) with local computations (since the impact of the change only affects one factor in the score function).

Learning structure with hidden variables is still considered to be somewhat 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.

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*, i.e., decomposing or partitioning a (usually multivariate) data set into groups so that those 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 which 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 size, chest size, and arm length and shirts of one size will be made for each group.

In contrast to this, one might want to see if 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 one might want to see if they fall into distinct classes in terms of these variables. Here one is not partitioning the data for practical convenience, but rather is hoping to discover something about the nature of the sample or the population from which it arose—to discover if the overall population is, in fact, heterogeneous.

Technically, this is what *cluster analysis* seeks to do—to see if 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. Having said that, 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 are a huge number of different algorithms for partitioning data in this way. The important thing is to match one's method with one's objective. This way, mistakes will not arise, whatever one calls the activity.

> **Display 9.3** *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 which might be of interest to them (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 kind 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 which 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 the above 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 if 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 which 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, if 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 the above that cluster analysis (which, for convenience, we are taking as including dissection techniques) hinges 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, have 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 analysis. As far as cluster analysis is 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 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 publications 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, for very few of the methods is a proper understanding of their properties and the way they behave with different kinds of data available.

One of the reasons is that it is difficult to tell if a cluster analysis has been successful. To see this, consider that in predictive modeling we can take a test data set and see how accurately we can predict the value of the target variable on the new data—typically this will provide an honest and absolute assessment of how well our model is performing (e.g., with an accuracy of 85% for a classification problem). For a clustering problem however, 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 formally pose the question whether or not the cluster structure discovered in the training data is genuinely present in new test data. Generally speaking, however, the validity of a clustering is often in the eye of the beholder, e.g., if a cluster produces an interesting scientific insight then we can judge it to be useful. Quantifying this precisely is of course quite difficult since the interpretation of how interesting a clustering is will inevitably be application-dependent and subjective to some degree.

It is very rare indeed that a single application of a technique (in exploratory data analysis or modeling, in general, not merely cluster analysis) leads, by itself and with no other support, to an enlightening discovery about the data. More typically, multiple analyses are needed, looking at the data this way and that, while an interesting structure gradually comes to light. Bearing all this in mind, some caution should be exercised when using algorithms such as clustering and the results examined with care, rather than being taken at face value.

As we shall see in the next few sections, different methods of cluster analysis are effective at detecting different *kinds* of clusters, and one should consider this when choosing a particular algorithm. That is, one should consider what it is one means by a "cluster." In effect, different clustering algorithms will be biased towards 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, one 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 which seeks to partition the data so as to minimize this maximum interpoint distance (more on this below). One would clearly expect such a method to produce compact, roughly spherical, clusters. On the other hand, one 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 could have been measured at different stages of some evolutionary process. For example, in a cluster analysis of people suffering from some illness, to see if there were different subtypes, one 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 learnt from this is that one must match the method to the objectives. In particular, one must adopt a cluster analytic tool which is effective at detecting clusters which conform to the definition of what means by "cluster" in the problem at hand. Having said that, it is perhaps worth adding that one should not be too rigid about it. Data mining, after all, is about discovering the *unexpected*, so one must not be too determined in imposing one's preconceptions on the analysis. Perhaps a search for a different kind of cluster structure will throw up things one had 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, i.e., 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. Typically, the *centroid* or *average* of the points belonging to a cluster is considered to be the "representation" for that cluster. There is often no explicit representation for the shape of a cluster or the shapes of the boundaries between clusters. For cluster representations based on the notion of a single "center" for each cluster, the boundaries between clusters will be implicitly defined, e.g., if a point $\mathbf{x}$ is assigned to a cluster based on 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 are often used to optimize the score function given a data set.

Note that in partition-based clustering, the number of clusters K is typically defined ahead of time. One can of course vary K to study the effect of different numbers of clusters. Later we will discuss other clustering techniques (such as probabilistic clustering) which provide a more natural framework for considering different values of K.

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 have 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 $\mathbf{x}(i) \in C_k$ which 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 mean (average) of the points in the cluster C_k as the cluster center. The cluster center $\mathbf{r}_k$ for C_k is defined as the average (mean) of the points belonging to C_k, i.e.,

$$\mathbf{r}_k = \frac{1}{n_k} \sum_{\mathbf{x} \in C_k} \mathbf{x}, \tag{9.16}$$

where n_k is 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:

$$wc(\mathcal{C}) = \sum_{k=1}^{K} wc(C_k) = \sum_{k=1}^{K} \sum_{\mathbf{x}(i) \in C_k} d\left(\mathbf{x}, \mathbf{r}_k\right)^2. \tag{9.17}$$

For the case where $d(\mathbf{x}, \mathbf{r}_k)$ 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:

$$bc(\mathcal{C}) = \sum_{1 \le j < k \le K} d\left(\mathbf{r}_j, \mathbf{r}_k\right)^2. \tag{9.18}$$

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 $bc(\mathcal{C})$.

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 $\mathcal{C}$ which minimizes the within cluster variation of Equation 9.17, for measurements $\mathbf{x}$ in a Euclidean space $\mathcal{R}^p$.

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:

$$wc(C_k) = \max_i \min_{\mathbf{y}(j) \in C_k} \{d\left(\mathbf{x}(i), \mathbf{y}(j)\right) \mid x(i) \in C_k, x \neq y\}. \tag{9.19}$$

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.

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.

We can use the notion of covariance to develop more general score functions for clusterings $\mathcal{C}$ in a Euclidean space. Recall that our p-dimensional vector of measurements is a column vector, i.e., $\mathbf{x} = (x_1, \ldots, x_p)^T$. For points within a particular cluster C_k, we can define a $p \times p$ matrix

$$\mathbf{W}_k = \sum_{\mathbf{x} \in C_k} (\mathbf{x} - \mathbf{r}_k)(\mathbf{x} - \mathbf{r})^T \tag{9.20}$$

which is an (unnormalized) *covariance matrix* for the points in cluster C_k (recall the discussion on covariance matrices of Chapter XXXX). Note that the within-cluster sum-of-squares for a particular cluster can be defined as the trace (sum of diagonal elements) of this matrix, $tr(\mathbf{W}_k)$, and thus, the total within-cluster sum-of-squares of Equation 9.17 can be expressed as

$$wc(\mathcal{C}) = \sum_k tr(\mathbf{W}_k). \tag{9.21}$$

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

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

$$\mathbf{B} = \sum_{k=1}^{c} n_k(\mathbf{r}_k - \hat{\mu})(\mathbf{r}_k - \hat{\mu})^T. \tag{9.22}$$

where $\hat{\mu}$ is the estimated global mean of all data points in D. This is a $p \times p$ matrix which 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 which emphasizes a "larger" $\mathbf{B}$ will tend to encourage the cluster means to be more separated.

An important point here (which is often overlooked) is 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 (e.g., different combinations of $\mathbf{W}$ and $\mathbf{B}$) can express significantly different preferences in terms of cluster structure. In effect these are *biases*: it is important in practical applications to consider if the bias of a particular score function is in fact appropriate for a given problem. For example, as mentioned below, the use of $tr(\mathbf{W})$ leads to a favoring of relatively spherical equal-sized clusters; for certain applications this may be appropriate, for others not so appropriate.

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. (See the example in Chapter 2.XXX where we compared the relative distances between three objects measured in terms of weight and length which resulted when different units were used.) 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. 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 (where, for example, 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 that it also detects elliptic structures as clusters, but does also favor equal sized clusters. Adjustments which 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 one's data are thought to be generated in that way, one might contemplate fitting a formal mixture model, as outlined in later in Section 9.2.4.)

The $tr(\mathbf{B}\mathbf{W}^{-1})$ 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 $\mathbf{B}\mathbf{W}^{-1}$ it will place most emphasis on the largest eigenvalue and hence have tendency to yield collinear clusters.

The property that the clusters obtained from using the above score functions tend to have similar shape is not attractive in all situations (indeed, it is probably a rare situation in which it is 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 similar 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, which 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 the above 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 that which 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 the simple distances. 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 class 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, i.e., each object has given rise to a binary vector of the form $(0011\ldots1)$, and suppose we have a proposed grouping into clusters. Then, for each group we can define a binary vector which 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 how many of the variables have values which 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 that which 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. One 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 one wants. Of course, one can rerun the analysis several times, with different numbers of clusters, but this still requires one to be able to choose between competing numbers. There is no "best" solution to this problem. One can, of course, examine how the clustering score function changes as one increases 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 errors for example 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 will be apparent from the above that cluster analysis is very much a data-driven tool, with relatively little formal model-building underlying it. Some researchers have attempted to put it on a sounder model-based footing. For example, one can supplement the procedures by assuming that there is also a random process generating sparsely distributed points uniformly across the whole space. This makes the methods less susceptible to outliers. A further assumption is to parametrically model the distribution of the data within each cluster using specific distributional assumptions—we will return to this in our discussion of probabilistic model-based clustering in Section 9.2.4.

9.4.2 Basic Algorithms for Partition-Based Clustering

We saw in the previous section that there exist a large variety of score functions that 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. One simply searches through the space of possible assignments $\mathcal{C}$ of points to clusters to find that which 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 which maximizes (or minimizes) our chosen score function. A little calculation shows the number of possible allocations is

$$\frac{1}{K!}\sum_{i=0}^{K}(-1)^i \left(\begin{array}{c} K \\ i \end{array} \right) (K-i)^n, \tag{9.23}$$

so that, for example, there are some 10^{30} possible allocations of 100 objects into 2 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 which 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, i.e., there is usually no direct method for finding a specific clustering $\mathcal{C}$ which optimizes the score function. Thus, since closed form solutions do not exist and exhaustive search is infeasible, one must resort to some form of systematic *search* of the space of possible clusters. 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 which 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, to then reassign points so as to give the greatest increase (or decrease) in the score function, to then 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 one does not know how good the clustering $\mathcal{C}$ which 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 (and which has close connection to the EM algorithm discussed in Chapter 8 and is mentioned again later in Section 9.2.4 of this chapter). The number K of clusters is fixed before running the algorithm (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, 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.

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 $j = 1, \ldots, K, j \neq k\}$;
  od;
  compute new cluster centers:
  for $k = 1, \ldots, K$ **do**
    $\mathbf{r}_k$ = the vector mean of the points in C_k
  od;
od;

Display 9.4 *Electro-mechanical 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 antenna control systems are quite sensitive to subtle changes in operating behavior and can be used for online health monitoring and fault detection. Figure 9.2 shows sample data from a 34m Deep Space Network antenna. Each bivariate data point corresponds to a 2-second window of motor-current measurements, which have been modeled by a simple autoregressive (linear) time-series model. 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 9.2 show which data points belong to which condition (3 groups, one normal and two fault conditions). Figure 9.3 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 (i.e., using the data in the upper plot of Figure 9.2 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 4 iterations (Figure 9.3) the algorithm quickly converges to a clustering (the trajectory of the cluster means are plotted in Figure 9.4. The final clustering after the fourth iteration (lower plot of Figure 9.3) produces three groups which very closely match the known grouping shown in Figure 9.2. For this data the grouping is relatively obvious of course in that the different 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 which 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, one can simply add in each data point, without the recycling. Further extensions (e.g. the ISODATA algorithm) include splitting and/or

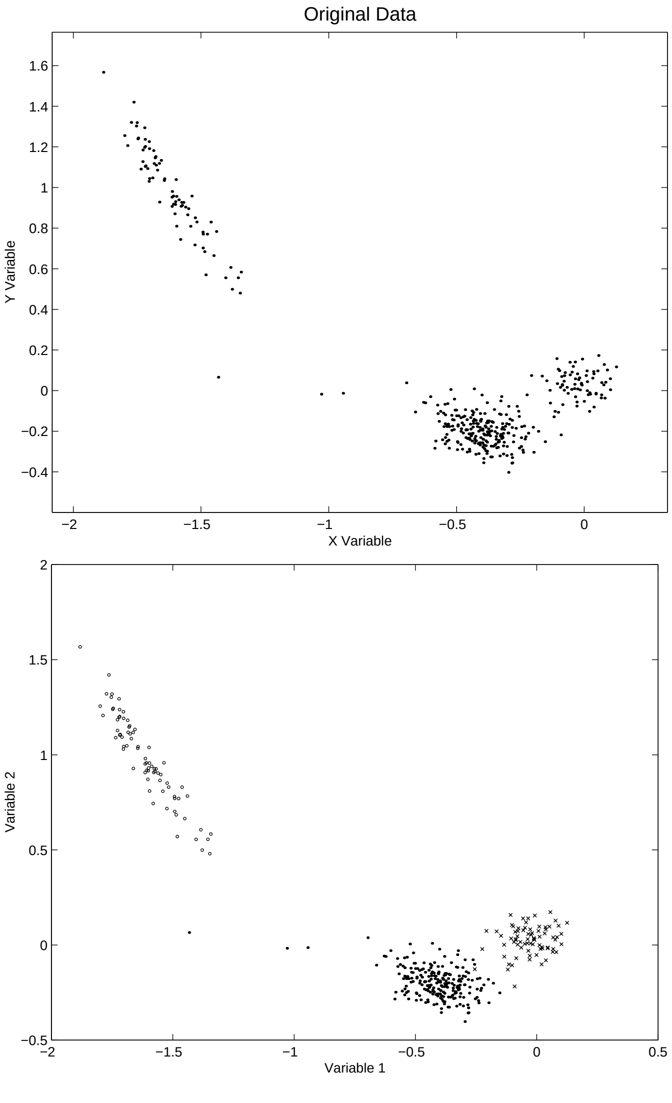

Figure 9.2: Antenna data. On top the data points are shown without class labels, and on the bottom different symbols are used for the three known classes (dots are normal, circles are tachometer noise, and x's are integrator short circuit.)

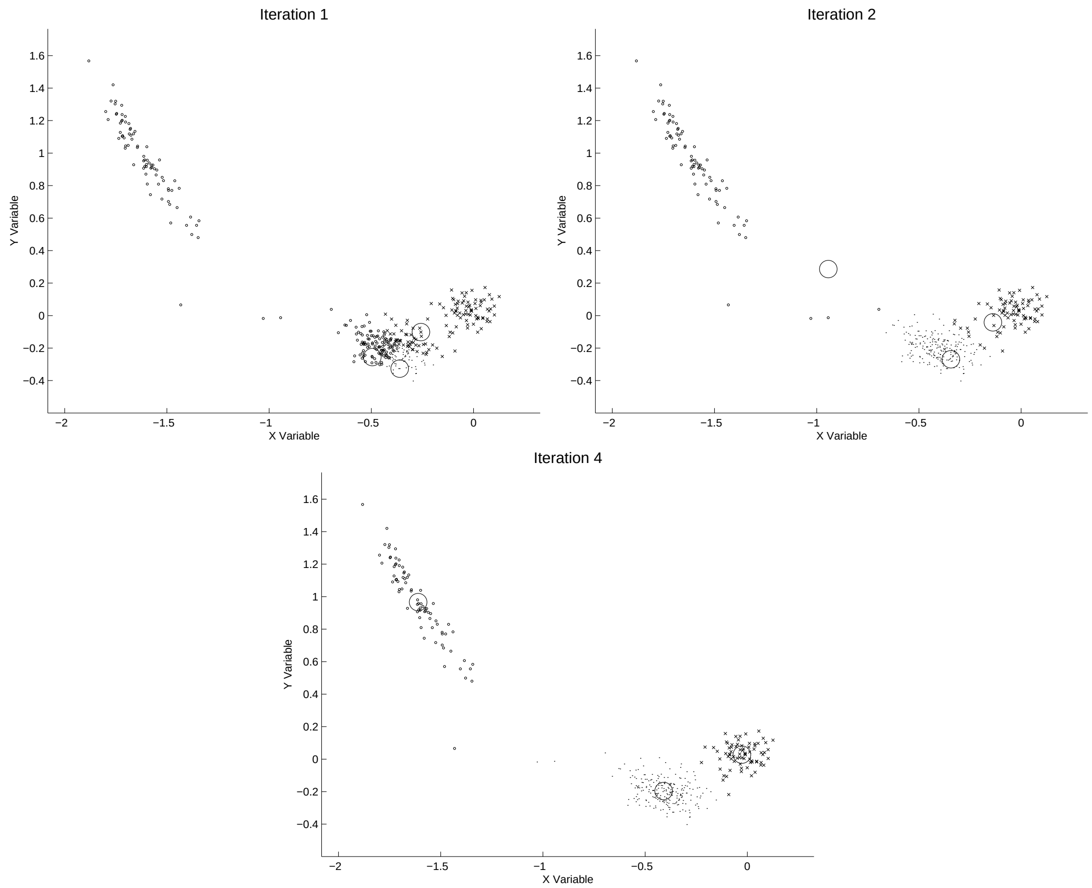

Figure 9.3: Example of running the K-means algorithm on the two-dimensional antenna data. The plots show the locations of the means of the clusters (large circles) at various iterations of the K-means algorithm, as well as the classification of the data points at each iteration according to the closest mean (dots, circles, and x's for each of the 3 clusters).

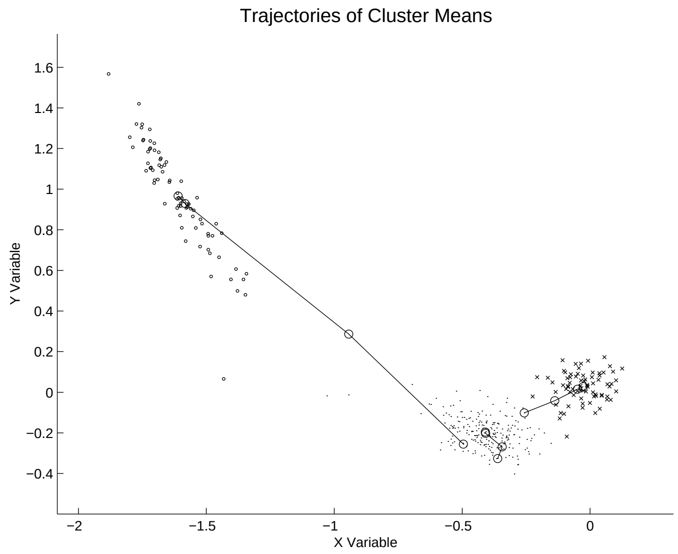

Figure 9.4: A summary of the trajectories of the 3 cluster means during the K-means iterations of Figure 9.3.

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 formulae 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. One 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 that which optimizes a specified score function, it will come as no surprise to learn that various kinds of mathematical programming methods have been applied to this problem. These include linear programming, dynamic programming, and linear and non-linear 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, one can try to compress the data set by replacing groups of objects by succint 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 which 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 method: the *agglomerative* (which merge) and the *divisive* (which divide). We shall deal with each in turn. 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 (see Chapters 6 and 11).

A notable feature of hierarchical clustering is that it is difficult to separate out the model, the score function, and the search method used to determine the best clustering. Thus, in this section we will focus on clustering algorithms directly, since the breakdown into models, score functions, and search methods is not particularly easy to do here. One can consider the final hierarchy to be a model, as a hierarchical mapping of data points to clusters, however, the nature of this model (i.e., the cluster "shapes") is somewhat 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 (i.e., 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) or 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 of most use of course when there are more than two variables. If there are only two, then one can eyeball a scatterplot, and look for structure. However, to illustrate the ideas on a data set where we can see what is going on, we 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 9.5 shows a scatterplot of the 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 dendrogram in this exposition, and have no other substantive significance.

As an example, Figure 9.6 shows the dendrogram which results from agglomerative merging the two clusters which leads to the smallest increase in within cluster sum of squares. The height of the crossbars in the dendrogram (where branches merge) shows value of this criterion. Thus, initially, the smallest increase is obtained by merging points 11 and 33, and from Figure 9.5 we can see that these are indeed very close (in fact, the closest). The next merger comes from merging points 2 and 32. After a few more mergers of individual pairs of

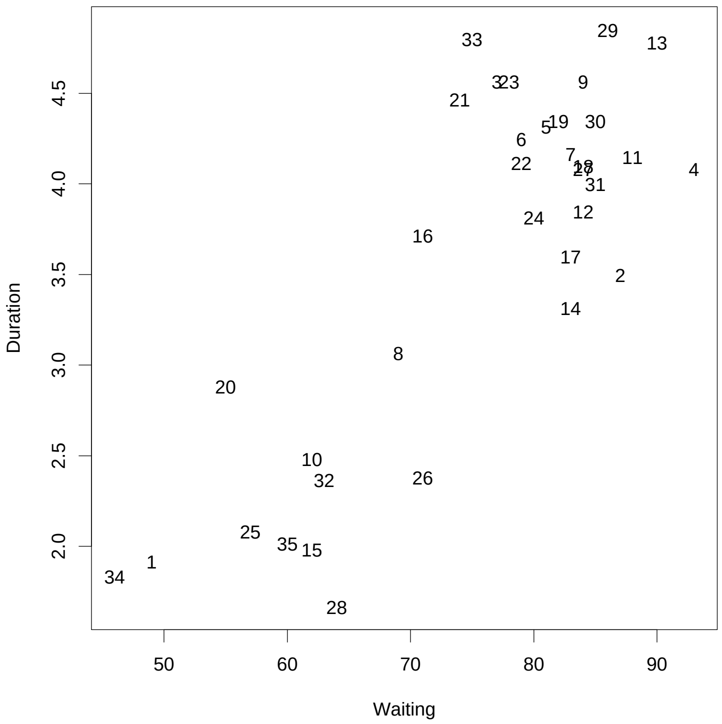

Figure 9.5: Duration of eruptions versus waiting-time between durations (in minutes) for the Old Faithful geyser in Yellowstone Park.

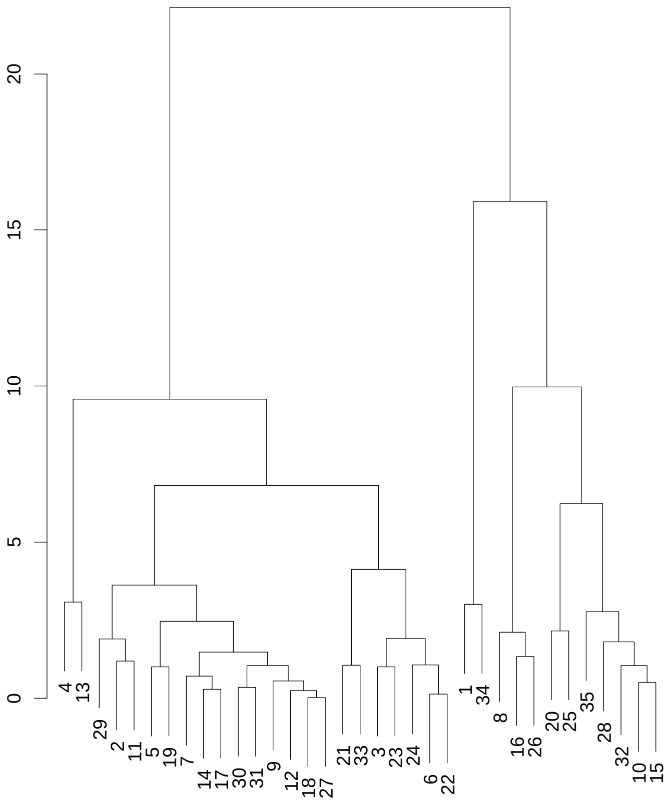

Figure 9.6: Dendrogram resulting from clustering of data in Figure 9.5 using the criterion of merging clusters which leads to the smallest increase in the total sum of squared errors.

neighboring points, point 31 is merged with the cluster consisting of the two points 8 and 15, this being the merger which 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 one 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 which 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 have n data points $D = \{\mathbf{x}(1), \ldots, \mathbf{x}(n)\}$, 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 $i = 1, \ldots, n$ let $C_i = \{\mathbf{x}(i)\}$;
while there are 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;
od;

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 one needs distances between individual data objects to begin the clustering, and during clustering one needs to be able to compute distances between groups of data points (i.e., distances between clusters). Thus, one advantage of this approach (over partition-based clustering for example) is the fact that we need not 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 which 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 (i.e., 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 (i.e., 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 (i.e., 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;

$$\mathcal{D}_{sl}(C_i, C_j) = \min_{\mathbf{x},\mathbf{y}}\{d(\mathbf{x},\mathbf{y}) \mid \mathbf{x} \in C_i, \mathbf{y} \in C_j\} \tag{9.24}$$

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 one's 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 discussed in Section 9XXX). 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 one is 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 9.5 is shown in Figure 9.7. Note that although the initial merges are the same as those in Figure 9.6, the two methods soon start to differ, and the final high level structure is quite different. Note in particular, that the final merge of the single link method is to combine a single point (number 30) with the cluster consisting of all of the other points.

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:

$$\mathcal{D}_{fl}(C_i, C_j) = \max_{\mathbf{x},\mathbf{y}}\{d(\mathbf{x},\mathbf{y}) \mid \mathbf{x} \in C_i, \mathbf{y} \in C_j\}, \tag{9.25}$$

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

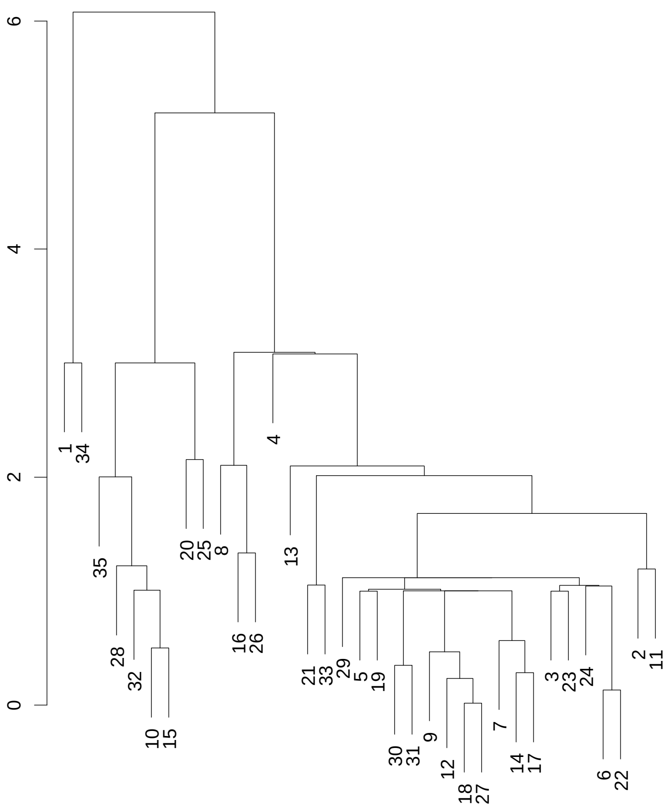

Figure 9.7: Dendrogram of the single link method applied to the data in Figure 9.5.

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—see the $tr(\mathbf{W})$ score discussed in Section 9XXX). Each such measure has slightly different properties.

Other variants also exist—the median measure for vector data ignores the size of clusters, taking the "center" of a combination of two clusters to be the mid-point of the line joining the centers of the two components. Since one is seeking the novel in data mining, it may well be worthwhile experimenting with several measures, in case one throws up something unusual and interesting.

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, it 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 which 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 inter-cluster 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 selected for transfer from a main cluster to a subcluster which 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 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

$$f(\mathbf{x}) = \sum_{k=1}^{K} p_k f_k(\mathbf{x}; \theta_k), \tag{9.26}$$

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 one wants to fit to the data, choose parametric models for each of these K clusters (e.g., multivariate normals are a common choice when appropriate), and then use the EM algorithm as described in the last section to determine the component parameters θ_k and component probabilities p_k from the data. (One 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.

In essence this is exactly the same approach as the standard mixture modeling framework of the last section except that the goal here is to identify and interpret the fitted components as clusters, i.e., the overall goal is cluster identification rather than just density estimation (in density estimate, using a mixture model, we need not necessarily interpret the model, i.e., it can be used as a "black box" for approximating an unknown density). As mentioned already, great caution is urged in the interpretation of fitted mixture components; the use of prior first principles knowledge of the data generating mechanism can be invaluable in separating artifacts of the sampling and modeling process from true discoveries. Below we illustrate an application of the method to a data set where the true class labels are in fact known but are removed and then "discovered" by the algorithm.

Display 9.5 *Individuals with chronic iron deficiency anemia tend to produce red blood cells of lower volume and lower hemoglobin concentration than normal. A blood sample can be taken to determine a person's mean red blood cell volume and hemoglobin concentration. The upper left corner of Figure 9.8 shows a scatter plot of the bivariate mean volume and hemoglobin concentrations for 182 individuals with labels determined by a separate lab test. A normal mixture model with $K = 2$ was fit to these individuals, with the labels removed. The results are shown in Figure 9.9, illustrating that a 2-component normal mixture appears to capture the main features of the data and would provides an excellent clustering if the group labels were unknown (i.e., if the lab test had not been performed). Figure 9.8 verifies that the likelihood (or equivalently, log-likelihood) is non-decreasing as a function of iteration number.*

The red blood cell example of Figure 9.9 illustrates several advantages 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

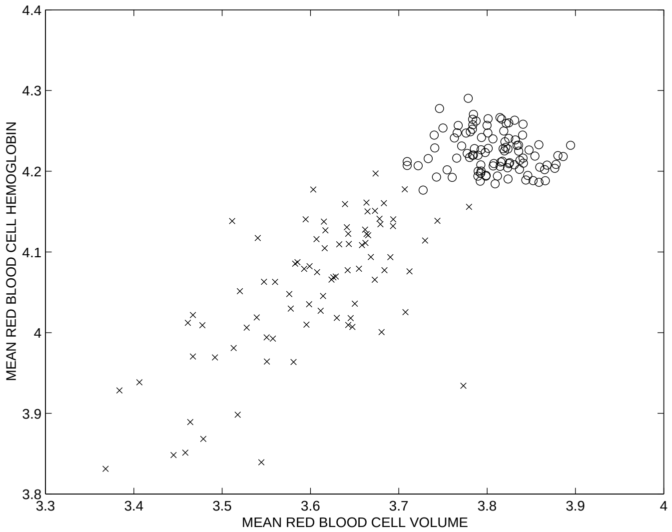

Figure 9.8: Red-blood cell measurements (mean volume and mean hemoglobin concentration) from 182 individuals showing the separation of the individuals into two groups; healthy (circles) and iron deficient anemia (crosses).

certainly agrees with our common sense intuition, and it is the type of information which 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 probability vector of cluster memberships, i.e., the probability that data point $(\mathbf{i})$ originated from component k (recall Equation **??**), $1 \leq i \leq n, 1 \leq k \leq K$. 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, i.e., there is uncertainty about which group they belong to. Again, from the viewpoint of exploring the data, such data points may be very valuable and worthy of detection and closer study (e.g., individuals who are 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 which 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 (e.g., a uniform density) to pick up outliers and background points which do not appear to

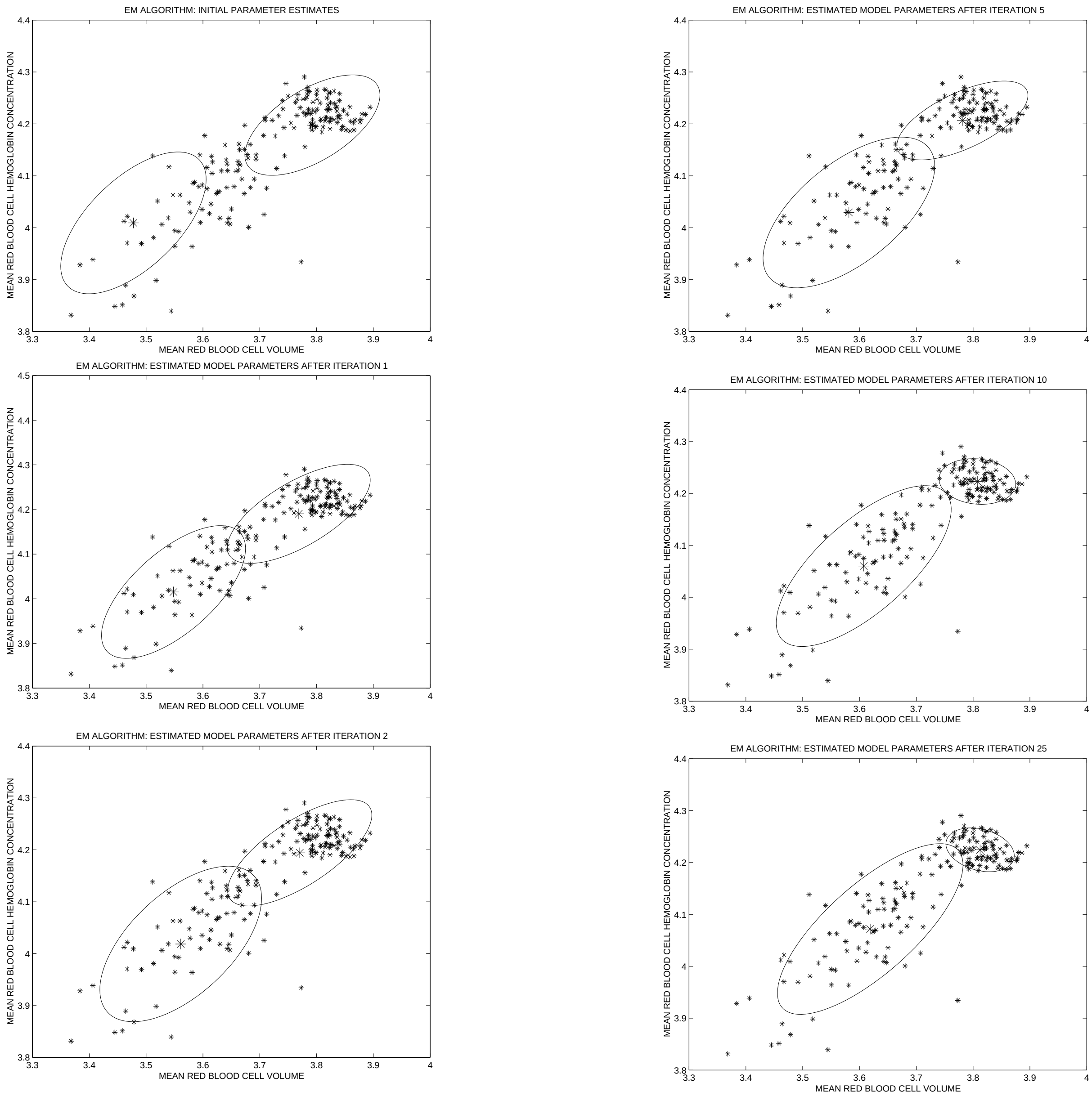

Figure 9.9: Example of running the EM algorithm on the red blood cell measurements of Figure 9.8. The plots (running top to bottom, left first, then right) show the 3σ covariance ellipses and means of the fitted components at various stages of the EM algorithm.

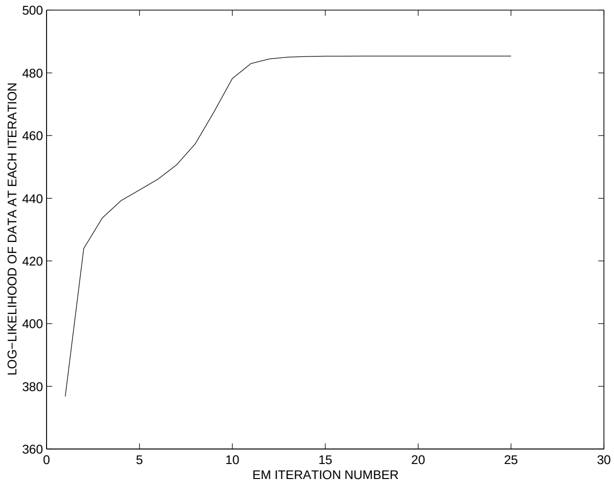

Figure 9.10: The log-likelihood of the red-blood cell data in the example of Figure 9.9, for each iteration of EM.

belong to any of the other K components; the relative weight p_{K+1} of this background component can be learned by EM directly from the data.

- The method can be extended to data which is not in p-dimensional vector form, e.g., one can cluster sequences using mixtures of probabilistic sequence models (e.g., mixtures of Markov models), cluster curves using mixtures of regression models, and so forth, all within the same general EM framework.

Having said all of this, it must of course be pointed out that 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 one has reason to believe that the distributional forms are appropriate. For our red blood cell data above 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. One can think of K-means as a rough approximation to EM with Gaussian mixture components, where in K-means the memberships are "quantized" so that each data point is assigned to the closest cluster only (no probabilities) and the covariance matrices for each cluster are in effect assumed to be common across all clusters and diagonal with the same variance in each direction (corresponding to the Euclidean distance assumption). Despite the relative simplicity of the underlying "models," K-means and many other partition-based clustering algorithms of Section 9.4 have the virtue of being relatively simple to implement and relatively non-parametric in their assumptions. One can think of the probabilistic model-based

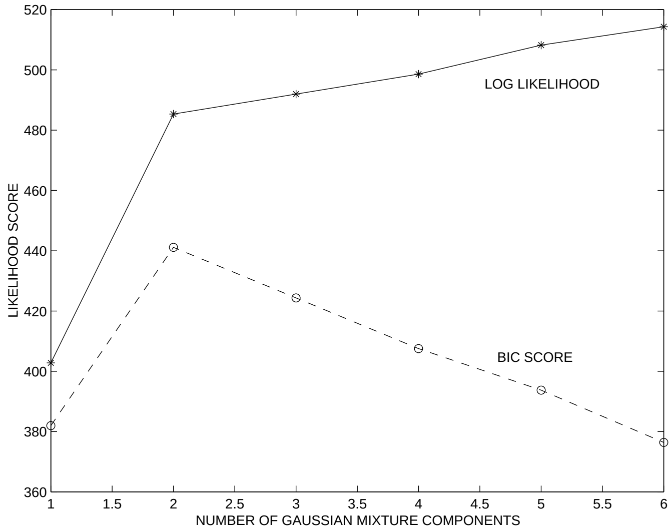

Figure 9.11: Log-likelihood and BIC score as a function of the number of Gaussian components fitted to the red blood cell data of Figure9.9.

approach as being somewhat more powerful; thus, as with any "power tool" the user must exercise care to use it wisely!

Display 9.6 *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 are independent (as discussed in Section 9.2.7), i.e., that we can write*

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

To specify a model for the data we just need to specify the probability of observing value 1 for the kth component. Denote this probability by θ_{ki}. Then we can write the component densities as

$$p_k(x_j, \theta_{kj}) = \theta_{kj}^{x_j}(1 - \theta_{kj})^{1-x_j}$$

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:

$$p(\mathbf{x}(i)) = \sum_{k=1}^{K} p_k \prod_j \theta_{kj}^{x_j(i)}(1 - \theta_{kj})^{1-x_j(i)}$$

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:

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

where $p(\mathbf{x}(i))$ is as defined in Equation 9.6. Calculation of $p(k|i)$ takes $O(nK)$ steps and the same amount of memory, since it must be done 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:

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

. Thus, the 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 in total we get $O(IKnp)$ as the basic complexity, which can be thought of as IK times the size of the data matrix.

For really large data sets which reside on disk, however, doing IK 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, i.e., 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 (the usual story of overfitting as one adds more complexity to the model). Thus, likelihood alone cannot tell us directly about which of the models as a function of K are closest to the true data generating process. This is an age-old problem in statistics. The usual approach of hypothesis testing (e.g., 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, where

$$BIC(K) = l(\theta_K) - \frac{p_K}{2}\log n \tag{9.29}$$

where $l(\theta_K)$ is the maximizing value of the log-likelihood and p_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 maximum taken as the most likely value of K. The original derivation of BIC was based on asymptotic arguments in a different (regression) context, arguments which 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 9.11 the BIC score function is plotted for the red-blood cell data and points to $K = 2$ as the best model (recall that we have independent 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, such as AIC (Akaike's information criterion) and MDL (minimum description length), but BIC is the most widely used.

Resampling Techniques: One 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, e.g., 10 times more for the application of 10-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 $p(K|D)$, i.e., 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 p_K-dimensional space) so various approximations are sought. Both analytic approximations (e.g., the Laplace approximation about the mode of the posterior distribution) and sampling techniques (such as Markov chain Monte Carlo) are used. Indeed, one can argue that the BIC method and the cross-validation method can be viewed as special cases of a more general Bayesian approach.

Earlier in this Chapter we briefly noted the difference between mixture decomposition, segmentation, and cluster analysis. Here, however, we want to draw attention to one important distinction which can characterize the difference between mixture models and cluster analysis. The aim of cluster analysis is to divide the data into naturally occurring regions in which the points are closely or densely clustered, with relatively sparse regions between them. 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. `Is the point here that a two-component mixture can be appropriate even when the data is unimodal and ''looks'' like coming from a single component?` Clearly, if the means are equal, then this will be unimodal. More interestingly, a sufficient condition for the mixture to be unimodal (for all values of the mixing proportions) when the means are different is

$$| \mu_1 - \mu_2 \ | \leq 2 \min(\sigma_1, \sigma_2). \tag{9.30}$$

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.

9.7 Further Reading

There are now many books on cluster analysis. Recommended ones include Anderberg (1973), Späth (1985), and Kaufman and Rousseeuw (1990). The distinction between dissection and finding natural partitions is not always appreciated, and yet it can be an important one and it should not be ignored. Examples of authors who have made the distinction include Kendall (1980), Gordon (1981), and Spath (1985). Banfield and Raftery (1993) proposed the idea of adding a 'cluster' which pervades the whole space by superimposing a separate Poisson process which 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. Marriott (1971) showed that the criterion was asymptotically constant for optimal partitions of a multivariate uniform distribution. Krzanowski and Marriott (1995), Table 10.6, gives a list of updating formulae for clustering criteria based on 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 poin" of each cluster is an element of that cluster, rather than being the centroid of the elements. A review of early work on mathematical programming methods applied in cluster analysis is given by Rao (1971). For a cluster analytic study of whiskies, see Lapointe and Legendre (1994). 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, which 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 described in Section 8a3.2.2 is due to MacNaughton-Smith *et al* (1964). The fuzzy clustering criterion described in Section 8a.3 is due to Bezdek (1974) and Dunn (1974), and a review of fuzzy clustering algorithms is given in Bezdek (1987). The overlapping cluster methods summarized in Section 8a.3 is due to Shepard and Arabie (1979). Scaling up clustering methods is discussed in, e.g., Ganti et al. (1999).

Books on mixture distributions, including applications to clustering, include Everitt and Hand (1981), Titterington, Smith, and Makov (1985), McLachlan and Basford (1988). Diebolt and Robert (1994) provides an example of the general Bayesian approach to mixture modeling. Mixture models in haematology are described in McLaren (1996). 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).

Banfield and Raftery (1993) is considered a seminal paper in probabilistic model-based clustering, with subsequent developments in Celeux and Govaert (1995). More recent overviews and descriptions of algorithm implementations are in Fraley and Raftery (1998) and McLachlan and Peel (1998). The application of mixture models to clustering sequences is described in Smyth (1997) and Gaffney and Smyth (1999) illustrate the use of mixtures of regression models to clustering curves and trajectory data. Jordan and Jacobs (1994) provide a generalization of standard mixtures to a mixture-based architecture called "mixtures of experts" which provides a general 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 (1996) describes the 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 K.

Cheng and Wallace (1993) describe a fascinating 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.

`Add mention of other applications such as clustering Web pages, clustering documents, clustering protein sequences, clustering images (eigenvalue methods).`

A formal description of path analysis is given in Wermuth (1980). Recent books on factor analysis include Bartholomew (1987), Reyment and Jöreskog (1993), and Basilevsky (1994). Bollen (1989) is a recommended book on structural equation models. Books on graphical models include those by Whittaker (1990), Edwards (1995), Cox and Wermuth (1996), and Lauritzen (1996). The bank loan model described in Display 8a.10XXX is developed in more detail in Hand, McConway, and Stanghellini (1997).

Possible add here more recent ideas on large data sets

Buhmann and Hofmann for speeding up hierarchical clustering?

Fayyad and Mangasarian for linear programming methods?

References

Anderberg M.R. (1973) *Cluster Analysis for Applications.* New York: Academic Press.

Azzalini A. and Bowman A.W. (1990) A look at some data on the Old Faithful geyser. *Applied Statistics*, **39**, 357-365.

Banfield J.D. and Raftery A.E. (1993) Model-based Gaussian and non-Gaussian clustering. *Biometrics*, **49**, 803-821.

Bartholomew D.J. (1987) *Latent Variable Models and Factor Analysis*. London: Charles Griffin and Co.

Basilevsky A. (1994) *Statistical Factor Analysis and Related Methods*. New York: Wiley.

Bezdek J.C. (1974) Numerical taxonomy with fuzzy sets. *Journal of Mathematical Biology*, **1**, 57-71

Bezdek J.C. (1987) Some non-standard clustering algorithms. In *Developments in Numerical Ecology*, ed. P. Legendre and L. Legendre, Berlin: Springer-Verlag, 225-287.

Bollen K.A. (1989) *Structural Equations with Latent Variables*. New York: Wiley.

Bradley, P. S., Fayyad, U., Reina, C. (1998) Scaling clustering algorithms to large databases. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, Agrawal, R., Stolorz, P., and Piatetsky-Shapiro, G. (eds.), Menlo Park, CA: AAAI Press, 9–15.

Celeux, G. and Govaert, G. (1995) Gaussian parsimonious clustering models. *Pattern Recognition*, 28, 781–793.

Cheeseman, P. and Stutz. J. (1996) Bayesian classification (AutoClass): theory and results. In *Advances in Knowledge Discovery and Data Mining*, U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (eds.), Cambridge, MA: AAAI/MIT Press, pp. 153–180.

Cheng, X., and Wallace, J. M. (1993) Cluster analysis of the Northern hemisphere wintertime 500-hPa height field: spatial patterns. *J. Atmos. Sci.*, 50(16), 2674–2696.

Chickering, D. M., and Heckerman, D. (1997) Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, 29(2/3), 181–244.

Cox D.R. and Wermuth N. (1996) *Multivariate Dependencies: Models, Analysis, and Interpretation*. London: Chapman and Hall.

Diebolt, J. and Robert, C. P. (1994) Bayesian estimation of finite mixture distributions. *Journal of the Royal Statistical Society B*, 56, 363–375.

Dunn J.C. (1974) A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, **3**, 32-57.

Edwards D. (1995) *Introduction to Graphical Modelling*. New York: Springer Verlag.

Everitt B.S. (1981) A Monte Carlo investigation of the likelihood ratio test for the number of components in a mixture of normal distributions. *Multivariate Behavioural Research*, **16**, 171-180.

Everitt B.S. and Hand D.J. (1981) *Finite Mixture Distributions.* London: Chapman and Hall.

Feng, Z. D., and McCulloch, C. E. (1996) Using bootstrap likelihood ratios in finite mixture models. *Journal of the Royal Statistical Society B*, 58(3), 609–617.

Florek K., Lukasziwicz J., Perkal J., Steinhaus H., and Zubrzycki S. (1951) Sur la liaison et la division des points d'un ensemble fini. *Colloquium Mathematicum*, **2**, 282-285.

Fraley, C. and Raftery, A. E. (1998) How many clusters? Which clustering method? Answers via Model-based cluster analysis. *Computer Journal*, 41, 578–588.

Gaffney, S. and Smyth P. (1999) Trajectory clustering with mixtures of regression models. In *Proceedings of the 1999 ACM Conference on Knowledge Discovery and Data Mining.*

Ganti, V., Ramakrishnan, R., and Gehrke, J. (1999) Clustering large datasets in arbitrary metric spaces. *Proceedings of the ???th International Conference on Data Engineering (ICD'99)*, pp. ???-???.

Gordon A. (1981) *Classification: Methods for the Exploratory Analysis of Multivariate Data.* London: Chapman and Hall.

Goer J.C. (1967) A comparison of some methods of cluster analysis. *Biometrics*, **23**, 623-628.

Gower J.C. (1974) Maximal predictive classification. *Biometrics*, **30**, 643-654.

Hall D.J. and Ball G.B. (1965) ISODATA: a novel method of cluster analysis and pattern classification. Technical Report, Stanford Research Institute, Menlo Park, California.

Hand D.J. (1981) *Discrimination and Classification.* Chichester: Wiley.

Hand D.J., McConway K.J., and Stanghellini E. (1997) Graphical models of applicants for credit. *IMA Journal of Mathematics Applied in Business and Industry*, **8**, 143-155.

Jordan, M. I. and Jacobs, R. A. (1994) Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6, 181-214.

Kass, R. E. and Raftery, A. E. (1995) Bayes factors. *Journal of the American Statistical Association*, 90, 773–795.

Kaufman L. and Rousseeuw P.J. (1990) *Finding Groups in Data: An Introduction to Cluster Analysis.* New York: Wiley.

Kendall M.G. (1980) *Multivariate Analysis.* (2nd ed.) London: Griffin.

Koontz W.L.G., Narendra P.M., and Fukunaga K. (1975) A branch and bound clustering algorithm. *IEEE Transactions on Computers*, **24**, 908-915.

Krzanowski W.J. and Marriott F.H.C. (1995) *Multivariate Analysis vol.2: Classification, Covariance Structures, and Repeated Measurements.* London: Arnold.

Lambert J.M. and Williams W.T. (1966) Multivariate methods in plant ecology IV: comparison of information analysis and association analysis. *Journal of Ecology*, **54**, 635-664.

Lance G.N. and Williams W.T. (1967) A general theory of classificatory sorting strategies: 1. Hierarchical systems. *Computer Journal*, **9**, 373-380.

Lapointe F.-J. and Legendre P. (1994) A classification of pure malt Scotch whiskies. *Applied Statistics*, **43**, 237-257.

Lauritzen S.L. (1996) *Graphical Models.* Oxford: Clarendon Press.

McLachlan G.J. (1987) On bootstrapping the likelihood ratio test for the number of components in a normal mixture. *Applied Statistics*, **36**, 318-324.

McLachlan G.J. and Basford K.E. (1988) *Mixture Models: Inference and Applications to Clustering.* New York: Marcel Dekker.

McLachlan, G. J. and D. Peel (1997) 'On a resampling approach to choosing the number of components in normal mixture models,' in *Computing Science and Statistics (Vol. 28)*, L. Billard and N.I. Fisher (Eds.). Fairfax Station, Virginia: Interface Foundation of North America, pp. 260-266.

McLachlan, G. J. and D. Peel (1998) 'MIXFIT: An algorithm for the automatic fitting and testing of normal mixture models,' *Proceedings of the 14th International Conference on Pattern Recognition*, Vol. I, Los Alamitos, CA: IEEE Computer Society, pp. 553-557, 1998.

McLaren C.E. (1996) Mixture models in haematology: a series of case studies. *Statistical Methods in Medical Research*, **5**, 129-153.

MacNaughton-Smith P., Williams W.T., Dale M.B., and Mockett L.G. (1964) Dissimilarity analysis. *Nature*, **202**, 1034-1035.

MacQueen J. (1967) Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, ed. L.M.Le Cam and J. Neyman, **1**, Berkeley, California: University of California Press, 281-297.

Marriott F.H.C. (1971) Practical problems in a method of cluster analysis. *Biometrics*, **27**, 501-514.

Mendell N.R., Finch S.J., and Thode H.C. (1993) Where is the likelihood ratio test powerful for detecting two component normal mixtures? *Biometrics*, **49**, 907-915.

Moore, A. (1999) Very fast EM-based mixture model clustering using multiresolution kd-trees. In *Advances in Neural Information Processing Systems 12*, San Francisco, CA: Morgan Kaufmann.

Neal, R. and Hinton, G. (1998) A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, Jordan, M. I. (ed.), Amsterdam: Kluwer, 355–371.

Rao M.R. (1971) Cluster analysis and mathematical programming. *Journal of the American Statistical Association*, **66**, 622-626.

Reyment R. and Jöreskog K.G. (1993) *Applied Factor Analysis in the Natural Sciences.* Cambridge: Cambridge University Press.

Shepard R.N. and Arabie P. (1979) Additive clustering: representation of similarities as combinations of discrete overlapping properties. *Psychological Review*, **86**, 87-123.

Schwarz, G. (1978) Estimating the dimension of a model. *Annals of Statistics*, 6: 461–462.

Sibson R. (1973) SLINK: an optimally efficient algorithm for the single link method. *Computer Journal*, **16**, 30-34.

Smyth, P. (1996) Clustering using Monte-Carlo cross validation. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Menlo Park, CA: AAAI Press, pp.126–133.

Smyth, P. (1997) Clustering sequences using hidden Markov models. In *Advances in Neural Information Processing 9*, M. C. Mozer, M. I. Jordan and T. Petsche (eds.), Cambridge, MA: MIT Press, 648–654.

Smyth, P., Ide, K., Ghil, M. (1999, in press) Multiple regimes in Northern hemisphere height fields via mixture model clustering. *Journal of Atmospheric Science.*

Spath H. (1985) *Cluster Analysis and Dissection.* Chichester: Ellis Horwood.

Titterington D.M., Smith A.F.M., and Makov U.E. (1985) *Statistical Analysis of Finite Mixture Distributions.* New York: Wiley.

Wermuth N. (1980) Linear recursive equations, covariance selection and path analysis. *Journal of the American Statistical Association*, **75**, 963-972.

Whittaker J. (1990) *Graphical Models in Applied Multivariate Statistics.* Chichester: Wiley.

Wright S. (1921) Correlation and causation. *Journal of Agricultural Research*, **20**, 557-585.

Wright S. (1923) Theory of path coefficients: A reply to Niles' criticism. *Genetics*, **8**, 239-255.

Wright S. (1934) The method of path coefficients. *Annals of Mathematical Statistics*, **5**, 161-215.

Notes and To Do List

1. The history of the chapter is:
   - original version written in Word by David
   - converted by Heikki to LaTeX, edited by Heikki, transferred to Padhraic June 18th
   - Padhraic made substantial edits to clustering sections, July 27th/28th.
   - Edited by Padhraic again from November 7th to 15th 1999.
2. David's section on path models and factor analysis needs to be edited (in light of earlier section changes), tables cleaned up: currently this section is commented out completely.
3. It would be nice to add some figures in the Gaussian/parametric models section, eg., (1) some simple shapes of Gaussian ellipses over simulated 2d data (easy to do in MATLAB), (2) show data for "income" (e.g., census data set?) to explain why it is non-Gaussian but might be log-normal.
4. Some of David's postscript figures are not readable (figure f10.4.ps, and some higher numbers: not sure we will be using these figures, but will need to fix this if we use them.
5. integrate and check Heikki's discrete mixtures example (add a note on applications to transaction data)
6. I removed fuzzy clustering section since the probabilistic clustering is the same concept (but a better way to do it in my opinion): the fuzzy cluster references should be removed or just stay?
7. Add mention of typically how many iterations k-means clustering takes? I imagine it is (weakly) dependent on n Are there any known results on this?
8. Check with David if in the discussion after the "single-link" and "furthest-link" cluster distances, that it is ok if $d(\mathbf{x}, \mathbf{y})$ is *any* distance measure: or is Euclidean distance implicitly implied in the discussion of shapes? If so, then add a phrase, "In the case when $\mathbf{x}$ is in a Euclidean space..."
9. Possibly add the EM normal multivariate update equation.
10. We could more figures to illustrate some of the sort of cluster structures we are talking about (e.g., single link, etc). (Duda and Hart for example have some nice examples on this, which are invaluable in teaching). Volunteers?
11. very minor point to Heikki: is the use of "od" at the ends of loops standard notation in pseudocode. Informally I usually omit the "do" statement entirely, and just put an "end" at the end of every for and while loop. Not a big deal, but the "od" looked odd to me at first until I realized it was "do" backwards.

12. We ideally still need a nice example of a belief network for this section: I can probably come up with one (I recall David that you mentioned you saw a nice example recently in JRSS?).

13. Clustering Web pages, documents, etc - there are some nice recent ideas (using graph theory) on clustering Web pages (Kleinberg + various IBM people). It would be nice to add a section or display on this at the end of the clustering section since (a) it is non-vector data and the clustering algorithms are somewhat unique, and (b) these are very massive data sets and computational issues are important. Heikki, care to write a section up on this? I think it would be a nice addition and make the material a bit more current. (This may be going in Chapter 14: Heikki? it might be better to put it in this Chapter though).

14. We could easily add 2 short subsections on (1) density and distributions on sequences and structured data, (2) then use this as the basis for a brief review of clustering sequences and time-series using mixture models.

15. There's a section on summary statistics at the start of the Chapter which is too basic (I think it came from an old version of Chapter 5 or 7 originally): it would be good to find a "home" for it in one of the earlier chapters.

16. Some of the accents on foreign names in the references got messed up in the conversion from Word (Spath for example)

17. Need to add a few more applications examples in the references, e.g., Poulsen, Wedel, etc.

18. Any reason that the book by Jain and Dubes on clustering is not referenced - it seems like a useful one and written from a computer science viewpoint.

19. Heikki had a note that another paragraph should be added on recent "fast" clustering algorithms from data mining/database people.