CHAPTER

10

UNSUPERVISED LEARNING AND CLUSTERING

10.1 INTRODUCTION

Until now we have assumed that the training samples used to design a classifier were labeled by their category membership. Procedures that use labeled samples are said to be supervised. Now we shall investigate a number of *unsupervised* procedures, which use unlabeled samples. That is, we shall see what can be done when all one has is a collection of samples without being told their categories.

One might wonder why anyone is interested in such an unpromising problem, and whether or not it is possible even in principle to learn anything of value from unlabeled samples. There are at least five basic reasons for interest in unsupervised procedures. First, collecting and labeling a large set of sample patterns can be surprisingly costly. For instance, recorded speech is virtually free, but accurately *labeling* the speech—marking what word or phoneme is being uttered at each instant—can be very expensive and time consuming. If a classifier can be crudely designed on a small set of labeled samples, and then "tuned up" by allowing it to run without supervision on a large, unlabeled set, much time and trouble can be saved. Second, one might wish to proceed in the reverse direction: Train with large amounts of (less expensive) unlabeled data, and only then use supervision to label the groupings found. This may be appropriate for large "data mining" applications, where the contents of a large database are not known beforehand. Third, in many applications the characteristics of the patterns can change slowly with time—for example, in automated food classification as the seasons change. If these changes can be tracked by a classifier running in an unsupervised mode, improved performance can be achieved. Fourth, we can use unsupervised methods to find *features* that will then be useful for categorization. There are unsupervised methods that provide a form of data-dependent "smart preprocessing" or "smart feature extraction." Lastly, in the early stages of an investigation it may be valuable to perform exploratory data analysis and thereby gain some insight into the nature or structure of the data. The discovery of distinct subclasses—clusters or groups of patterns whose members are more similar to each other than they are to other patterns—or of major departures from expected characteristics may suggest we significantly alter our approach to designing the classifier.

The answer to the question of whether or not it is possible in principle to learn anything from unlabeled data depends upon the assumptions one is willing to accept—theorems cannot be proved without premises. We shall begin with the very restrictive assumption that the functional forms for the underlying probability densities are known and that the only thing that must be learned is the value of an unknown parameter vector. Interestingly enough, the formal solution to this problem will turn out to be almost identical to the solution for the problem of supervised learning given in Chapter 3. Unfortunately, in the unsupervised case the solution suffers from the usual problems associated with parametric assumptions without providing any of the benefits of computational simplicity. This will lead us to various attempts to reformulate the problem as one of partitioning the data into subgroups or clusters. While some of the resulting clustering procedures have no known significant theoretical properties, they are still among the more useful tools for pattern recognition problems.

10.2 MIXTURE DENSITIES AND IDENTIFIABILITY

We begin by assuming that we know the complete probability structure for the problem with the sole exception of the values of some parameters. To be more specific, we make the following assumptions:

1. The samples come from a known number c of classes.
2. The prior probabilities $P(\omega_j)$ for each class are known, $j = 1, \ldots, c$.
3. The forms for the class-conditional probability densities $p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)$ are known, $j = 1, \ldots, c$.
4. The values for the c parameter vectors $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_c$ are unknown.
5. The category labels are unknown.

We assume the samples were obtained by selecting a state of nature ω_j with probability $P(\omega_j)$ and then selecting an $\mathbf{x}$ according to the probability law $p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)$. Thus, the probability density function for the samples is given by

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{c} p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j) P(\omega_j), \tag{1}$$

COMPONENT DENSITIES
MIXING PARAMETERS

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_c)^t$. For obvious reasons, a density function of this form is called a *mixture density*. The conditional densities $p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)$ are called the *component densities*, and the prior probabilities $P(\omega_j)$ are called the *mixing parameters*. The mixing parameters can also be included among the unknown parameters, but for the moment we shall assume that only $\boldsymbol{\theta}$ is unknown.

Our basic goal will be to use samples drawn from this mixture density to estimate the unknown parameter vector $\boldsymbol{\theta}$. Once we know $\boldsymbol{\theta}$, we can decompose the mixture into its components and use a maximum *a posteriori* classifier on the derived densities, if indeed classification is our final goal. Before seeking explicit solutions to this problem, however, let us ask whether or not it is possible in principle to recover $\boldsymbol{\theta}$ from the mixture. Suppose that we had an unlimited number of samples and that we used one of the nonparametric methods of Chapter 4 to determine the value of $p(\mathbf{x}|\boldsymbol{\theta})$ for every $\mathbf{x}$. If there is only one value of $\boldsymbol{\theta}$ that will produce the observed values for $p(\mathbf{x}|\boldsymbol{\theta})$, then a solution is at least possible in principle. However, if several

different values of $\boldsymbol{\theta}$ can produce the same values for $p(\mathbf{x}|\boldsymbol{\theta})$, then there is no hope of obtaining a unique solution.

These considerations lead us to the following definition: A density $p(\mathbf{x}|\boldsymbol{\theta})$ is said to be *identifiable* if $\boldsymbol{\theta} \neq \boldsymbol{\theta}'$ implies that there exists an $\mathbf{x}$ such that $p(\mathbf{x}|\boldsymbol{\theta}) \neq p(\mathbf{x}|\boldsymbol{\theta}')$. Or put another way, a density $p(\mathbf{x}|\boldsymbol{\theta})$ is *not* identifiable if we cannot recover a unique $\boldsymbol{\theta}$, even from an infinite amount of data. In the discouraging situation where we cannot infer *any* of the individual parameters (i.e., components of $\boldsymbol{\theta}$), the density is *completely unidentifiable*. Note that the identifiability of $\boldsymbol{\theta}$ is a property of the *model*, irrespective of any procedure we might use to determine its value. As one might expect, the study of unsupervised learning is greatly simplified if we restrict ourselves to identifiable mixtures. Fortunately, most mixtures of commonly encountered density functions are identifiable, as are most complex or high-dimensional density functions encountered in real-world problems.

COMPLETE UNIDENTIFIABILITY

Mixtures of discrete distributions are not always so obliging. As a simple example consider the case where x is binary and $P(x|\boldsymbol{\theta})$ is the mixture

$$P(x|\boldsymbol{\theta}) = \frac{1}{2}\theta_1^x(1-\theta_1)^{1-x} + \frac{1}{2}\theta_2^x(1-\theta_2)^{1-x}$$
$$= \begin{cases} \frac{1}{2}(\theta_1+\theta_2) & \text{if } x = 1 \\ 1 - \frac{1}{2}(\theta_1+\theta_2) & \text{if } x = 0. \end{cases}$$

Suppose, for example, that we know for our data that $P(x = 1|\boldsymbol{\theta}) = 0.6$ and hence that $P(x = 0|\boldsymbol{\theta}) = 0.4$. Then we know the function $P(x|\boldsymbol{\theta})$, but we cannot determine $\boldsymbol{\theta}$, and hence cannot extract the component distributions. The most we can say is that $\theta_1 + \theta_2 = 1.2$. Thus, here we have a case in which the mixture distribution is completely unidentifiable, and hence a case for which unsupervised learning is impossible in principle. Related situations may permit us to determine one or *some* parameters, but not all (Problem 3).

This kind of problem commonly occurs with discrete distributions. If there are too many components in the mixture, there may be more unknowns than independent equations, and identifiability can be a serious problem. For the continuous case, the problems are less severe, although certain minor difficulties can arise due to the possibility of special cases. While it can be shown that mixtures of normal densities are usually identifiable, the parameters in the simple mixture density

$$p(x|\boldsymbol{\theta}) = \frac{P(\omega_1)}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x-\theta_1)^2\right] + \frac{P(\omega_2)}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x-\theta_2)^2\right] \quad (2)$$

cannot be uniquely identified if $P(\omega_1) = P(\omega_2)$, because then θ_1 and θ_2 can be interchanged without affecting $p(x|\boldsymbol{\theta})$. To avoid such irritations, we shall acknowledge that identifiability can be a problem, but shall henceforth assume that the mixture densities we are working with are identifiable.

10.3 MAXIMUM-LIKELIHOOD ESTIMATES

Suppose now that we are given a set $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of n unlabeled samples drawn independently from the mixture density

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{c} p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)P(\omega_j), \quad (1)$$

where the full parameter vector $\boldsymbol{\theta}$ is fixed but unknown. The likelihood of the observed samples is, by definition, the joint density

$$p(\mathcal{D}|\boldsymbol{\theta}) \equiv \prod_{k=1}^{n} p(\mathbf{x}_k|\boldsymbol{\theta}). \tag{3}$$

The maximum-likelihood estimate $\hat{\boldsymbol{\theta}}$ is that value of $\boldsymbol{\theta}$ that maximizes $p(\mathcal{D}|\boldsymbol{\theta})$.

If we assume that $p(\mathcal{D}|\boldsymbol{\theta})$ is a differentiable function of $\boldsymbol{\theta}$, then we can derive some interesting necessary conditions for $\hat{\boldsymbol{\theta}}$. Let l be the logarithm of the likelihood, and let $\nabla_{\boldsymbol{\theta}_i} l$ be the gradient of l with respect to $\boldsymbol{\theta}_i$. Then

$$l = \sum_{k=1}^{n} \ln p(\mathbf{x}_k|\boldsymbol{\theta}) \tag{4}$$

and

$$\nabla_{\boldsymbol{\theta}_i} l = \sum_{k=1}^{n} \frac{1}{p(\mathbf{x}_k|\boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}_i} \left[\sum_{j=1}^{c} p(\mathbf{x}_k|\omega_j, \boldsymbol{\theta}_j) P(\omega_j) \right]. \tag{5}$$

If we assume that the elements of $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ are functionally independent if $i \neq j$, and if we introduce the posterior probability

$$P(\omega_i|\mathbf{x}_k, \boldsymbol{\theta}) = \frac{p(\mathbf{x}_k|\omega_i, \boldsymbol{\theta}_i) P(\omega_i)}{p(\mathbf{x}_k|\boldsymbol{\theta})}, \tag{6}$$

we see that the gradient of the log-likelihood can be written in the interesting form

$$\nabla_{\boldsymbol{\theta}_i} l = \sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}_i} \ln p(\mathbf{x}_k|\omega_i, \boldsymbol{\theta}_i). \tag{7}$$

Because the gradient must vanish at the value of $\boldsymbol{\theta}_i$ that maximizes l, the maximum-likelihood estimate $\hat{\boldsymbol{\theta}}_i$ must satisfy the conditions

$$\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) \nabla_{\boldsymbol{\theta}_i} \ln p(\mathbf{x}_k|\omega_i, \hat{\boldsymbol{\theta}}_i) = 0 \qquad i = 1, \ldots, c. \tag{8}$$

Among the solutions to these equations for $\hat{\boldsymbol{\theta}}_i$ we will find the maximum-likelihood solution.

It is not hard to generalize these results to include the prior probabilities $P(\omega_i)$ among the unknown quantities. In this case the search for the maximum value of $p(\mathcal{D}|\boldsymbol{\theta})$ extends over $\boldsymbol{\theta}$ and $P(\omega_i)$, subject to the constraints

$$P(\omega_i) \geq 0 \qquad i = 1, \ldots, c \tag{9}$$

and

$$\sum_{i=1}^{c} P(\omega_i) = 1. \tag{10}$$

Let $\hat{P}(\omega_i)$ be the maximum-likelihood estimate for $P(\omega_i)$, and let $\hat{\boldsymbol{\theta}}_i$ be the maximum-likelihood estimate for $\boldsymbol{\theta}_i$. It can be shown (Problem 6) that if the likelihood function is differentiable and if $\hat{P}(\omega_i) \neq 0$ for any i, then $\hat{P}(\omega_i)$ and $\hat{\boldsymbol{\theta}}_i$ must satisfy

$$\hat{P}(\omega_i) = \frac{1}{n}\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) \tag{11}$$

and

$$\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})\nabla_{\boldsymbol{\theta}_i} \ln p(\mathbf{x}_k|\omega_i, \hat{\boldsymbol{\theta}}_i) = 0, \tag{12}$$

where

$$\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) = \frac{p(\mathbf{x}_k|\omega_i, \hat{\boldsymbol{\theta}}_i)\hat{P}(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}_k|\omega_j, \hat{\boldsymbol{\theta}}_j)\hat{P}(\omega_j)}. \tag{13}$$

These equations have the following interpretation. Equation 11 states that the maximum-likelihood estimate of the probability of a category is the average over the entire data set of the estimate derived from each sample—each sample is weighted equally. Equation 13 is ultimately related to Bayes Theorem, but notice that in estimating the probability for class ω_i, the numerator on the right-hand side depends on $\hat{\boldsymbol{\theta}}_i$ and not the full $\hat{\boldsymbol{\theta}}$ directly. While Eq. 12 is a bit subtle, we can understand it clearly in the trivial $n = 1$ case. Because $\hat{P} \neq 0$, this case states merely that the probability density is maximized as a function of $\boldsymbol{\theta}_i$—surely what is needed for the maximum-likelihood solution.

10.4 APPLICATION TO NORMAL MIXTURES

It is enlightening to see how these general results apply to the case where the component densities are multivariate normal, $p(\mathbf{x}|\omega_i, \boldsymbol{\theta}_i) \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. The following table illustrates a few of the different cases that can arise depending upon which parameters are known (×) and which are unknown (?):

Case	$\boldsymbol{\mu}_i$	$\boldsymbol{\Sigma}_i$	$P(\omega_i)$	c
1	?	×	×	×
2	?	?	?	×
3	?	?	?	?

Case 1 is the simplest, and it will be considered in detail because of its pedagogical value. Case 2 is more realistic, though somewhat more involved. Case 3 represents the problem we face on encountering a completely unknown set of data; unfortunately, it cannot be solved by maximum-likelihood methods. We shall postpone discussion of what can be done when the number of classes is unknown until Section 10.10.

10.4.1 Case 1: Unknown Mean Vectors

If the only unknown quantities are the mean vectors $\boldsymbol{\mu}_i$, then of course $\boldsymbol{\theta}_i$ consists of the components of $\boldsymbol{\mu}_i$. Equation 8 can then be used to obtain necessary conditions on the maximum-likelihood estimate for $\boldsymbol{\mu}_i$. Because the likelihood is

$$\ln p(\mathbf{x}|\omega_i, \boldsymbol{\mu}_i) = -\ln\left[(2\pi)^{d/2}|\boldsymbol{\Sigma}_i|^{1/2}\right] - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i), \tag{14}$$

its derivative is

$$\nabla_{\boldsymbol{\mu}_i} \ln p(\mathbf{x}|\omega_i, \boldsymbol{\mu}_i) = \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i). \tag{15}$$

Thus according to Eq. 8, the maximum-likelihood estimate $\hat{\boldsymbol{\mu}}_i$ must satisfy

$$\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}})\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i) = 0, \qquad \text{where } \hat{\boldsymbol{\mu}} = (\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_c)^t. \tag{16}$$

After multiplying by $\boldsymbol{\Sigma}_i$ and rearranging terms, we obtain the solution

$$\hat{\boldsymbol{\mu}}_i = \frac{\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}})\mathbf{x}_k}{\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}})}. \tag{17}$$

This equation is intuitively very satisfying. It shows that the maximum-likelihood estimate for $\boldsymbol{\mu}_i$ is merely a weighted average of the samples; the weight for the kth sample is an estimate of how likely it is that $\mathbf{x}_k$ belongs to the ith class. If $P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}})$ happened to be 1.0 for some of the samples and 0.0 for the rest, then $\hat{\boldsymbol{\mu}}_i$ would be the mean of those samples estimated to belong to the ith class. More generally, suppose that $\hat{\boldsymbol{\mu}}_i$ is sufficiently close to the true value of $\boldsymbol{\mu}_i$ that $P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}})$ is essentially the true posterior probability for ω_i. If we think of $P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}})$ as the fraction of those samples having value $\mathbf{x}_k$ that come from the ith class, then we see that Eq. 17 essentially gives $\hat{\boldsymbol{\mu}}_i$ as the average of the samples coming from the ith class.

Unfortunately, Eq. 17 does not give $\hat{\boldsymbol{\mu}}_i$ explicitly, and if we substitute

$$P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}}) = \frac{p(\mathbf{x}_k|\omega_i, \hat{\boldsymbol{\mu}}_i)P(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}_k|\omega_j, \hat{\boldsymbol{\mu}}_j)P(\omega_j)}$$

with $p(\mathbf{x}|\omega_i, \hat{\boldsymbol{\mu}}_i) \sim N(\hat{\boldsymbol{\mu}}_i, \boldsymbol{\Sigma}_i)$, we obtain a tangled snarl of coupled simultaneous nonlinear equations. These equations usually do not have a unique solution, and we must test the solutions we get to find the one that actually maximizes the likelihood.

If we have some way of obtaining fairly good initial estimates $\hat{\boldsymbol{\mu}}_i(0)$ for the unknown means, Eq. 17 suggests the following iterative scheme for improving the estimates:

$$\hat{\boldsymbol{\mu}}_i(j+1) = \frac{\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}}(j))\mathbf{x}_k}{\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}}(j))}. \tag{18}$$

This is basically a gradient ascent or hill-climbing procedure for maximizing the log-likelihood function. If the overlap between component densities is small, then the coupling between classes will be small and convergence will be fast. However, when convergence does occur, all that we can be sure of is that the gradient is zero. Like all hill-climbing procedures, this one carries no guarantee of yielding the global

maximum. Note too that if the model is misspecified (e.g., we assume the "wrong" number of clusters), then the estimate of the log-likelihood given by Eq. 18 can actually decrease.

To illustrate the kind of behavior that can occur in gradient searches in unsupervised learning, consider the simple two-component one-dimensional normal mixture:

$$p(x|\mu_1, \mu_2) = \underbrace{\frac{1}{3\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x-\mu_1)^2\right]}_{\omega_1} + \underbrace{\frac{2}{3\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x-\mu_2)^2\right]}_{\omega_2}, \quad (19)$$

where ω_i denotes a Gaussian component. The 25 samples shown in the table below were drawn sequentially from this mixture with $\mu_1 = -2$ and $\mu_2 = 2$. Let us use these samples to compute the log-likelihood function

$$l(\mu_1, \mu_2) = \sum_{k=1}^{n} \ln p(x_k|\mu_1, \mu_2) \quad (20)$$

for various values of μ_1 and μ_2. The bottom of Fig. 10.1 shows how l varies with μ_1 and μ_2. The maximum value of l occurs at $\hat{\mu}_1 = -2.130$ and $\hat{\mu}_2 = 1.668$, which is in the rough vicinity of the true values $\mu_1 = -2$ and $\mu_2 = 2$. However, l reaches another peak of comparable height at $\hat{\mu}_1 = 2.085$ and $\hat{\mu}_2 = -1.257$.

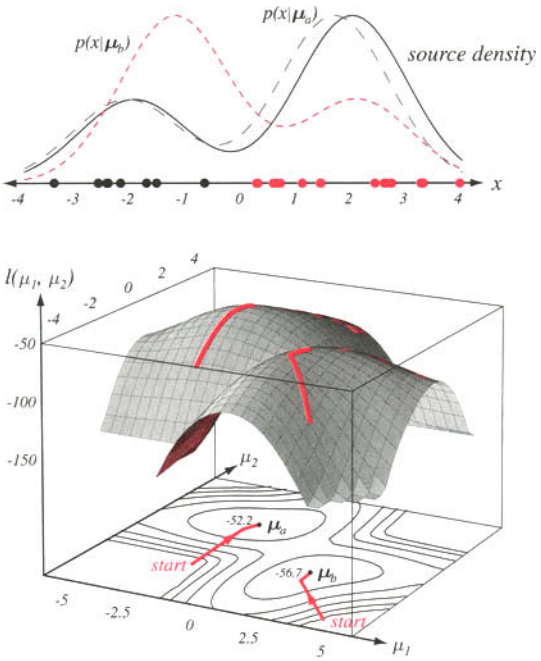


FIGURE 10.1. (Above) The source mixture density used to generate sample data, and two maximum-likelihood estimates based on the data in the table. (Bottom) Log-likelihood of a mixture model consisting of two univariate Gaussians as a function of their means, for the data in the table. Trajectories for the iterative maximum-likelihood estimation of the means of a two-Gaussian mixture model based on the data are shown as red lines. Two local optima (with log-likelihoods −52.2 and −56.7) correspond to the two density estimates shown above.

Roughly speaking, this solution corresponds to interchanging μ_1 and μ_2. Note that had the prior probabilities been equal, interchanging μ_1 and μ_2 would have produced no change in the log-likelihood function. Thus, as we mentioned before, when the mixture density is not identifiable, the maximum-likelihood solution is not unique.

k	x_k	ω_1	ω_2	k	x_k	ω_1	ω_2	k	x_k	ω_1	ω_2
1	0.608		×	9	0.262		×	17	−3.458	×	
2	−1.590	×		10	1.072		×	18	0.257		×
3	0.235		×	11	−1.773	×		19	2.569		×
4	3.949		×	12	0.537		×	20	1.415		×
5	−2.249	×		13	3.240		×	21	1.410		×
6	2.704		×	14	2.400		×	22	−2.653	×	
7	−2.473	×		15	−2.499	×		23	1.396		×
8	0.672		×	16	2.608		×	24	3.286		×
								25	−0.712	×	

Additional insight into the nature of these multiple solutions can be obtained by examining the resulting estimates for the mixture density. The figure at the top shows the true (source) mixture density and the estimates obtained by using the two maximum-likelihood estimates as if they were the true parameter values. The 25 sample values are shown as a scatter of points along the abscissa—ω_1 points in black, ω_2 points in red. Note that the peaks of both the true mixture density and the maximum-likelihood solutions are located so as to encompass two major groups of data points. The estimate corresponding to the smaller local maximum of the log-likelihood function has a mirror-image shape, but its peaks also encompass reasonable groups of data points. To the eye, neither of these solutions is clearly superior, and both are interesting.

If Eq. 18 is used to determine solutions to Eq. 17 iteratively, the results depend on the starting values $\hat{\mu}_1(0)$ and $\hat{\mu}_2(0)$. The bottom figure shows trajectories from two different starting points. Although not shown, if $\hat{\mu}_1(0) = \hat{\mu}_2(0)$, convergence to a saddle point occurs in one step. This is not a coincidence; it happens for the simple reason that for this starting point $P(\omega_i|x_k, \hat{\mu}_1(0), \hat{\mu}_2(0)) = P(\omega_i)$. In such a case Eq. 18 yields the mean of all of the samples for $\hat{\mu}_1$ and $\hat{\mu}_2$ for all successive iterations. Clearly, this is a general phenomenon, and such saddle-point solutions can be expected if the starting point does not bias the search away from a symmetric answer.

10.4.2 Case 2: All Parameters Unknown

If $\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$, and $P(\omega_i)$ are all unknown and if no constraints are placed on the covariance matrix, then the maximum-likelihood principle yields useless singular solutions. The reason for this can be appreciated from the following simple example in one dimension. Let $p(x|\mu, \sigma^2)$ be the two-component normal mixture:

$$p(x|\mu, \sigma^2) = \frac{1}{2\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] + \frac{1}{2\sqrt{2\pi}} \exp\left[-\frac{1}{2}x^2\right].$$

The likelihood function for n samples drawn from this probability density is merely the product of the n densities $p(x_k|\mu, \sigma^2)$. Suppose that we make the choice $\mu = x_1$, so that

$$p(x_1|\mu, \sigma^2) = \frac{1}{2\sqrt{2\pi}\sigma} + \frac{1}{2\sqrt{2\pi}} \exp\left[-\frac{1}{2}x_1^2\right].$$

Clearly, for the rest of the samples

$$p(x_k|\mu,\sigma^2) \geq \frac{1}{2\sqrt{2\pi}} \exp\left[-\frac{1}{2}x_k^2\right],$$

so that

$$p(x_1,\ldots,x_n|\mu,\sigma^2) \geq \left\{\frac{1}{\sigma} + \exp\left[-\frac{1}{2}x_1^2\right]\right\} \frac{1}{(2\sqrt{2\pi})^n} \exp\left[-\frac{1}{2}\sum_{k=2}^{n} x_k^2\right].$$

Thus, the first factor at the right shows that by letting σ approach zero we can make the likelihood arbitrarily large, and the maximum-likelihood solution is singular.

Ordinarily, singular solutions are of no interest, and we are forced to conclude that the maximum-likelihood principle fails for this class of normal mixtures. However, it is an empirical fact that meaningful solutions can still be obtained if we restrict our attention to the largest of the finite local maxima of the likelihood function. Assuming that the likelihood function is well-behaved at such maxima, we can use Eqs. 11–13 to obtain estimates for $\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$, and $P(\omega_i)$. When we include the elements of $\boldsymbol{\Sigma}_i$ in the elements of the parameter vector $\boldsymbol{\theta}_i$, we must remember that only half of the off-diagonal elements are independent. In addition, it turns out to be much more convenient to let the independent elements of $\boldsymbol{\Sigma}_i^{-1}$ rather than $\boldsymbol{\Sigma}_i$ be the unknown parameters. With these observations, the actual differentiation of

$$\ln p(\mathbf{x}_k|\omega_i,\boldsymbol{\theta}_i) = \ln \frac{|\boldsymbol{\Sigma}_i^{-1}|^{1/2}}{(2\pi)^{d/2}} - \frac{1}{2}(\mathbf{x}_k - \boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}_k - \boldsymbol{\mu}_i) \tag{21}$$

with respect to the elements of $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i^{-1}$ is relatively routine. Let $x_p(k)$ be the pth element of $\mathbf{x}_k$, $\mu_p(i)$ be the pth element of $\boldsymbol{\mu}_i$, $\sigma_{pq}(i)$ be the pqth element of $\boldsymbol{\Sigma}_i$, and $\sigma^{pq}(i)$ be the pqth element of $\boldsymbol{\Sigma}_i^{-1}$. Then differentiation gives

$$\nabla_{\boldsymbol{\mu}_i} \ln p(\mathbf{x}_k|\omega_i,\boldsymbol{\theta}_i) = \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}_k - \boldsymbol{\mu}_i) \tag{22}$$

and

$$\frac{\partial \ln p(\mathbf{x}_k|\omega_i,\boldsymbol{\theta}_i)}{\partial \sigma^{pq}(i)} = \left(1 - \frac{\delta_{pq}}{2}\right)\left[\sigma_{pq}(i) - (x_p(k) - \mu_p(i))(x_q(k) - \mu_q(i))\right], \tag{23}$$

where δ_{pq} is the Kronecker delta. We substitute these results into Eq. 12 and perform a small amount of algebraic manipulation (Problem 17) and thereby obtain the following equations for the local-maximum-likelihood estimates $\hat{\boldsymbol{\mu}}_i$, $\hat{\boldsymbol{\Sigma}}_i$, and $\hat{P}(\omega_i)$:

$$\hat{P}(\omega_i) = \frac{1}{n}\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k,\hat{\boldsymbol{\theta}}) \tag{24}$$

$$\hat{\boldsymbol{\mu}}_i = \frac{\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k,\hat{\boldsymbol{\theta}})\mathbf{x}_k}{\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k,\hat{\boldsymbol{\theta}})} \tag{25}$$

$$\hat{\boldsymbol{\Sigma}}_i = \frac{\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k,\hat{\boldsymbol{\theta}})(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)^t}{\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k,\hat{\boldsymbol{\theta}})} \tag{26}$$

where

$$\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) = \frac{p(\mathbf{x}_k|\omega_i, \hat{\boldsymbol{\theta}}_i)\hat{P}(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}_k|\omega_j, \hat{\boldsymbol{\theta}}_j)\hat{P}(\omega_j)}$$

$$= \frac{|\hat{\boldsymbol{\Sigma}}_i|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)^t \hat{\boldsymbol{\Sigma}}_i^{-1}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)\right]\hat{P}(\omega_i)}{\sum_{j=1}^{c} |\hat{\boldsymbol{\Sigma}}_j|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_j)^t \hat{\boldsymbol{\Sigma}}_j^{-1}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_j)\right]\hat{P}(\omega_j)}. \quad (27)$$

While the notation may make these equations appear to be rather formidable, their interpretation is actually quite simple. In the extreme case where $\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})$ is 1.0 when $\mathbf{x}_k$ is from Class ω_i and 0.0 otherwise, $\hat{P}(\omega_i)$ is the fraction of samples from ω_i, $\hat{\boldsymbol{\mu}}_i$ is the mean of those samples, and $\hat{\boldsymbol{\Sigma}}_i$ is the corresponding sample covariance matrix. More generally, $\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})$ is between 0.0 and 1.0, and all of the samples play some role in the estimates. However, the estimates are basically still frequency ratios, sample means, and sample covariance matrices.

The problems involved in solving these implicit equations are similar to the problems discussed in Section 10.4.1, with the additional complication of having to avoid singular solutions. Of the various techniques that can be used to obtain a solution, the most obvious approach is to use initial estimates to evaluate Eq. 27 for $\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})$ and then to use Eqs. 24–26 to update these estimates. If the initial estimates are very good, having perhaps been obtained from a fairly large set of labeled samples, convergence can be quite rapid. However, the results do depend upon the starting point, and the problem of multiple solutions is always present. Furthermore, the repeated computation and inversion of the sample covariance matrices can be quite time consuming.

Considerable simplification can be obtained if it is possible to assume that the covariance matrices are diagonal. This has the added virtue of reducing the number of unknown parameters, which is very important when the number of samples is not large. If this assumption is too strong, it still may be possible to obtain some simplification by assuming that the c covariance matrices are equal, which also may eliminate the problem of singular solutions (Problem 17).

10.4.3 *k*-Means Clustering

Of the various techniques that can be used to simplify the computation and accelerate convergence, we shall briefly consider one elementary but very popular approximate method. We are tempted to call it the c-means procedure, since its goal is to find the c mean vectors $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_c$. However, it is much more widely known as k-means clustering, where k (which is the same as our c) is the number of cluster centers. Thus we shall follow common practice and call it by that name.

From Eq. 27, it is clear that the probability $\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})$ is large when the squared Mahalanobis distance $(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)^t \hat{\boldsymbol{\Sigma}}_i^{-1}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)$ is small. Suppose that we merely compute the squared Euclidean distance $\|\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i\|^2$, find the mean $\hat{\boldsymbol{\mu}}_m$ nearest to $\mathbf{x}_k$, and approximate $\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})$ as

$$\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) \simeq \begin{cases} 1 & \text{if } i = m \\ 0 & \text{otherwise.} \end{cases} \quad (28)$$

Then the iterative application of Eq. 25 leads to the following procedure for finding $\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_c$. In the absence of other information, we may need to guess the "proper"

number of clusters, c. Likewise, we may assign c based on the final application. For example we might set $c = 26$ in a handwritten English letter application, whether or not the data fall into 26 natural clusters. We shall return in Section 10.10 to this problem of cluster validity.

Here and throughout, we denote the known number of patterns as n, and the desired number of clusters c. It is traditional to let c samples randomly chosen from the data set serve as initial cluster centers. The algorithm is then:

■ **Algorithm 1. (k-Means Clustering)**

```
1 begin initialize n, c, μ1, μ2, ..., μc
2         do classify n samples according to nearest μi
3            recompute μi
4         until no change in μi
5   return μ1, μ2, ..., μc
6 end
```

The computational complexity of the algorithm is $O(ndcT)$ where d is the number of features and T is the number of iterations (Problem 16). In practice, the number of iterations is generally much less than the number of samples.

This is typical of a class of procedures that are known as *clustering* procedures or algorithms. Later on we shall place it in the class of iterative optimization procedures, because the means tend to move so as to minimize a squared-error criterion function. For the moment we view it merely as an approximate way to obtain maximum-likelihood estimates for the means. The values obtained can be accepted as the answer, or can be used as starting points for the more exact computations.

It is interesting to see how this procedure behaves on the example data we saw in Fig. 10.1. Figure 10.2 shows the sequence of values for $\hat{\mu}_1$ and $\hat{\mu}_2$ obtained for several different starting points. Since interchanging $\hat{\mu}_1$ and $\hat{\mu}_2$ merely interchanges

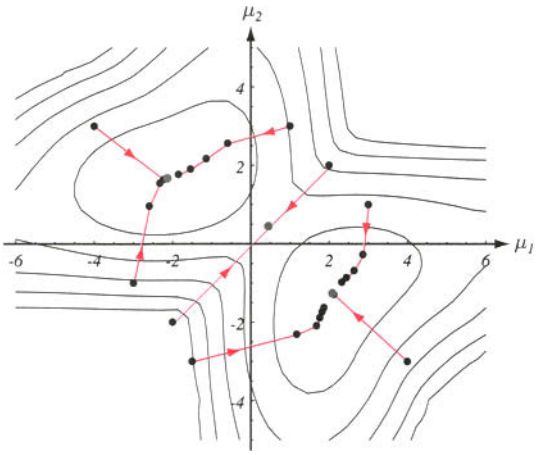


FIGURE 10.2. The k-means clustering procedure is a form of stochastic hill climbing in the log-likelihood function. The contours represent equal log-likelihood values for the one-dimensional data in Fig. 10.1. The dots indicate parameter values after different iterations of the k-means algorithm. Six of the starting points shown lead to local maxima, whereas two (i.e., $\mu_1(0) = \mu_2(0)$) lead to a saddle point near $\boldsymbol{\mu} = \mathbf{0}$.

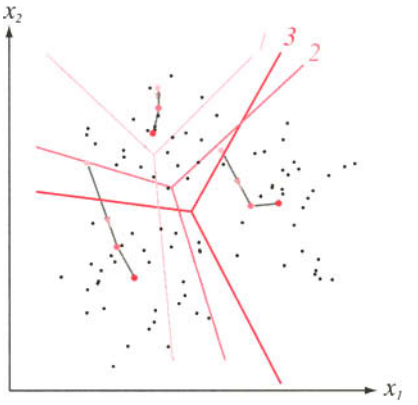


FIGURE 10.3. Trajectories for the means of the k-means clustering procedure applied to two-dimensional data. The final Voronoi tesselation (for classification) is also shown—the means correspond to the "centers" of the Voronoi cells. In this case, convergence is obtained in three iterations.

the labels assigned to the data, the trajectories are symmetric about the line $\hat{\mu}_1 = \hat{\mu}_2$. The trajectories lead either to the point $\hat{\mu}_1 = -2.176, \hat{\mu}_2 = 1.684$ or to its symmetric image. This is close to the solution found by the maximum-likelihood method (namely, $\hat{\mu}_1 = -2.130$ and $\hat{\mu}_2 = 1.688$), and the trajectories show a general resemblance to those shown in Fig. 10.1. In general, when the overlap between the component densities is small the maximum-likelihood approach and the k-means procedure can be expected to give similar results.

Figure 10.3 shows a two-dimensional example, with the assumption of $c = 3$ clusters. The three initial cluster centers, chosen randomly from the training points, and their associated Voronoi tesselation, are shown in pink. According to the algorithm, the points in each of the three Voronoi cells are used to calculate new cluster centers (pink), and so on. Here, after the third iteration the algorithm has converged (red). Because the k-means algorithm is very simple and works well in practice, it is a staple of clustering methods.

*10.4.4 Fuzzy k-Means Clustering

In every iteration of the classical k-means procedure, each data point is assumed to be in exactly one cluster, as implied by Eq. 28 and by lines 2 and 3 of Algorithm 1. We can relax this condition and assume that each sample $\mathbf{x}_j$ has some graded or "fuzzy" membership in a cluster. At root, these "memberships" are equivalent to the probabilities $\hat{P}(\omega_i|\mathbf{x}_j, \hat{\boldsymbol{\theta}})$ given in Eq. 27, where $\hat{\boldsymbol{\theta}}$ is the parameter vector for the membership functions.

The fuzzy k-means clustering algorithm seeks a minimum of a heuristic global cost function

$$J_{fuz} = \sum_{i=1}^{c} \sum_{j=1}^{n} [\hat{P}(\omega_i|\mathbf{x}_j, \hat{\boldsymbol{\theta}})]^b ||\mathbf{x}_j - \boldsymbol{\mu}_i||^2, \tag{29}$$

where b is a free parameter chosen to adjust the "blending" of different clusters. If b is set to 0, J_{fuz} is merely a sum-of-squared errors criterion with each pattern assigned

to a single cluster, which we shall see again in Eq. 54. For $b > 1$, the criterion allows each pattern to belong to multiple clusters.

The probabilities of cluster membership for each point are normalized as

$$\sum_{i=1}^{c} \hat{P}(\omega_i|\mathbf{x}_j) = 1, \qquad j = 1, \ldots, n, \tag{30}$$

where, for simplicity, we have not explicitly shown the dependence on $\hat{\boldsymbol{\theta}}$. We let $\hat{P}_j$ denote the prior probability of $\hat{P}(\omega_j)$, then at the solution (i.e., the minimum of J_{fuz}), we have

$$\partial J_{fuz}/\partial \boldsymbol{\mu}_i = 0 \qquad \text{and} \qquad \partial J_{fuz}/\partial \hat{P}_j = 0. \tag{31}$$

These lead (Problem 15) to the solutions

$$\boldsymbol{\mu}_j = \frac{\sum_{j=1}^{n}[\hat{P}(\omega_i|\mathbf{x}_j)]^b \mathbf{x}_j}{\sum_{j=1}^{n}[\hat{P}(\omega_i|\mathbf{x}_j)]^b} \tag{32}$$

and

$$\hat{P}(\omega_i|\mathbf{x}_j) = \frac{(1/d_{ij})^{1/(b-1)}}{\sum_{r=1}^{c}(1/d_{rj})^{1/(b-1)}} \qquad \text{and} \quad d_{ij} = ||\mathbf{x}_j - \boldsymbol{\mu}_i||^2. \tag{33}$$

In general, the J_{fuz} criterion is minimized when the cluster centers $\boldsymbol{\mu}_j$ are near those points that have high estimated probability of being in cluster j. Because Eqs. 32 and 33 rarely have analytic solutions, the cluster means and point probabilities are estimated iteratively according to the following algorithm:

■ **Algorithm 2. (Fuzzy *k*-Means Clustering)**

```
1 begin initialize n, c, b, μ_1, ..., μ_c, P̂(ω_i | x_j), i = 1..., c; j = 1, ..., n
2         normalize P̂(ω_i | x_j) by Eq. 30
3         do recompute μ_i by Eq. 32
4            recompute P̂(ω_i | x_j) by Eq. 33
5         until small change in μ_i and P̂(ω_i | x_j)
6   return μ_1, μ_2, ..., μ_c
7 end
```

Figure 10.4 illustrates the algorithm. At early iterations the means lie near the center of the full data set because each point has a nonnegligible "membership" (i.e., probability) in each cluster. At later iterations the means separate and each membership tends toward the value 1.0 or 0.0. Clearly, the classical k-means algorithm is just a special case where the memberships for all points obey

$$P(\omega_i|\mathbf{x}_j) = \begin{cases} 1 & \text{if } \|\mathbf{x}_j - \boldsymbol{\mu}_i\| < \|\mathbf{x}_j - \boldsymbol{\mu}_{i'}\| \text{ for all } i' \neq i \\ 0 & \text{otherwise,} \end{cases} \tag{34}$$

as given by Eq. 17.

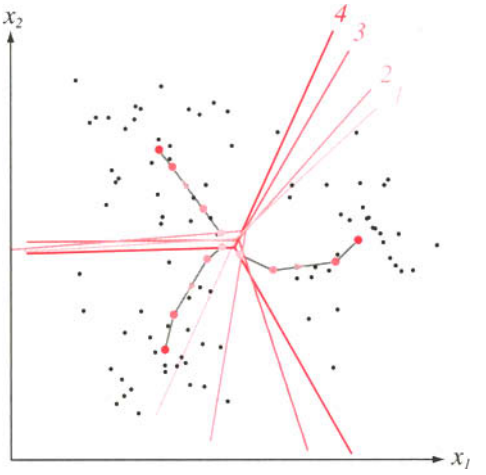


FIGURE 10.4. At each iteration of the fuzzy k-means clustering algorithm, the probability of category memberships for each point are adjusted according to Eqs. 32 and 33 (here $b = 2$). While most points have nonnegligible memberships in two or three clusters, we nevertheless draw the boundary of a Voronoi tesselation to illustrate the progress of the algorithm. After four iterations, the algorithm has converged to the red cluster centers and associated Voronoi tesselation.

The incorporation of probabilities (as graded memberships) sometimes improves convergence of k-means over its classical counterpart. One drawback of the method is that according to Eq. 30 the probability of "membership" of a point $\mathbf{x}_j$ in a cluster i depends implicitly on the number of clusters, and when this number is specified incorrectly, serious problems may arise (Computer exercise 4).

10.5 UNSUPERVISED BAYESIAN LEARNING

10.5.1 The Bayes Classifier

As we saw in Chapter 3, maximum-likelihood methods do not assume the parameter vector $\boldsymbol{\theta}$ to be random—it is just unknown. In such methods, prior knowledge about the likely values for $\boldsymbol{\theta}$ is not directly relevant, although in practice such knowledge may be used in choosing good starting points for hill-climbing procedures. In this section, however, we shall take a Bayesian approach to unsupervised learning. That is, we shall assume that $\boldsymbol{\theta}$ is a random variable with a known prior distribution $p(\boldsymbol{\theta})$, and we shall use the training samples to compute the posterior density $p(\boldsymbol{\theta}|\mathcal{D})$. Interestingly enough, the analysis will closely parallel the analysis of supervised Bayesian learning in Chapter 3, showing that the two problems are formally very similar.

We begin with an explicit statement of our basic assumptions. We assume the following:

1. The number c of classes is known.
2. The prior probabilities $P(\omega_j)$ for each class are known, $j = 1, \ldots, c$.
3. The forms for the class-conditional probability densities $p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)$ are known, $j = 1, \ldots, c$, but the full parameter vector $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_c)^t$ is not known.

4. Part of our knowledge about $\boldsymbol{\theta}$ is contained in a known prior density $p(\boldsymbol{\theta})$.
5. The rest of our knowledge about $\boldsymbol{\theta}$ is contained in a set $\mathcal{D}$ of n samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ drawn independently from the familiar mixture density

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{c} p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j) P(\omega_j). \tag{35}$$

At this point we could go directly to the calculation of $p(\boldsymbol{\theta}|\mathcal{D})$. However, let us first see how this density is used to determine the Bayes classifier. Suppose that a state of nature is selected with probability $P(\omega_i)$ and a feature vector $\mathbf{x}$ is selected according to the probability law $p(\mathbf{x}|\omega_i, \boldsymbol{\theta}_i)$. To derive the Bayes classifier we must use all of the information at our disposal to compute the posterior probability $P(\omega_i|\mathbf{x})$. We exhibit the role of the samples explicitly by writing this as $P(\omega_i|\mathbf{x}, \mathcal{D})$. By Bayes formula, we have

$$P(\omega_i|\mathbf{x}, \mathcal{D}) = \frac{p(\mathbf{x}|\omega_i, \mathcal{D}) P(\omega_i|\mathcal{D})}{\sum_{j=1}^{c} p(\mathbf{x}|\omega_j, \mathcal{D}) P(\omega_j|\mathcal{D})}. \tag{36}$$

Because the selection of the state of nature ω_i was done independently of the previously drawn samples, $P(\omega_i|\mathcal{D}) = P(\omega_i)$, and we obtain

$$P(\omega_i|\mathbf{x}, \mathcal{D}) = \frac{p(\mathbf{x}|\omega_i, \mathcal{D}) P(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}|\omega_j, \mathcal{D}) P(\omega_j)}. \tag{37}$$

Central to the Bayesian approach is the introduction of the unknown parameter vector $\boldsymbol{\theta}$ via

$$\begin{aligned} p(\mathbf{x}|\omega_i, \mathcal{D}) &= \int p(\mathbf{x}, \boldsymbol{\theta}|\omega_i, \mathcal{D})\, d\boldsymbol{\theta} \\ &= \int p(\mathbf{x}|\boldsymbol{\theta}, \omega_i, \mathcal{D}) p(\boldsymbol{\theta}|\omega_i, \mathcal{D})\, d\boldsymbol{\theta}. \end{aligned} \tag{38}$$

Because the selection of $\mathbf{x}$ is independent of the samples, we have $p(\mathbf{x}|\boldsymbol{\theta}, \omega_i, \mathcal{D}) = p(\mathbf{x}|\omega_i, \boldsymbol{\theta}_i)$. Similarly, because knowledge of the state of nature when $\mathbf{x}$ is selected tells us nothing about the distribution of $\boldsymbol{\theta}$, we have $p(\boldsymbol{\theta}|\omega_i, \mathcal{D}) = p(\boldsymbol{\theta}|\mathcal{D})$, and thus

$$p(\mathbf{x}|\omega_i, \mathcal{D}) = \int p(\mathbf{x}|\omega_i, \boldsymbol{\theta}_i) p(\boldsymbol{\theta}|\mathcal{D})\, d\boldsymbol{\theta}. \tag{39}$$

That is, our best estimate of $p(\mathbf{x}|\omega_i)$ is obtained by averaging $p(\mathbf{x}|\omega_i, \boldsymbol{\theta}_i)$ over $\boldsymbol{\theta}_i$. Whether or not this is a good estimate depends on the nature of $p(\boldsymbol{\theta}|\mathcal{D})$, and thus our attention turns at last to that density.

10.5.2 Learning the Parameter Vector

We can use Bayes formula to write

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta})}{\int p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta})\, d\boldsymbol{\theta}}, \tag{40}$$

where the independence of the samples yields the likelihood

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{k=1}^{n} p(\mathbf{x}_k|\boldsymbol{\theta}). \tag{41}$$

Alternatively, letting $\mathcal{D}^n$ denote the set of n samples, we can write Eq. 40 in the recursive form

$$p(\boldsymbol{\theta}|\mathcal{D}^n) = \frac{p(\mathbf{x}_n|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}^{n-1})}{\int p(\mathbf{x}_n|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}^{n-1})\,d\boldsymbol{\theta}}. \tag{42}$$

These are the basic equations for unsupervised Bayesian learning. Equation 40 emphasizes the relation between the Bayesian and the maximum-likelihood solutions. If $p(\boldsymbol{\theta})$ is essentially uniform over the region where $p(\mathcal{D}|\boldsymbol{\theta})$ peaks, then $p(\boldsymbol{\theta}|\mathcal{D})$ peaks at the same place. If the only significant peak occurs at $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$ and if the peak is very sharp, then Eqs. 37 and 39 yield

$$p(\mathbf{x}|\omega_i, \mathcal{D}) \simeq p(\mathbf{x}|\omega_i, \hat{\boldsymbol{\theta}}) \tag{43}$$

and

$$P(\omega_i|\mathbf{x}, \mathcal{D}) \simeq \frac{p(\mathbf{x}|\omega_i, \hat{\boldsymbol{\theta}}_i)P(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}|\omega_j, \hat{\boldsymbol{\theta}}_j)P(\omega_j)}. \tag{44}$$

That is, these conditions justify the use of the maximum-likelihood estimate $\hat{\boldsymbol{\theta}}$ as if it were the true value of $\boldsymbol{\theta}$ in designing the Bayes classifier.

As we saw in Chapter 3, in the limit of large amounts of data, maximum-likelihood and the Bayes methods will agree (or nearly agree). While in many *small* sample size problems they will agree, there exist small problems where the approximations are poor (Fig. 10.5). As we saw in the analogous case in supervised learning, whether one chooses to use the maximum-likelihood or the Bayes method depends not only on how confident one is of the prior distributions, but also on computational considerations; maximum-likelihood techniques are often easier to implement than Bayesian ones.

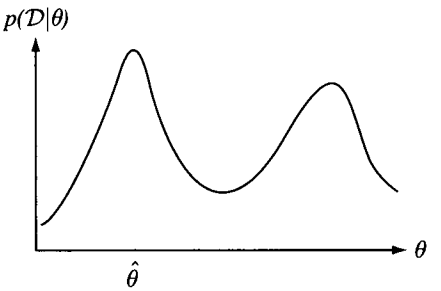


FIGURE 10.5. In a highly skewed or multiple peak posterior distribution such as illustrated here, the maximum-likelihood solution $\hat{\theta}$ will yield a density very different from a Bayesian solution, which requires the integration over the full range of parameter space θ.

Of course, if $p(\boldsymbol{\theta})$ has been obtained by supervised learning using a large set of labeled samples, it will be far from uniform, and it will have a dominant influence on $p(\boldsymbol{\theta}|\mathcal{D}^n)$ when n is small. Equation 42 shows how the observation of an additional unlabeled sample modifies our opinion about the true value of $\boldsymbol{\theta}$, and it highlights the ideas of updating and learning. If the mixture density $p(\mathbf{x}|\boldsymbol{\theta})$ is identifiable, then each additional sample tends to sharpen $p(\boldsymbol{\theta}|\mathcal{D}^n)$, and under fairly general conditions $p(\boldsymbol{\theta}|\mathcal{D}^n)$ can be shown to converge (in probability) to a Dirac delta function centered at the true value of $\boldsymbol{\theta}$ (Problem 9). Thus, even though we do not know the categories of the samples, identifiability assures us that we can learn the unknown parameter vector $\boldsymbol{\theta}$, and thereby learn the component densities $p(\mathbf{x}|\omega_i, \boldsymbol{\theta})$.

This, then, is the formal Bayesian solution to the problem of unsupervised learning. In retrospect, the fact that unsupervised learning of the parameters of a mixture density is so similar to supervised learning of the parameters of a component density is not at all surprising. Indeed, if the component density is itself a mixture, there would appear to be no essential difference between the two problems.

There are, however, some significant differences between supervised and unsupervised learning. One of the major differences concerns the issue of identifiability. With supervised learning, the lack of identifiability merely means that instead of obtaining a unique parameter vector we obtain an equivalence class of parameter vectors. Because all of these yield the same component density, lack of identifiability presents no theoretical difficulty. A lack of identifiability is much more serious in unsupervised learning. When $\boldsymbol{\theta}$ cannot be determined uniquely, the mixture cannot be decomposed into its true components. Thus, while $p(\mathbf{x}|\mathcal{D}^n)$ may still converge to $p(\mathbf{x})$, $p(\mathbf{x}|\omega_i, \mathcal{D}^n)$ given by Eq. 39 will not in general converge to $p(\mathbf{x}|\omega_i)$, and a theoretical barrier to learning exists. It is here that a few *labeled* training samples would be valuable for "decomposing" the mixture into its components.

Another serious problem for unsupervised learning is computational complexity. With supervised learning, the possibility of finding sufficient statistics allows solutions that are analytically pleasing and computationally feasible. With unsupervised learning, there is no way to avoid the fact that the samples are obtained from a mixture density,

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{c} p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j) P(\omega_j), \tag{1}$$

and this gives us little hope of ever finding simple exact solutions for $p(\boldsymbol{\theta}|\mathcal{D})$. Such solutions are tied to the existence of a simple sufficient statistic (Chapter 3), and the factorization theorem requires the ability to factor $p(\mathcal{D}|\boldsymbol{\theta})$ as

$$p(\mathcal{D}|\boldsymbol{\theta}) = g(\mathbf{s}, \boldsymbol{\theta}) h(\mathcal{D}). \tag{45}$$

But from Eqs. 41 and 1, we see that the likelihood can be written as

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{k=1}^{n} \left[\sum_{j=1}^{c} p(\mathbf{x}_k|\omega_j, \boldsymbol{\theta}_j) P(\omega_j) \right]. \tag{46}$$

Thus, $p(\mathcal{D}|\boldsymbol{\theta})$ is the sum of c^n products of component densities. Each term in this sum can be interpreted as the joint probability of obtaining the samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ bearing a particular labeling, with the sum extending over all of the ways that the samples could be labeled. Clearly, this results in a thorough mixture of $\boldsymbol{\theta}$ and the $\mathbf{x}$'s,

and no simple factoring should be expected. An exception to this statement arises if the component densities do not overlap; thus as $\boldsymbol{\theta}$ varies, only one term in the mixture density is nonzero. In that case, $p(\mathcal{D}|\boldsymbol{\theta})$ is the product of the n nonzero terms and may possess a simple sufficient statistic. However, because that case allows the class of any sample to be determined, it actually reduces the problem to one of supervised learning and thus is not a significant exception.

Another way to compare supervised and unsupervised learning is to substitute the mixture density for $p(\mathbf{x}_n|\boldsymbol{\theta})$ in Eq. 42 and obtain

$$p(\boldsymbol{\theta}|\mathcal{D}^n) = \frac{\sum_{j=1}^{c} p(\mathbf{x}_n|\omega_j, \boldsymbol{\theta}_j)P(\omega_j)}{\sum_{j=1}^{c} \int p(\mathbf{x}_n|\omega_j, \boldsymbol{\theta}_j)P(\omega_j)p(\boldsymbol{\theta}|\mathcal{D}^{n-1})\, d\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D}^{n-1}). \quad (47)$$

If we consider the special case where $P(\omega_1) = 1$ and all the other prior probabilities are zero, corresponding to the supervised case in which all samples come from Class ω_1, then Eq. 47 simplifies to

$$p(\boldsymbol{\theta}|\mathcal{D}^n) = \frac{p(\mathbf{x}_n|\omega_1, \boldsymbol{\theta}_1)}{\int p(\mathbf{x}_n|\omega_1, \boldsymbol{\theta}_1)p(\boldsymbol{\theta}|\mathcal{D}^{n-1})\, d\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D}^{n-1}). \quad (48)$$

Let us compare Eqs. 47 and 48 to see how observing an additional sample changes our estimate of $\boldsymbol{\theta}$. In each case we can ignore the normalizing denominator, which is independent of $\boldsymbol{\theta}$. Thus, the only significant difference is that in the supervised case we multiply the "prior" density for $\boldsymbol{\theta}$ by the component density $p(\mathbf{x}_n|\omega_1, \boldsymbol{\theta}_1)$, while in the unsupervised case we multiply it by the mixture density $\sum_{j=1}^{c} p(\mathbf{x}_n|\omega_j, \boldsymbol{\theta}_j)P(\omega_j)$. Assuming that the sample really did come from Class ω_1, we see that the effect of not knowing this category membership in the unsupervised case is to diminish the influence of $\mathbf{x}_n$ on changing $\boldsymbol{\theta}$. Because $\mathbf{x}_n$ could have come from any of the c classes, we cannot use it with full effectiveness in changing the component(s) of $\boldsymbol{\theta}$ associated with any one category. Rather, we must distributed its effect over the various categories in accordance with the probability that it arose from each category.

EXAMPLE 1 Unsupervised Learning of Gaussian Data

As an example, consider the one-dimensional, two-component mixture with $p(x|\omega_1) \sim N(\mu, 1)$, $p(x|\omega_2, \theta) \sim N(\theta, 1)$, where μ, $P(\omega_1)$ and $P(\omega_2)$ are known. Here we have

$$p(x|\theta) = \underbrace{\frac{P(\omega_1)}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x-\mu)^2\right]}_{\omega_1} + \underbrace{\frac{P(\omega_2)}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x-\theta)^2\right]}_{\omega_2},$$

and we seek the mean of the second component.

Viewed as a function of x, this mixture density is a superposition of two normal densities—one peaking at $x = \mu$ and the other peaking at $x = \theta$. Viewed as a function of θ, $p(x|\theta)$ has a single peak at $\theta = x$. Suppose that the prior density $p(\theta)$ is uniform from a to b. Then after one observation ($x = x_1$) we have

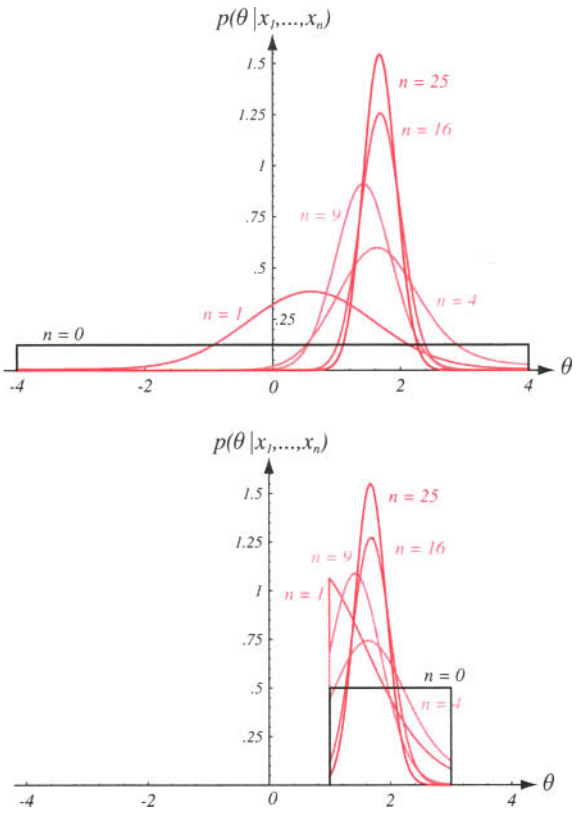
$$p(\theta|x_1) = \alpha p(x_1|\theta)p(\theta) = \begin{cases} \alpha'\{P(\omega_1)\exp[-\frac{1}{2}(x_1-\mu)^2] + \\ \qquad P(\omega_2)\exp[-\frac{1}{2}(x_1-\theta)^2]\} & a \le \theta \le b \\ 0 \qquad\qquad \text{otherwise} \end{cases},$$

where α and α' are normalizing constants that are independent of θ. If the sample x_1 is in the range $a \leq x_1 \leq b$, then $p(\theta|x_1)$ peaks at $\theta = x_1$, of course. Otherwise it peaks either at $\theta = a$ if $x_1 < a$ or at $\theta = b$ if $x_1 > b$. Note that the additive constant $\exp[-(1/2)(x_1 - \mu)^2]$ is large if x_1 is near μ, and thus the peak of $p(\theta|x_1)$ is less pronounced if x_1 is near μ. This corresponds to the fact that if x_1 is near μ, it is more likely to have come from the $p(x|\omega_1)$ component, and hence its influence on our estimate for θ is diminished.

With the addition of a second sample x_2, $p(\theta|x_1)$ changes to

$$p(\theta|x_1, x_2) = \beta p(x_2|\theta)p(\theta|x_1)$$

$$= \begin{cases} \beta' \Big\{ P(\omega_1)P(\omega_1) \exp\left[-\frac{1}{2}(x_1-\mu)^2 - \frac{1}{2}(x_2-\mu)^2\right] \\ \quad + P(\omega_1)P(\omega_2) \exp\left[-\frac{1}{2}(x_1-\mu)^2 - \frac{1}{2}(x_2-\theta)^2\right] \\ \quad + P(\omega_2)P(\omega_1) \exp\left[-\frac{1}{2}(x_1-\theta)^2 - \frac{1}{2}(x_2-\mu)^2\right] \\ \quad + P(\omega_2)P(\omega_2) \exp\left[-\frac{1}{2}(x_1-\theta)^2 - \frac{1}{2}(x_2-\theta)^2\right] \Big\} \\ \qquad\qquad\qquad\qquad\qquad\qquad a \leq \theta \leq b \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad \text{otherwise.} \end{cases}$$



In unsupervised Bayesian learning of the parameter θ, the density becomes more peaked as the number of samples increases. The top figures uses a wide uniform prior $p(\theta) = 1/8$, for $-4 \leq \theta \leq 4$ while the bottom figure uses a narrower one, $p(\theta) = 1/2$, for $1 \leq \theta \leq 3$. Despite these different prior distributions, after all 25 samples have been used, the posterior densities are virtually identical in the two cases—the information in the samples overwhelms the prior information.

Unfortunately, the primary thing we learn from this expression is that $p(\theta|\mathcal{D}^n)$ is already complicated when $n = 2$. The four terms in the sum correspond to the four ways in which the samples could have been drawn from the two component populations. With n samples there will be 2^n terms, and no simple sufficient statistics can be found to facilitate understanding or to simplify computations.

It is possible to use the relation

$$p(\theta|\mathcal{D}^n) = \frac{p(x_n|\theta)p(\theta|\mathcal{D}^{n-1})}{\int p(x_n|\theta)p(\theta|\mathcal{D}^{n-1})\,d\theta}$$

and numerical integration to obtain an approximate numerical solution for $p(\theta|\mathcal{D}^n)$. This was done for the data in Fig. 10.1 using the values $\mu = 2$, $P(\omega_1) = 1/3$, and $P(\omega_2) = 2/3$. A prior density $p(\theta)$ uniform from -4 to $+4$ encompasses the data in the table. When this was used to start the recursive computation of $p(\theta|\mathcal{D}^n)$, the results are shown in the figure were obtained. As n goes to infinity, we can confidently expect $p(\theta|\mathcal{D}^n)$ to approach an impulse centered at $\theta = 2$. This graph gives some idea of the rate of convergence.

One of the main differences between the Bayesian and the maximum-likelihood approaches to unsupervised learning is the presence of the prior density $p(\theta)$. The figure shows how $p(\theta|\mathcal{D}^n)$ changes when $p(\theta)$ is assumed to be uniform from 1 to 3, corresponding to more certain initial knowledge about θ. The results of this change are most pronounced when n is small. It is here (just as in the classification case in Chapter 3) that the differences between the Bayesian and the maximum-likelihood solutions are most significant. As n increases, the importance of prior knowledge diminishes, and in the particular case the curves for $n = 25$ are virtually identical. In general, one would expect the difference to be small when the number of unlabeled samples is several times the effective number of labeled samples used to determine $p(\theta)$.

10.5.3 Decision-Directed Approximation

Although the problem of unsupervised learning can be stated as merely the problem of estimating parameters of a mixture density, neither the maximum-likelihood nor the Bayesian approach yields analytically simple results. Exact solutions for even the simplest nontrivial examples lead to computational requirements that grow exponentially with the number of samples. The problem of unsupervised learning is too important to abandon just because exact solutions are hard to find, however, and numerous procedures for obtaining approximate solutions have been suggested.

Because the important difference between supervised and unsupervised learning is the presence or absence of labels for the samples, an obvious approach to unsupervised learning is to use the prior information to design a classifier and to use the decisions of this classifier to label the samples. This is called the *decision-directed* approach to unsupervised learning, and it is subject to many variations. It can be applied sequentially on-line by updating the classifier each time an unlabeled sample is classified. Alternatively, it can be applied in parallel (batch mode) by waiting until all n samples are classified before updating the classifier. If desired, this process can be repeated until no changes occur in the way the samples are labeled. Various heuristics can be introduced to make the extent of any corrections depend upon the confidence of the classification decision.

There are some obvious dangers associated with the decision-directed approach. If the initial classifier is not reasonably good or if an unfortunate sequence of samples is encountered, the errors in classifying the unlabeled samples can drive the classifier

the wrong way, resulting in a solution corresponding roughly to one of the lesser peaks of the likelihood function. Even if the initial classifier is optimal, in general the resulting labeling will not be the same as the true class membership; the act of classification will exclude samples from the tails of the desired distribution and will include samples from the tails of the other distributions. Thus, if there is significant overlap between the component densities, one can expect biased estimates and less-than-optimal results.

Despite these drawbacks, the simplicity of decision-directed procedures makes the Bayesian approach computationally feasible, and a flawed solution is often better than none. If conditions are favorable, performance that is nearly optimal can be achieved at far less computational expense. In practice it is found that most of these procedures work well if the parametric assumptions are valid, if there is little overlap between the component densities, and if the initial classifier design is at least roughly correct (Computer exercise 7).

10.6 DATA DESCRIPTION AND CLUSTERING

Let us reconsider our original problem of learning the structure of multidimensional patterns from a set of unlabeled samples. Viewed geometrically, these samples may form clouds of points in a d-dimensional space. Suppose that we knew, somehow, that these points came from a single normal distribution. Then the most we could learn form the data would be contained in the sufficient statistics—the sample mean and the sample covariance matrix. In essence, these statistics constitute a compact description of the data. The sample mean locates the center of gravity of the cloud; it can be thought of as the single point $\mathbf{m}$ that best represents all of the data in the sense of minimizing the sum of squared distances from $\mathbf{m}$ to the samples. The sample covariance matrix describes the amount the data scatters along various directions around $\mathbf{m}$. If the data points are actually normally distributed, then the cloud has a simple hyperellipsoidal shape, and the sample mean tends to fall in the region where the samples are most densely concentrated.

Of course, if the samples are not normally distributed these statistics can give a very misleading description of the data. Figure 10.6 shows four different data sets that all have the same mean and covariance matrix. Obviously, second-order statistics are incapable of revealing all of the structure in an arbitrary set of data.

If we assume that the samples come from a mixture of c normal distributions, we can approximate a greater variety of situations. In essence, this corresponds to assuming that the samples fall in hyperellipsoidally shaped clouds of various sizes and orientations. If the number of component densities is sufficiently high, we can approximate virtually any density function as a mixture model in this way and use the parameters of the mixture to describe the data. Alas, we have seen that the problem of estimating the parameters of a mixture density is not trivial. Furthermore, in situations where we have relatively little prior knowledge about the nature of the data, the assumption of particular parametric forms may lead to poor or meaningless results. Instead of finding structure in the data, we would be imposing structure on it.

One alternative is to use one of the nonparametric methods described in Chapter 4 to estimate the unknown mixture density. If accurate, the resulting estimate is certainly a complete description of what we can learn from the data. Regions of high local density, which might correspond to significant subclasses in the population, can be found from the peaks or modes of the estimated density.

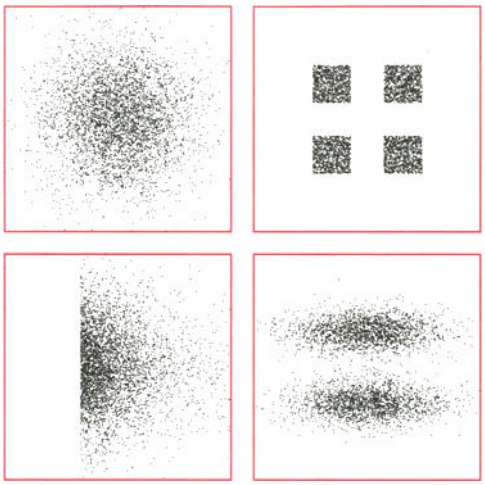


FIGURE 10.6. These four data sets have identical statistics up to second-order—that is, the same mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. In such cases it is important to include in the model more parameters to represent the structure more completely.

CLUSTERING PROCEDURE

If the goal is to find subclasses, a more direct alternative is to use a *clustering procedure*. Roughly speaking, clustering procedures yield a data description in terms of clusters or groups of data points that possess strong internal similarities. Formal clustering procedures use a criterion function, such as the sum of the squared distances from the cluster centers, and seek the grouping that extremizes the criterion function. Because even this can lead to unmanageable computational problems, other procedures have been proposed that are intuitively appealing but that lead to solutions having few, if any, established properties. Their use is usually justified on the grounds that they are easy to apply and often yield interesting results that may guide the application of more rigorous procedures.

10.6.1 Similarity Measures

Once we describe the clustering problem as one of finding natural groupings in a set of data, we are obliged to define what we mean by a natural grouping. In what sense are we to say that the samples in one cluster are more like one another than like samples in other clusters? This question actually involves two separate issues:

- How should one measure the similarity between samples?
- How should one evaluate a partitioning of a set of samples into clusters?

In this section we address the first of these issues.

The most obvious measure of the similarity (or dissimilarity) between two samples is the distance between them. One way to begin a clustering investigation is to define a suitable metric (Section 4.6) and compute the matrix of distances between all pairs of samples. If distance is a good measure of dissimilarity, then one would expect the distance between samples in the *same* cluster to be significantly less than the distance between samples in *different* clusters.

Suppose for the moment that we say that two samples belong to the same cluster if the Euclidean distance between them is less than some threshold distance d_0. It is immediately obvious that the choice of d_0 is very important. If d_0 is very large, all of

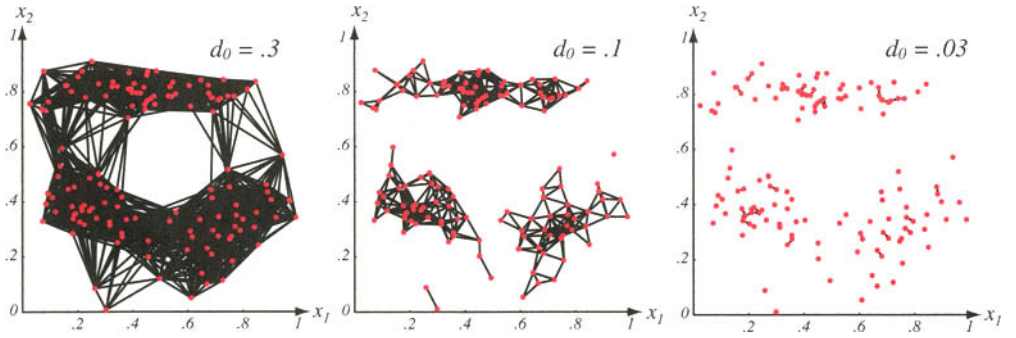


FIGURE 10.7. The distance threshold affects the number and size of clusters in similarity based clustering methods. For three different values of distance d_0, lines are drawn between points closer than d_0—the smaller the value of d_0, the smaller and more numerous the clusters.

the samples will be assigned to one cluster. If d_0 is very small, each sample will form an isolated, singleton cluster. To obtain "natural" clusters, d_0 will have to be greater than the typical within-cluster distances and less than the typical between-cluster distances (Fig. 10.7).

Less obvious perhaps is the fact that the results of clustering depend on the choice of Euclidean distance as a measure of dissimilarity. That particular choice is generally justified if the feature space is isotropic and the data are spread roughly evenly along all directions. Clusters defined by Euclidean distance will be invariant to translations or rotations in feature space—rigid-body motions of the data points. However, they will not be invariant to linear transformations in general, or to other transformations that distort the distance relationships. Thus, as Fig. 10.8 illustrates, a simple scaling of the coordinate axes can result in a different grouping of the data into clusters. Of course, this is of no concern for problems in which arbitrary rescaling is an unnatural or meaningless transformation. However, if clusters are to mean anything, they should be invariant to transformations natural to the problem.

One way to achieve invariance is to normalize the data prior to clustering. For example, to obtain invariance to displacement and scale changes, one might translate and scale the axes so that all of the features have zero mean and unit variance—standardize the data. To obtain invariance to rotation, one might rotate the axes so that they coincide with the eigenvectors of the sample covariance matrix. This transformation to *principal components* (Section 10.13.1) can be preceded or followed by normalization for scale.

However, we should not conclude that this kind of normalization is necessarily desirable. Consider, for example, the matter of translating and scaling the axes so that each feature has zero mean and unit variance. The rationale usually given for this normalization is that it prevents certain features from dominating distance calculations merely because they have large numerical values, much as we saw in networks trained with backpropagation (Chapter 6). Subtracting the mean and dividing by the standard deviation is an appropriate normalization if this spread of values is due to normal random variation; however, it can be quite inappropriate if the spread is due to the presence of subclasses (Fig. 10.9). Thus, this routine normalization may be less than helpful in the cases of greatest interest.

Instead of scaling axes, we can change the metric in interesting ways. For instance, one broad class of metrics is of the form

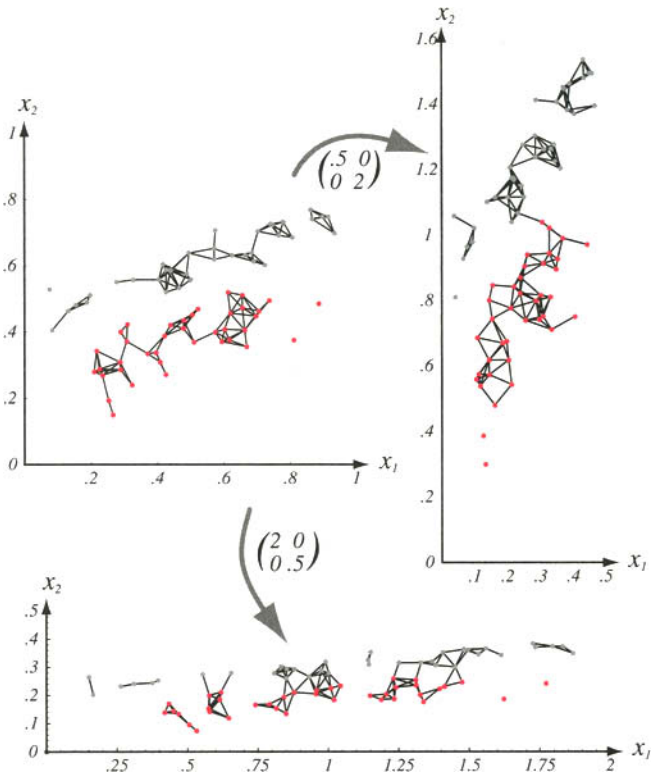


FIGURE 10.8. Scaling axes affects the clusters in a minimum distance cluster method. The original data and minimum-distance clusters are shown in the upper left; points in one cluster are shown in red, while the others are shown in gray. When the vertical axis is expanded by a factor of 2.0 and the horizontal axis shrunk by a factor of 0.5, the clustering is altered (as shown at the right). Alternatively, if the vertical axis is shrunk by a factor of 0.5 and the horizontal axis is expanded by a factor of 2.0, smaller more numerous clusters result (shown at the bottom). In both these scaled cases, the assignment of points to clusters differ from that in the original space.

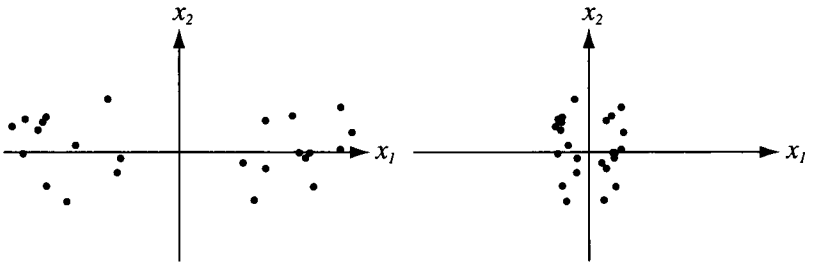


FIGURE 10.9. If the data fall into well-separated clusters (left), normalization by scaling for unit variance for the full data may reduce the separation, and hence be undesirable (right). Such a normalization may in fact be appropriate if the full data set arises from a single fundamental process (with noise), but inappropriate if there are several different processes, as shown here.

$$d(\mathbf{x}, \mathbf{x}') = \left(\sum_{k=1}^{d} |x_k - x'_k|^q \right)^{1/q}, \tag{49}$$

MINKOWSKI METRIC

CITY BLOCK METRIC

where $q \geq 1$ is a selectable parameter—the general *Minkowski metric* we considered in Chapter 4. Setting $q = 2$ gives the familiar Euclidean metric while setting $q = 1$ gives the Manhattan or *city block* metric—the sum of the absolute distances along each of the d coordinate axes. Note that only $q = 2$ is invariant to an arbitrary rotation or translation in feature space. Another alternative is to use some kind of metric based on the data itself, such as the Mahalanobis distance.

SIMILARITY FUNCTION

More generally, one can abandon the use of distance altogether and introduce a nonmetric *similarity function* $s(\mathbf{x}, \mathbf{x}')$ to compare two vectors $\mathbf{x}$ and $\mathbf{x}'$. Conventionally, this is a symmetric function whose value is large when $\mathbf{x}$ and $\mathbf{x}'$ are somehow "similar." For example, when the angle between two vectors is a meaningful measure of their similarity, then the normalized inner product

$$s(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^t \mathbf{x}'}{\|\mathbf{x}\| \, \|\mathbf{x}'\|} \tag{50}$$

may be an appropriate similarity function. This measure, which is the cosine of the angle between $\mathbf{x}$ and $\mathbf{x}'$, is invariant to rotation and dilation, though it is not invariant to translation and general linear transformations.

When the features are binary-valued (0 or 1), the similarity function of Eq. 50 has a simple nongeometrical interpretation in terms of shared features or shared attributes. Let us say that a sample $\mathbf{x}$ *possesses* the ith attribute if $x_i = 1$. Then $\mathbf{x}^t \mathbf{x}'$ is merely the number of attributes possessed by both $\mathbf{x}$ and $\mathbf{x}'$, and $\|\mathbf{x}\| \, \|\mathbf{x}'\| = (\mathbf{x}^t \mathbf{x} \mathbf{x}'^t \mathbf{x}')^{1/2}$ is the geometric mean of the number of attributes possessed by $\mathbf{x}$ and the number possessed by $\mathbf{x}'$. Thus, $s(\mathbf{x}, \mathbf{x}')$ is a measure of the relative possession of common attributes. Some simple variations are

$$s(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^t \mathbf{x}'}{d}, \tag{51}$$

the fraction of attributes shared, and

$$s(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^t \mathbf{x}'}{\mathbf{x}^t \mathbf{x} + \mathbf{x}'^t \mathbf{x}' - \mathbf{x}^t \mathbf{x}'}, \tag{52}$$

TANIMOTO DISTANCE

the ratio of the number of shared attributes to the number possessed by $\mathbf{x}$ or $\mathbf{x}'$. This latter measure (sometimes known as the Tanimoto coefficient or *Tanimoto distance*) is frequently encountered in the fields of information retrieval and biological taxonomy. Related measures of similarity arise in other applications, with the variety of measures testifying to the diversity of problem domains.

Fundamental issues in measurement theory are involved in the use of any distance or similarity function. The calculation of the similarity between two vectors always involves combining the values of their components. Yet in many pattern recognition applications the components of the feature vector measure seemingly noncomparable quantities, such as meters and kilograms. Recall our example of classifying fish: How can we compare the lightness of the skin to the length or weight of the fish? Should the comparison depend on whether the length is measured in meters or inches? How do we treat vectors whose components have a mixture of nominal, ordinal, interval and ratio scales? Ultimately, there are rarely clear methodological answers to these

questions. When a designer selects a particular similarity function or normalizes the data in a particular way, information is introduced that gives the procedure meaning. We have given examples of some alternatives that have proved to be useful. Beyond that we can do little more than alert the unwary to these pitfalls of clustering.

Amidst all this discussion of clustering, we must not lose sight of the fact that often the clusters found will later be labeled (e.g., by resorting to a teacher or small number of labeled samples) and that the clusters can then be used for classification. In that case, the same similarity (or metric) should be used for classification as was used for forming the clusters (Computer exercise 8).

10.7 CRITERION FUNCTIONS FOR CLUSTERING

We have just considered the first major issue in clustering: how to measure "similarity." Now we turn to the second major issue: the criterion function to be optimized. Suppose that we have a set $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of n samples that we want to partition into exactly c disjoint subsets $\mathcal{D}_1, \ldots, \mathcal{D}_c$. Each subset is to represent a cluster, with samples in the same cluster being somehow more similar to each other than they are to samples in other clusters. One way to make this into a well-defined problem is to define a criterion function that measures the clustering quality of any partition of the data. Then the problem is one of finding the partition that extremizes the criterion function. In this section we examine the characteristics of several basically similar criterion functions, postponing until later the question of how to find an optimal partition.

10.7.1 The Sum-of-Squared-Error Criterion

The simplest and most widely used criterion function for clustering is the sum-of-squared-error criterion. Let n_i be the number of samples in $\mathcal{D}_i$ and let $\mathbf{m}_i$ be the mean of those samples,

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x}. \tag{53}$$

Then the sum-of-squared errors is defined by

$$J_e = \sum_{i=1}^{c} \sum_{\mathbf{x} \in \mathcal{D}_i} \|\mathbf{x} - \mathbf{m}_i\|^2. \tag{54}$$

This criterion function has a simple interpretation: For a given cluster $\mathcal{D}_i$, the mean vector $\mathbf{m}_i$ is the best representative of the samples in $\mathcal{D}_i$ in the sense that it minimizes the sum of the squared lengths of the "error" vectors $\mathbf{x} - \mathbf{m}_i$ in $\mathcal{D}_i$. Thus, J_e measures the total squared error incurred in representing the n samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ by the c cluster centers $\mathbf{m}_1, \ldots, \mathbf{m}_c$. The value of J_e depends on how the samples are grouped into clusters and the number of clusters; the optimal partitioning is defined as one that minimizes J_e. Clusterings of this type are often called *minimum variance* partitions.

MINIMUM VARIANCE

What kind of clustering problems are well-suited to a sum-of-squared-error criterion? Basically, J_e is an appropriate criterion when the clusters form compact clouds that are rather well-separated from one another. A less obvious problem arises when there are great differences in the number of samples in different clusters. In that case

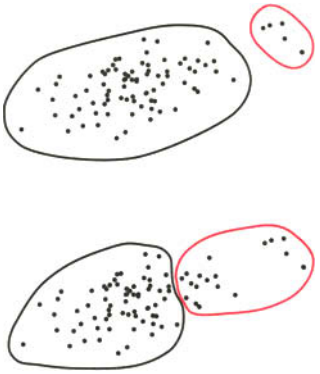


FIGURE 10.10. When two natural groupings have very different numbers of points, the clusters minimizing a sum-squared-error criterion J_e of Eq. 54 may not reveal the true underlying structure. Here the criterion is smaller for the two clusters at the bottom than for the more natural clustering at the top.

it can happen that a partition that splits a large cluster is favored over one that maintains the integrity of the natural clusters, as illustrated in Fig. 10.10. This situation frequently arises because of the presence of "outliers" or "wild shots" and brings up the problem of interpreting and evaluating the results of clustering. Because little can be said about that problem, we shall merely observe that if additional considerations render the results of minimizing J_e unsatisfactory, then these considerations should be used, if possible, in formulating a better criterion function.

10.7.2 Related Minimum Variance Criteria

By some simple algebraic manipulation (Problem 20) we can eliminate the mean vectors from the expression for J_e and obtain the equivalent expression

$$J_e = \frac{1}{2}\sum_{i=1}^{c} n_i \bar{s}_i, \tag{55}$$

where

$$\bar{s}_i = \frac{1}{n_i^2}\sum_{\mathbf{x}\in\mathcal{D}_i}\sum_{\mathbf{x}'\in\mathcal{D}_i} \|\mathbf{x}-\mathbf{x}'\|^2. \tag{56}$$

Equation 56 leads us to interpret $\bar{s}_i$ as the average squared distance between points in the ith cluster, and it emphasizes the fact that the sum-of-squared-error criterion uses Euclidean distance as the measure of similarity. It also suggests an obvious way of obtaining other criterion functions. For example, one can replace $\bar{s}_i$ by the average, the median, or perhaps the maximum distance between points in a cluster. More generally, one can introduce an appropriate similarity function $s(\mathbf{x}, \mathbf{x}')$ and replace $\bar{s}_i$ by functions such as

$$\bar{s}_i = \frac{1}{n_i^2}\sum_{\mathbf{x}\in\mathcal{D}_i}\sum_{\mathbf{x}'\in\mathcal{D}_i} s(\mathbf{x}, \mathbf{x}') \tag{57}$$

or

$$\bar{s}_i = \min_{\mathbf{x},\mathbf{x}' \in \mathcal{D}_i} s(\mathbf{x}, \mathbf{x}'). \tag{58}$$

As in Chapter 4, we define an optimal partition as one that extremizes the criterion function. This creates a well-defined problem, and the hope is that its solution discloses the intrinsic structure of the data.

10.7.3 Scatter Criteria

The Scatter Matrices

Another interesting class of criterion functions can be derived from the scatter matrices used in multiple discriminant analysis. The definitions in Table 10.1 directly parallel those given in Chapter 3.

Table 10.1. Mean Vectors and Scatter Matrices Used in Clustering Criteria

	Depend on cluster center? Yes	No	
Mean vector for the ith cluster		×	$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x}$ (59)
Total mean vector		×	$\mathbf{m} = \frac{1}{n} \sum_{\mathcal{D}} \mathbf{x} = \frac{1}{n} \sum_{i=1}^{c} n_i \mathbf{m}_i$ (60)
Scatter matrix for the ith cluster	×		$\mathbf{S}_i = \sum_{\mathbf{x} \in \mathcal{D}_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^t$ (61)
Within-cluster scatter matrix	×		$\mathbf{S}_W = \sum_{i=1}^{c} \mathbf{S}_i$ (62)
Between-cluster scatter matrix	×		$\mathbf{S}_B = \sum_{i=1}^{c} n_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^t$ (63)
Total scatter matrix		×	$\mathbf{S}_T = \sum_{\mathbf{x} \in \mathcal{D}} (\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^t$ (64)

As before, it follows from the definitions in Table 10.1 that the total scatter matrix is the sum of the within-cluster scatter matrix and the between-cluster scatter matrix:

$$\mathbf{S}_T = \mathbf{S}_W + \mathbf{S}_B. \tag{65}$$

Note that the total scatter matrix does not depend on how the set of samples is partitioned into clusters; it depends only on the total set of samples. The within-cluster

and between-cluster scatter matrices taken separately do depend on the partitioning, of course. Roughly speaking, there is an exchange between these two matrices: The between-cluster scatter goes up as the within-cluster scatter goes down. This is fortunate, because by trying to minimize the within-cluster scatter we will also tend to maximize the between-cluster scatter.

To be more precise in talking about the amount of within-cluster or between-cluster scatter, we need a scalar measure of the "size" of a scatter matrix. The two measures that we shall consider are the *trace* and the *determinant*. In the univariate case, these two measures are equivalent, and we can define an optimal partition as one that minimizes $\mathbf{S}_W$ or maximizes $\mathbf{S}_B$. In the multivariate case, things are somewhat more complicated, and a number of related but distinct optimality criteria have been suggested.

The Trace Criterion

Perhaps the simplest scalar measure of a scatter matrix is its trace—the sum of its diagonal elements. Roughly speaking, the trace measures the square of the scattering radius, because it is proportional to the sum of the variances in the coordinate directions. Thus, an obvious criterion function to minimize is the trace of $\mathbf{S}_W$. In fact, this criterion is nothing more or less than the sum-of-squared-error criterion, because the definitions of scatter matrices (Eqs. 61 and 62) yield

$$\mathrm{tr}[\mathbf{S}_W] = \sum_{i=1}^{c} \mathrm{tr}[\mathbf{S}_i] = \sum_{i=1}^{c} \sum_{\mathbf{x}\in\mathcal{D}_i} \|\mathbf{x} - \mathbf{m}_i\|^2 = J_e. \tag{66}$$

Because $\mathrm{tr}[\mathbf{S}_T] = \mathrm{tr}[\mathbf{S}_W] + \mathrm{tr}[\mathbf{S}_B]$ and $\mathrm{tr}[\mathbf{S}_T]$ is independent of how the samples are partitioned, we see that no new results are obtained by trying to maximize $\mathrm{tr}[\mathbf{S}_B]$. However, it is comforting to know that in seeking to minimize the within-cluster criterion $J_e = \mathrm{tr}[\mathbf{S}_W]$ we are also maximizing the between-cluster criterion

$$\mathrm{tr}[S_B] = \sum_{i=1}^{c} n_i \|\mathbf{m}_i - \mathbf{m}\|^2. \tag{67}$$

The Determinant Criterion

We have used the determinant of the scatter matrix to obtain a scalar measure of scatter. Roughly speaking, the determinant measures the square of the scattering volume, since it is proportional to the product of the variances in the directions of the principal axes. Because $\mathbf{S}_B$ will be singular if the number of clusters is less than or equal to the dimensionality, $|\mathbf{S}_B|$ is obviously a poor choice for a criterion function. Furthermore, $\mathbf{S}_B$ may become singular, and will certainly be so if $n - c$ is less than the dimensionality d (Problem 29). However, if we assume that $\mathbf{S}_W$ is nonsingular, we are led to consider the determinant criterion function

$$J_d = |\mathbf{S}_W| = \left| \sum_{i=1}^{c} \mathbf{S}_i \right|. \tag{68}$$

The partition that minimizes J_d is often similar to the one that minimizes J_e, but the two need not be the same, as shown in Example 2. We observed before that the minimum-squared-error partition might change if the axes are scaled, though this

does not happen with J_d (Problem 27). Thus J_d is to be favored under conditions where there may be unknown or irrelevant linear transformations of the data.

Invariant Criteria

It is not particularly hard to show that the eigenvalues $\lambda_1, \ldots, \lambda_d$ of $\mathbf{S}_W^{-1}\mathbf{S}_B$ are invariant under nonsingular linear transformations of the data (Problem 28). Indeed, these eigenvalues are the basic linear invariants of the scatter matrices. Their numerical values measure the ratio of between-cluster to within-cluster scatter in the direction of the eigenvectors, and partitions that yield large values are usually desirable. Of course, as we have seen before, the fact that the rank of $\mathbf{S}_B$ can not exceed $c - 1$ means that no more than $c - 1$ of these eigenvalues can be nonzero. Nevertheless, good partitions are ones for which the nonzero eigenvalues are large.

One can invent a great variety of invariant clustering criteria by composing appropriate functions of these eigenvalues. Some of these follow naturally from standard matrix operations. For example, because the trace of a matrix is the sum of its eigenvalues, we might elect to maximize the criterion function

$$\mathrm{tr}[\mathbf{S}_W^{-1}\mathbf{S}_B] = \sum_{i=1}^{d} \lambda_i. \tag{69}$$

By using the relation $\mathbf{S}_T = \mathbf{S}_W + \mathbf{S}_B$, we can derive the following invariant relatives of $\mathrm{tr}[\mathbf{S}_W]$ and $|\mathbf{S}_W|$, as asked in Problem 26:

$$J_f = \mathrm{tr}[\mathbf{S}_T^{-1}\mathbf{S}_W] = \sum_{i=1}^{d} \frac{1}{1+\lambda_i} \tag{70}$$

and

$$\frac{|\mathbf{S}_W|}{|\mathbf{S}_T|} = \prod_{i=1}^{d} \frac{1}{1+\lambda_i}. \tag{71}$$

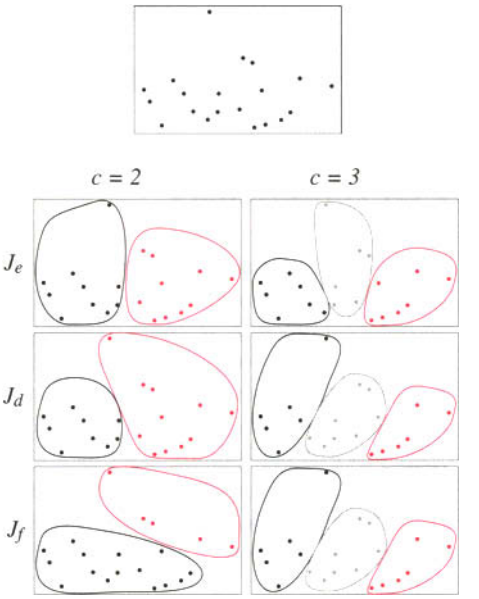
Because all of these criterion functions are invariant to linear transformations, the same is true of the partitions that extremize them. In the special case of two clusters, only one eigenvalue is nonzero, and all of these criteria yield the same clustering. However, when the samples are partitioned into more than two clusters, the optimal partitions, though often similar, need not be the same, as shown in Example 2.

EXAMPLE 2 Clustering Criteria

We can gain some intuition by considering these clustering criteria applied to the below data set. All of the clusterings seem reasonable, and there is no strong argument to favor one over the others. For the case $c = 2$, the clusters minimizing the J_e indeed tend to favor clusters of roughly equal numbers of points; in contrast, J_d favors one large and one fairly small cluster. Because the full data set happens to be spread horizontally more than vertically, the eigenvalue in the horizontal direction is greater than that in the vertical direction. As such, the clusters are "stretched" horizontally somewhat. In general, the differences between

Sample	x_1	x_2
1	−1.82	0.24
2	−0.38	−0.39
3	−0.13	0.16
4	−1.17	0.44
5	−0.92	0.16
6	−1.69	−0.01
7	0.33	−0.17
8	−0.71	−0.21
9	1.27	−0.39
10	−0.16	−0.23

Sample	x_1	x_2
11	0.41	0.91
12	1.70	0.48
13	0.92	−0.49
14	2.41	0.32
15	1.48	−0.23
16	−0.34	1.88
17	0.83	0.23
18	0.62	0.81
19	−1.42	−0.51
20	0.67	−0.55



The raw data shown at the top does not exhibit any obvious clusters. The clusters found by minimizing a criterion depends upon the criterion function as well as the assumed number of clusters. The sum-of-squared-error criterion J_e (Eq. 54), the determinant criterion J_d (Eq. 68) and the more subtle trace criterion J_f (Eq. 70) were applied to the 20 points in the table with the assumption of $c = 2$ and $c = 3$ clusters. (Each point in the table is shown, with bounding boxes defined by $-1.8 < x_1 < 2.5$ and $-0.6 < x_2 < 1.9$.)

the cluster criteria become less pronounced for large numbers of clusters. For the $c = 3$ case, for instance, the clusters depend only mildly upon the cluster criterion—indeed, two of the clusterings are identical.

With regard to the criterion function involving $\mathbf{S}_T$, note that $\mathbf{S}_T$ does not depend on how the samples are partitioned into clusters. Thus, the clusterings that minimize $|\mathbf{S}_W|/|\mathbf{S}_T|$ are exactly the same as the ones that minimize $|\mathbf{S}_W|$. If we rotate and scale the axes so that $\mathbf{S}_T$ becomes the identity matrix, we see that minimizing $\mathrm{tr}[\mathbf{S}_T^{-1}\mathbf{S}_W]$ is equivalent to minimizing the sum-of-squared-error criterion $\mathrm{tr}[\mathbf{S}_W]$ after performing this normalization. Clearly, this criterion suffers from the very defects that we warned about in Section 10.6.1, and it is probably the least desirable of these criteria.

One final warning about invariant criteria is in order. If different apparent clusters can be obtained by scaling the axes or by applying any other linear transformation, then all of these groupings will be exposed by invariant procedures. Thus, invariant criterion functions are more likely to possess multiple local extrema and are correspondingly more difficult to optimize.

The variety of the criterion functions we have discussed and the somewhat subtle differences between them should not be allowed to obscure their essential similarity. In every case the underlying model is that the samples form c fairly well separated clouds of points. The within-cluster scatter matrix $\mathbf{S}_W$ is used to measure the compactness of these clouds, and the basic goal is to find the most compact grouping. While this approach has proved useful for many problems, it is not universally applicable. For example, it will not extract a very dense cluster embedded in the center of a diffuse cluster, or separate intertwined line-like clusters. When the minimum of the cluster criterion is not sufficiently low, and the cluster structure is not inferred by the algorithm, we must devise other criterion functions that are better matched to the structure present or being sought.

*10.8 ITERATIVE OPTIMIZATION

Once a criterion function has been selected, clustering becomes a well-defined problem in discrete optimization: Find those partitions of the set of samples that extremize the criterion function. Because the sample set is finite, there are only a finite number of possible partitions. Thus, in theory the clustering problem can always be solved by exhaustive enumeration. However, the computational complexity renders such an approach unthinkable for all but the simplest problems; there are approximately $c^n/c!$ ways of partitioning a set of n elements into c subsets, and this exponential growth with n is overwhelming (Problem 18). For example, an exhaustive search for the best set of 5 clusters in 100 samples would require considering more than 10^{67} partitionings. Simply put, in most applications an exhaustive search is completely infeasible.

The approach most frequently used in seeking optimal partitions is iterative optimization. The basic idea is to find some reasonable initial partition and to "move" samples from one group to another if such a move will improve the value of the criterion function. Like hill-climbing procedures in general, these approaches guarantee local but not global optimization. Different starting points can lead to different solutions, and one never knows whether or not the best solution has been found. Despite these limitations, the fact that the computational requirements are bearable makes this approach attractive.

Let us consider the use of iterative improvement to minimize the sum-of-squared-error criterion J_e, written as

$$J_e = \sum_{i=1}^{c} J_i, \tag{72}$$

where an effective error per cluster is defined to be

$$J_i = \sum_{\mathbf{x}\in\mathcal{D}_i} \|\mathbf{x} - \mathbf{m}_i\|^2 \tag{73}$$

and the mean of each cluster is, as before,

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x}. \tag{53}$$

Suppose that a sample $\hat{\mathbf{x}}$ currently in cluster $\mathcal{D}_i$ is tentatively moved to $\mathcal{D}_j$. Then $\mathbf{m}_j$ changes to

$$\mathbf{m}_j^* = \mathbf{m}_j + \frac{\hat{\mathbf{x}} - \mathbf{m}_j}{n_j + 1} \tag{74}$$

and J_j increases to

$$\begin{aligned} J_j^* &= \sum_{\mathbf{x} \in \mathcal{D}_j} \|\mathbf{x} - \mathbf{m}_j^*\|^2 + \|\hat{\mathbf{x}} - \mathbf{m}_j^*\|^2 \\ &= \left(\sum_{\mathbf{x} \in \mathcal{D}_i} \left\| \mathbf{x} - \mathbf{m}_j - \frac{\hat{\mathbf{x}} - \mathbf{m}_j}{n_j + 1} \right\|^2 \right) + \left\| \frac{n_j}{n_j + 1} (\hat{\mathbf{x}} - \mathbf{m}_j) \right\|^2 \\ &= J_j + \frac{n_j}{n_j + 1} \|\hat{\mathbf{x}} - \mathbf{m}_j\|^2. \end{aligned} \tag{75}$$

Under the assumption that $n_i \neq 1$ (singleton clusters should not be destroyed), a similar calculation (Problem 31) shows that $\mathbf{m}_i$ changes to

$$\mathbf{m}_i^* = \mathbf{m}_i - \frac{\hat{\mathbf{x}} - \mathbf{m}_i}{n_i - 1} \tag{76}$$

and J_i decreases to

$$J_i^* = J_i - \frac{n_i}{n_i - 1} \|\hat{\mathbf{x}} - \mathbf{m}_i\|^2. \tag{77}$$

These equations greatly simplify the computation of the change in the criterion function. The transfer of $\hat{\mathbf{x}}$ from $\mathcal{D}_i$ to $\mathcal{D}_j$ is advantageous if the decrease in J_i is greater than the increase in J_j. This is the case if

$$\frac{n_i}{n_i - 1} \|\hat{\mathbf{x}} - \mathbf{m}_i\|^2 > \frac{n_j}{n_j + 1} \|\hat{\mathbf{x}} - \mathbf{m}_j\|^2, \tag{78}$$

which typically happens whenever $\hat{\mathbf{x}}$ is closer to $\mathbf{m}_j$ than $\mathbf{m}_i$. If reassignment is profitable, the greatest decrease in sum of squared error is obtained by selecting the cluster for which $n_j \|\hat{\mathbf{x}} - \mathbf{m}_j\|^2 / (n_j + 1)$ is minimum. This leads to the following clustering procedure:

■ **Algorithm 3. (Basic Iterative Minimum-Squared-Error Clustering)**

```
1 begin initialize n, c, m_1, m_2, ..., m_c
2       do randomly select a sample x̂
3          i ← arg min_{i'} ||m_{i'} − x̂||      (classify x̂)
```

```
4        if n_i ≠ 1 then compute
5                ρ_j = { (n_j/(n_j+1)) ||x̂ − m_j||^2   j ≠ i
                       { (n_j/(n_j−1)) ||x̂ − m_i||^2   j = i
6                if ρ_k ≤ ρ_j for all j then transfer x̂ to D_k
7                   recompute J_e, m_i, m_k
8           until no change in J_e in n attempts
9        return m_1, m_2, ..., m_c
10   end
```

A moment's consideration will show that this procedure is essentially a sequential version of the k-means procedure (Algorithm 1) described in Section 10.4.3. Where the k-means procedure waits until all n samples have been reclassified before updating, the Basic Iterative Minimum-Squared-Error procedure updates after each sample is reclassified. It has been experimentally observed that this procedure is more susceptible to being trapped in local minima, and it has the further disadvantage of making the results depend on the order in which the candidates are selected. However, it is at least a stepwise optimal procedure, and it can be easily modified to apply to problems in which samples are acquired sequentially and clustering must be done on-line.

One question that plagues all hill-climbing procedures is the choice of the starting point. Unfortunately, there is no simple, universally good solution to this problem. One approach is to select c samples randomly for the initial cluster centers, using them to partition the data on a minimum-distance basis. Repetition with different random selections can give some indication of the sensitivity of the solution to the starting point. Yet another approach is to find the c-cluster starting point from the solutions to the $(c-1)$-cluster problem. The solution for the one-cluster problem is the total sample mean; the starting point for the c-cluster problem can be the final means for the $(c-1)$-cluster problem plus the sample that is farthest from the nearest cluster center. This approach leads us directly to the so-called hierarchical clustering procedures, which are simple methods that can provide very good starting points for iterative optimization.

10.9 HIERARCHICAL CLUSTERING

Up to now, our methods have formed disjoint clusters—in computer science terminology, we would say that the data description is "flat." However, there are many times when clusters have subclusters, these have subsubclusters, and so on. In biological taxonomy, for instance, kingdoms are split into phyla, which are split into subphyla, which are split into orders, suborders, families, subfamilies, genus and species, and so on, all the way to a particular individual organism. Thus we might have `kingdom = animal`, `phylum = Chordata`, `subphylum = Vertebrata`, `class = Osteichthyes`, `subclass = Actinopterygii`, `order = Salmoniformes`, `family = Salmonidae`, `genus = Oncorhynchus`, `species = Oncorhynchus kisutch`, and finally the `individual` being the particular Coho salmon caught in our net. Organisms that lie in the animal kingdom—such as a salmon and a moose—share important attributes that are not present in organisms in the plant kingdom, such as redwood trees. In fact, this kind of hierarchical clustering

permeates classifactory activities in the sciences. Thus we now turn to clustering methods which will lead to representations that are "hierarchical," rather than flat.

10.9.1 Definitions

Let us consider a sequence of partitions of the n samples into c clusters. The first of these is a partition into n clusters, each cluster containing exactly one sample. The next is a partition into $n - 1$ clusters, the next a partition into $n - 2$, and so on until the nth, in which all the samples form one cluster. We shall say that we are at level k in the sequence when $c = n - k + 1$. Thus, level one corresponds to n clusters and level n corresponds to one cluster. Given any two samples $\mathbf{x}$ and $\mathbf{x}'$, at *some* level they will be grouped together in the same cluster. If the sequence has the property that whenever two samples are in the same cluster at level k they remain together at all higher levels, then the sequence is said to be a *hierarchical clustering*.

DENDROGRAM

The most natural representation of hierarchical clustering is a corresponding tree, called a *dendrogram*, which shows how the samples are grouped. Figure 10.11 shows a dendrogram for a simple problem involving eight samples. Level $k = 1$ shows the eight samples as singleton clusters. At level 2, samples $\mathbf{x}_6$ and $\mathbf{x}_7$ have been grouped to form a cluster, and they stay together at all subsequent levels. If it is possible to measure the similarity between clusters, then the dendrogram is usually drawn to scale to show the similarity between the clusters that are grouped. In Fig. 10.11, for example, the similarity between the two groups of samples that are merged at level 5 has a value of roughly 60.

We shall see shortly how such similarity values can be obtained, but first note that the similarity values can be used to help determine whether groupings are natural or forced. If the similarity values for the levels are roughly evenly distributed throughout the range of possible values, then there is no principled argument that any particular number of clusters is better or "more natural" than another. Conversely, suppose that there is a unusually large gap between the similarity values for the levels corresponding to $c = 3$ and to $c = 4$ clusters. In such a case, one can argue that $c = 3$ is the most natural number of clusters (Problem 37).

Another representation for hierarchical clustering is based on sets, in which each level of cluster may contain sets that are subclusters, as shown in Fig. 10.12.

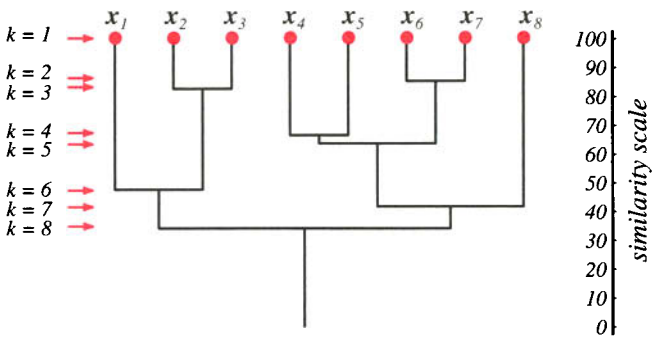


FIGURE 10.11. A dendrogram can represent the results of hierarchical clustering algorithms. The vertical axis shows a generalized measure of similarity among clusters. Here, at level 1 all eight points lie in singleton clusters; each point in a cluster is highly similar to itself, of course. Points $\mathbf{x}_6$ and $\mathbf{x}_7$ happen to be the most similar, and are merged at level 2, and so forth.

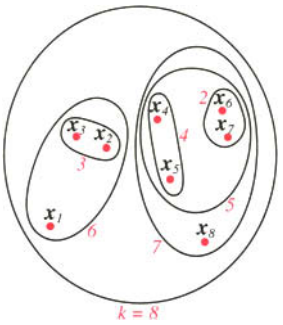


FIGURE 10.12. A set or Venn diagram representation of two-dimensional data (which was used in the dendrogram of Fig. 10.11) reveals the hierarchical structure but not the quantitative distances between clusters. The levels are numbered by k, in red.

Yet another, textual, representation uses brackets, such as $\{\{\mathbf{x}_1, \{\mathbf{x}_2, \mathbf{x}_3\}\}, \{\{\mathbf{x}_4, \mathbf{x}_5\}, \{\mathbf{x}_6, \mathbf{x}_7\}\}, \mathbf{x}_8\}\}$. While such representations may reveal the hierarchical structure of the data, they do not naturally represent the similarities *quantitatively*. For this reason, dendrograms are generally preferred.

Because of their conceptual simplicity, hierarchical clustering procedures are among the best known of unsupervised methods. The procedures themselves can be divided according to two distinct approaches: agglomerative and divisive. *Agglomerative* (bottom-up, clumping) procedures start with n singleton clusters and form the sequence by successively merging clusters. *Divisive* (top-down, splitting) procedures start with all of the samples in one cluster and form the sequence by successively splitting clusters. The computation needed to go from one level to another is usually simpler for the agglomerative procedures. However, when there are many samples and one is interested in only a small number of clusters, this computation will have to be repeated many times. For simplicity, we shall concentrate on agglomerative procedures, and we shall merely touch on some divisive methods in Section 10.12.

AGGLOMERATIVE

DIVISIVE

10.9.2 Agglomerative Hierarchical Clustering

The major steps in agglomerative clustering are contained in the following procedure, where c is the desired number of final clusters:

■ **Algorithm 4. (Agglomerative Hierarchical Clustering)**

```
1 begin initialize c, ĉ ← n, D_i ← {x_i}, i = 1, ..., n
2         do ĉ ← ĉ − 1
3             find nearest clusters, say, D_i and D_j
4             merge D_i and D_j
5         until c = ĉ
6   return c clusters
7 end
```

As described, this procedure terminates when the specified number of clusters has been obtained and returns the clusters, described as set of points (rather than as mean or representative vectors). If we continue until $c = 1$ we can produce a dendrogram like that in Fig. 10.11. At any level the "distance" between nearest clusters can pro-

vide the dissimilarity value for that level. Note that we have not said how to measure the distance between two clusters, and hence how to find the "nearest" clusters, required by line 3 of Algorithm 4. The considerations here are much like those involved in selecting a general clustering criterion function. For simplicity, we shall generally restrict our attention to the following distance measures:

$$d_{min}(\mathcal{D}_i, \mathcal{D}_j) = \min_{\substack{\mathbf{x}\in\mathcal{D}_i \\ \mathbf{x}'\in\mathcal{D}_j}} \|\mathbf{x} - \mathbf{x}'\| \tag{79}$$

$$d_{max}(\mathcal{D}_i, \mathcal{D}_j) = \max_{\substack{\mathbf{x}\in\mathcal{D}_i \\ \mathbf{x}'\in\mathcal{D}_j}} \|\mathbf{x} - \mathbf{x}'\| \tag{80}$$

$$d_{avg}(\mathcal{D}_i, \mathcal{D}_j) = \frac{1}{n_i n_j} \sum_{\mathbf{x}\in\mathcal{D}_i} \sum_{\mathbf{x}'\in\mathcal{D}_j} \|\mathbf{x} - \mathbf{x}'\| \tag{81}$$

$$d_{mean}(\mathcal{D}_i, \mathcal{D}_j) = \|\mathbf{m}_i - \mathbf{m}_j\|. \tag{82}$$

All of these measures have a minimum-variance flavor, and they usually yield the same results if the clusters are compact and well separated. However, if the clusters are close to one another, or if their shapes are not basically hyperspherical, quite different results can be obtained. Below we shall illustrate some of the differences.

But first let us consider the computational complexity of a particularly simple agglomerative clustering algorithm. Suppose we have n patterns in d-dimensional space, and we seek to form c clusters using $d_{min}(\mathcal{D}_i, \mathcal{D}_j)$ defined in Eq. 79. We will, once and for all, need to calculate $n(n-1)$ interpoint distances—each of which is an $O(d)$ calculation—and place the results in an interpoint distance table. The space complexity is, then, $O(n^2)$. Finding the minimum distance pair (for the first merging) requires that we step through the complete list, keeping the index of the smallest distance. Thus for the first agglomerative step, the complexity is $O(n(n-1)(d+1)) = O(n^2 d)$. For an arbitrary agglomeration step (i.e., from $\hat{c}$ to $\hat{c}-1$), we need merely step through the $n(n-1) - \hat{c}$ "unused" distances in the list and find the smallest for which $\mathbf{x}$ and $\mathbf{x}'$ lie in different clusters. This is, again, $O(n(n-1) - \hat{c})$. The full time complexity is thus $O(cn^2 d)$, and in typical conditions $n \gg c$.*

The Nearest-Neighbor Algorithm

MINIMUM ALGORITHM
SINGLE-LINKAGE ALGORITHM

When $d_{min}(\cdot,\cdot)$ is used to measure the distance between clusters (Eq. 79) the algorithm is sometimes called the nearest-neighbor cluster algorithm, or *minimum algorithm* Moreover, if it is terminated when the distance between nearest clusters exceeds an arbitrary threshold, it is called the *single-linkage algorithm*. Suppose that we think of the data points as being nodes of a graph, with edges forming a path between the nodes in the same subset $\mathcal{D}_i$. When $d_{min}(\cdot,\cdot)$ is used to measure the distance between subsets, the nearest-neighbor nodes determine the nearest subsets. The merging of $\mathcal{D}_i$ and $\mathcal{D}_j$ corresponds to adding an edge between the nearest pair of nodes in $\mathcal{D}_i$ and $\mathcal{D}_j$. Because edges linking clusters always go between distinct clusters, the resulting graph never has any closed loops or circuits; in the terminology of graph theory, this procedure generates a *tree*. If it is allowed to continue until all of the subsets are linked, the result is a *spanning tree*—a tree with a path from any

SPANNING TREE

*There are methods for sorting or arranging the entries in the interpoint distance table so as to easily avoid inspection of points in the same cluster, but these typically do not improve the complexity results significantly.

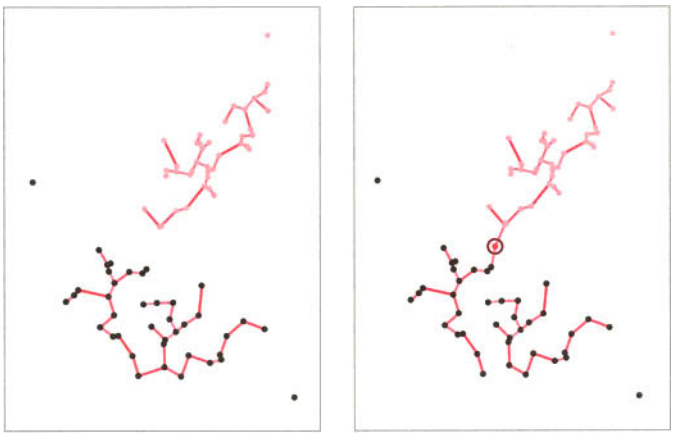


FIGURE 10.13. Two Gaussians were used to generate two-dimensional samples, shown in pink and black. The nearest-neighbor clustering algorithm gives two clusters that well approximate the generating Gaussians (left). If, however, another particular sample is generated (circled red point at the right) and the procedure is restarted, the clusters do not well approximate the Gaussians. This illustrates how the algorithm is sensitive to the details of the samples.

node to any other node. Moreover, it can be shown that the sum of the edge lengths of the resulting tree will not exceed the sum of the edge lengths for any other spanning tree for that set of samples (Problem 39). Thus, with the use of $d_{min}(\cdot, \cdot)$ as the distance measure, the agglomerative clustering procedure becomes an algorithm for generating a *minimal spanning tree*.

Figure 10.13 shows the results of applying this procedure to Gaussian data. In both cases the procedure was stopped, giving two large clusters (plus three singleton outliers); a minimal spanning tree can be obtained by adding the shortest possible edge between the two clusters. In the first case where the clusters are fairly well separated, the obvious clusters are found. In the second case, the presence of a point located so as to produce a bridge between the clusters results in a rather unexpected grouping into one large, elongated cluster and one small, compact cluster. This behavior is often called the "chaining effect" and is sometimes considered to be a defect of this distance measure. To the extent that the results are very sensitive to noise or to slight changes in position of the data points, this is certainly a valid criticism.

The Farthest-Neighbor Algorithm

When $d_{max}(\cdot, \cdot)$ of Eq. 80 is used to measure the distance between subsets, the algorithm is sometimes called the farthest-neighbor clustering algorithm, or *maximum algorithm*. If it is terminated when the distance between nearest clusters exceeds an arbitrary threshold, it is called the *complete-linkage algorithm*. The farthest-neighbor algorithm discourages the growth of elongated clusters. Application of the procedure can be thought of as producing a graph in which edges connect all of the nodes in a cluster. In the terminology of graph theory, every cluster constitutes a *complete* subgraph. The distance between two clusters is determined by the most distant nodes in the two clusters. When the nearest clusters are merged, the graph is changed by adding edges between every pair of nodes in the two clusters.

MAXIMUM ALGORITHM
COMPLETE-LINKAGE ALGORITHM
COMPLETE SUBGRAPH

If we define the diameter of a partition as the largest diameter for clusters in the partition, then each iteration increases the diameter of the partition as little as possi-

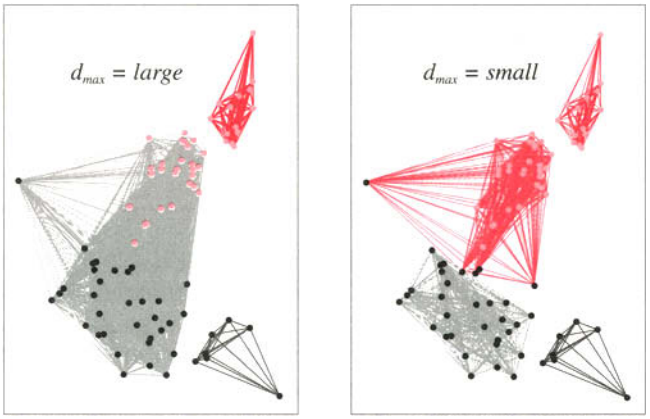


FIGURE 10.14. The farthest-neighbor clustering algorithm uses the separation between the most distant points as a criterion for cluster membership. If this distance is set very large, then all points lie in the same cluster. In the case shown at the left, a fairly large d_{max} leads to three clusters; a smaller d_{max} gives four clusters, as shown at the right.

ble. As Fig. 10.14 illustrates, this is advantageous when the true clusters are compact and roughly equal in size. Nevertheless, when this is not the case—as happens with the two elongated clusters—the resulting groupings can be meaningless. This is another example of imposing structure on data rather than finding structure in it.

Compromises

The minimum and maximum measures represent two extremes in measuring the distance between clusters. Like all procedures that involve minima or maxima, they tend to be overly sensitive to "outliers" or "wildshots." The use of averaging is an obvious way to ameliorate these problems, and $d_{avg}(\cdot,\cdot)$ and $d_{mean}(\cdot,\cdot)$ in Eqs. 81 and 82 are natural compromises between $d_{min}(\cdot,\cdot)$ and $d_{max}(\cdot,\cdot)$. Computationally, $d_{mean}(\cdot,\cdot)$ is the simplest of all of these measures, because the others require computing all $n_i n_j$ pairs of distances $\|\mathbf{x}-\mathbf{x}'\|$. However, a measure such as $d_{avg}(\cdot,\cdot)$ can be used when the distances $\|\mathbf{x}-\mathbf{x}'\|$ are replaced by similarity measures, where the similarity between mean vectors may be difficult or impossible to define.

10.9.3 Stepwise-Optimal Hierarchical Clustering

We observed earlier that if clusters are grown by merging the nearest pair of clusters, then the results have a minimum variance flavor. However, when the measure of distance between clusters is chosen arbitrarily, we can rarely assert that the resulting partition extremizes any particular criterion function. In effect, hierarchical clustering defines a cluster as whatever results from applying the clustering procedure. Nevertheless, with a simple modification it is possible to obtain a stepwise-optimal procedure for extremizing a criterion function. This is done merely by replacing line 3 of the Basic Iterative Agglomerative Clustering Procedure (Algorithm 4) by a more general form to get the following algorithm:

■ **Algorithm 5. (Stepwise Optimal Hierarchical Clustering)**

1 **begin** **initialize** $c, \hat{c} \leftarrow n, \mathcal{D}_i \leftarrow \{\mathbf{x}_i\}, i = 1, \ldots, n$
2 **do** $\hat{c} \leftarrow \hat{c} - 1$

```
3            find clusters whose merger changes the criterion the least,
             say, D_i and D_j
4            merge D_i and D_j
5        until c = ĉ
6    return c clusters
7 end
```

We saw earlier that the use of $d_{max}(\cdot,\cdot)$ causes the smallest possible stepwise increase in the diameter of the partition. Another simple example is provided by the sum-of-squared-error criterion function J_e. By an analysis very similar to that used in Section 10.8, we find that the pair of clusters whose merger increases J_e as little as possible is the pair for which the "distance"

$$d_e(\mathcal{D}_i, \mathcal{D}_j) = \sqrt{\frac{n_i n_j}{n_i + n_j}} \|\mathbf{m}_i - \mathbf{m}_j\| \tag{83}$$

is minimum (Problem 36). Thus, in selecting clusters to be merged, this criterion takes into account the number of samples in each cluster as well as the distance between clusters. In general, the use of $d_e(\cdot,\cdot)$ tends to favor growth by merging singletons or small clusters with large clusters over merging medium-sized clusters. While the final partition may not minimize J_e, it usually provides a very good starting point for further iterative optimization.

10.9.4 Hierarchical Clustering and Induced Metrics

DISSIMILARITY

Suppose that we are unable to supply a metric for our data, but that we can measure a *dissimilarity* value $\delta(\mathbf{x}, \mathbf{x}')$ for every pair of samples, where $\delta(\mathbf{x}, \mathbf{x}') \geq 0$, with equality holding if and only if $\mathbf{x} = \mathbf{x}'$. Then agglomerative clustering can still be used, with the understanding that the nearest pair of clusters is the least dissimilar pair. Interestingly enough, if we define the dissimilarity between two clusters by

$$\delta_{min}(\mathcal{D}_i, \mathcal{D}_j) = \min_{\substack{\mathbf{x}\in\mathcal{D}_i \\ \mathbf{x}'\in\mathcal{D}_j}} \delta(\mathbf{x}, \mathbf{x}') \tag{84}$$

or

$$\delta_{max}(\mathcal{D}_i, \mathcal{D}_j) = \max_{\substack{\mathbf{x}\in\mathcal{D}_i \\ \mathbf{x}'\in\mathcal{D}_j}} \delta(\mathbf{x}, \mathbf{x}'), \tag{85}$$

then the hierarchical clustering procedure will induce a distance function for the given set of n samples. Furthermore, the ranking of the distances between samples will be invariant to any monotonic transformation of the dissimilarity values (Problem 19).

We can now define a generalized *distance* $d(\mathbf{x}, \mathbf{x}')$ between $\mathbf{x}$ and $\mathbf{x}'$ as the value of the lowest level clustering for which $\mathbf{x}$ and $\mathbf{x}'$ are in the same cluster. To show that this is a legitimate distance function, or *metric*, we need to show four things: For all vectors $\mathbf{x}$, $\mathbf{x}'$, and $\mathbf{x}''$,

METRIC

nonnegativity: $d(\mathbf{x}, \mathbf{x}') \geq 0$

reflexivity: $d(\mathbf{x}, \mathbf{x}') = 0$ if and only if $\mathbf{x} = \mathbf{x}'$

symmetry: $d(\mathbf{x}, \mathbf{x}') = d(\mathbf{x}', \mathbf{x})$
triangle inequality: $d(\mathbf{x}, \mathbf{x}') + d(\mathbf{x}', \mathbf{x}'') \geq d(\mathbf{x}, \mathbf{x}'')$.

It is easy to see that these requirements are satisfied and hence that dissimilarity can induce a metric. For our formula for dissimilarity, we have moreover that

$$d(\mathbf{x}, \mathbf{x}'') \leq \max[d(\mathbf{x}, \mathbf{x}'), d(\mathbf{x}', \mathbf{x}'')] \text{ for any } \mathbf{x}' \tag{86}$$

ULTRAMETRIC

in which case we say that $d(\cdot, \cdot)$ is an *ultrametric* (Problem 33). Ultrametric criteria can be more immune to local minima problems since stricter ordering of distances among clusters is maintained.

*10.10 THE PROBLEM OF VALIDITY

With almost all of the procedures considered thus far we have assumed that the number of clusters is known. That is a reasonable assumption if we are upgrading a classifier that has been designed on a small labeled set, or if we are tracking slowly time-varying patterns in an environment with known number of clusters. However, it may be an unjustified assumption if we are exploring a data set whose properties are unknown. Thus, a recurring problem in cluster analysis is that of deciding just how many clusters are present.

When clustering is done by extremizing a criterion function, a common approach is to repeat the clustering procedure for $c = 1$, $c = 2$, $c = 3$, and so on, and to see how the criterion function changes with c. For example, it is clear that the sum-of-squared-error criterion J_e must decrease monotonically with c, because the squared error can be reduced each time c is increased merely by transferring a single sample to a new singleton cluster. If the n samples are really grouped into $\hat{c}$ compact, well-separated clusters, one would expect to see J_e decrease rapidly until $\hat{c} = c$, decreasing much more slowly thereafter until it reaches zero at $c = n$. Similar arguments have been advanced for hierarchical clustering procedures and can be apparent in a dendrogram, the usual assumption being that a large disparity in the levels at which clusters merge indicates the presence of natural groupings.

A more formal approach to this problem is to devise some measure of goodness of fit that expresses how well a given c-cluster description matches the data. The chi-squared and Kolmogorov-Smirnov statistics are the traditional measures of goodness of fit, but the curse of dimensionality usually demands the use of simpler measures, some criterion function, which we denote $J(c)$. Because we expect a description in terms of $c + 1$ clusters to give a better fit than a description in terms of c clusters, we would like to know what constitutes a statistically significant improvement in $J(c)$.

A formal way to proceed is first, to advance the *null hypothesis* that there are exactly c clusters present and, next, to compute the sampling distribution for $J(c+1)$ under this hypothesis. This distribution tells us what kind of apparent improvement to expect when a c-cluster description is actually correct. The decision procedure would be to accept the null hypothesis if the observed value of $J(c + 1)$ falls within limits corresponding to an acceptable probability of false rejection.

Unfortunately, it is usually very difficult to do anything more than crudely estimate the sampling distribution of $J(c + 1)$. The resulting solutions are not above suspicion, and the statistical problem of testing cluster validity is still essentially unsolved. However, under the assumption that a suspicious test is better than none,

we include the following approximate analysis for the simple sum-of-squared-error criterion that closely parallels our discussion in Chapter 8.

Suppose that we have a set $\mathcal{D}$ of n samples and we want to decide whether or not there is any justification for assuming that they form more than one cluster. Let us advance the null hypothesis that all n samples come from a normal population with mean $\boldsymbol{\mu}$ and covariance matrix $\sigma^2\mathbf{I}$.* If this hypothesis were true, multiple clusters found would have to have been formed by chance, and any observed decrease in the sum-of-squared error obtained by clustering would have no significance.

The sum of squared error $J_e(1)$ is a random variable, because it depends on the particular set of samples:

$$J_e(1) = \sum_{\mathbf{x}\in\mathcal{D}} \|\mathbf{x} - \mathbf{m}\|^2, \tag{87}$$

where $\mathbf{m}$ is the sample mean of the full data set. Under the null hypothesis, the distribution for $J_e(1)$ is approximately normal with mean $nd\sigma^2$ and variance $2nd\sigma^4$ (Problem 40). Suppose now that we partition the set of samples into two subsets $\mathcal{D}_1$ and $\mathcal{D}_2$ so as to minimize $J_e(2)$, where

$$J_e(2) = \sum_{i=1}^{2} \sum_{\mathbf{x}\in\mathcal{D}_i} \|\mathbf{x} - \mathbf{m}_i\|^2, \tag{88}$$

$\mathbf{m}_i$ being the mean of the samples in $\mathcal{D}_i$. Under the null hypothesis, this partitioning is spurious, but it nevertheless results in a value for $J_e(2)$ that is smaller than $J_e(1)$. If we knew the sampling distribution for $J_e(2)$, we could determine how small $J_e(2)$ would have to be before we were forced to abandon a one-cluster null hypothesis. Lacking an analytical solution for the optimal partitioning, we cannot derive an exact solution for the sampling distribution. However, we can obtain a rough estimate by considering the suboptimal partition provided by a hyperplane through the sample mean. For large n, it can be shown that the sum of squared error for this partition is approximately normal with mean $n(d - 2/\pi)\sigma^2$ and variance $2n(d - 8/\pi^2)\sigma^4$.

This result agrees with our statement that $J_e(2)$ is smaller than $J_e(1)$, because the mean of $J_e(2)$ for the suboptimal partition—$n(d - 2/\pi)\sigma^2$—is less than the mean for $J_e(1)$—$nd\sigma^2$. To be considered significant, the reduction in the sum-of-squared error must certainly be greater than this. We can obtain an approximate critical value for $J_e(2)$ by assuming that the suboptimal partition is nearly optimal, by using the normal approximation for the sampling distribution, and by estimating σ^2 according to

$$\hat{\sigma}^2 = \frac{1}{nd} \sum_{\mathbf{x}\in\mathcal{D}} \|\mathbf{x} - \mathbf{m}\|^2 = \frac{1}{nd} J_e(1). \tag{89}$$

The final result can be stated as follows (Problem 41): Reject the null hypothesis at the p-percent significance level if

$$\frac{J_e(2)}{J_e(1)} < 1 - \frac{2}{\pi d} - \alpha\sqrt{\frac{2(1 - 8/\pi^2 d)}{nd}}, \tag{90}$$

*We could of course assume a different cluster form, but in the absence of further information the Gaussian can be justified on the grounds we have discussed before.

where α is determined by

$$p = 100 \int_{\alpha}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-u^2/2}\, du = 50(1 - \text{erf}(\alpha/\sqrt{2})), \tag{91}$$

ERROR FUNCTION

and erf(·) is the standard *error function*. This provides us with a test for deciding whether or not the splitting of a cluster is justified. Clearly the c-cluster problem can be treated by applying the same test to all clusters found.

*10.11 ON-LINE CLUSTERING

All of the clustering procedures that we have considered up to this point either explicitly or implicitly attempt to optimize a global criterion function with a known or assumed number of clusters. While this leads to crisp formulations of the problem of unsupervised learning, it does not always produce appropriate or expected results. It frequently leads to cluster structures that are sensitive to small changes in the criterion function, sensitive in ways that may be difficult to appreciate. In particular, when these kinds of procedures are used for on-line learning, it occasionally happens that the cluster structures are not stable, but instead continually wander and drift. Of course, to gain the advantages of being able to learn from new data, a system must be adaptive or exhibit "plasticity," possibly allowing the creation of new clusters, if the data warrant it. On the other hand, if the cluster structures are unstable and the most recently acquired piece of information can cause major reorganization, then it is difficult to ascribe much significance to any particular clustering description. This general problem has been called the *stability/plasticity dilemma*.

STABILITY/ PLASTICITY DILEMMA

One source of this problem is that with clustering based on a global criterion, every sample can have an influence on the location of a cluster center, regardless of how remote it might be. This observation has led to so-called *competitive learning* procedures in which learning adjustments are confined to the cluster that is most similar to the pattern currently being presented. As a result, the characterizations of previously discovered clusters that are unrelated to the current pattern are not disrupted. Competitive learning developed from neural network research, and we shall use neural network terminology to describe these procedures. We begin with a simple procedure that can be thought of as a modification of a sequential k-means algorithm, for reasons that will become clear.

Competitive learning is related to decision-directed versions of k-means (Algorithm 1) and is based on neural network learning rules (Chapter 6). In both procedures, the number of desired clusters and their centers are initialized, and during clustering each pattern is provisionally classified into one of the clusters. The methods of updating the cluster centers differ, however. In the decision-directed method, each cluster center is calculated as the mean of the current provisional members. In competitive learning, the adjustment is confined to the single cluster center most similar to the pattern presented. As a result, in competitive learning, clusters that are "far away" from the current pattern tend not to be altered (but see Section 10.11.2)—sometimes considered a desirable property. The drawback is that the solution need not minimize a single easily defined global cost or criterion function.

We now turn to the specific competitive learning algorithm. For reasons that will become clear, each d-dimensional pattern is augmented (with $x_0 = 1$) and normalized to have length $\|\mathbf{x}\| = 1$; thus all patterns lie on the surface of a $(d + 1)$-

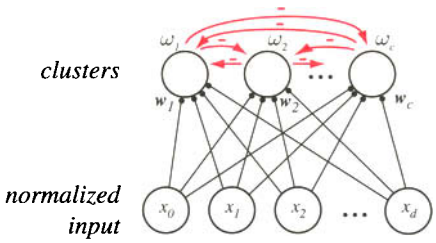


FIGURE 10.15. The two-layer network that implements the competitive learning algorithm consists of $d+1$ input units and c output or cluster units. Each augmented input pattern is normalized to unit length (i.e., $\|\mathbf{x}\| = 1$), as is the set of weights at each cluster unit. When a pattern is presented, each of the cluster units computes its net activation $net_j = \mathbf{w}_j^t\mathbf{x}$; only the weights at the most active cluster unit are modified. (The suppression of activity in all but the most active cluster units can be implemented by competition among these units, as indicated by the red arrows.) The weights of the most active unit are then modified to be more similar to the pattern presented.

dimensional hypersphere. The competitive learning algorithm can be understood by its neural network implementation (Fig. 10.15), which resembles a Perceptron network (Chapter 5), with input units fully connected to c output or cluster units.

Each of the c cluster centers is initialized with a randomly chosen weight vector, also normalized $\|\mathbf{w}_j\| = 1$, $j = 1, \ldots, c$. It is traditional, but not required, to initialize cluster centers to be c points randomly selected from the data. When a new pattern is presented, each of the cluster units computes its net activation, $net_j = \mathbf{w}_j^t\mathbf{x}$. Only the unit with the largest net (i.e., whose weight vector is closest to the new pattern), is permitted to update its weights. While this selection of the most active unit is algorithmically trivial, it appears to be an additional operation, external to the network. However, if desired, it can be implemented in a winner-take-all network, where each cluster unit j inhibits others by an amount proportional to net_j, as shown by the red arrows in Fig. 10.15. It is this competition between cluster units, along with the resulting suppression of activity in all but the one with the largest net, that gives the algorithm its name.

As mentioned, learning is confined to the weights at the most active unit. The weight vector at this unit is updated to be more like the pattern:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta\mathbf{x}, \tag{92}$$

where η is a learning rate. The weights are then normalized to ensure $\sum_{i=0}^{d} w_i^2 = 1$. This normalization guarantees that the net activation $\mathbf{w}^t\mathbf{x}$ depends only on the angle between $\mathbf{w}$ and $\mathbf{x}$, and does not depend on the magnitude of $\mathbf{w}$; without such weight normalization, a single weight, say $\mathbf{w}_j$, could grow in magnitude and forever give the greatest value net_j, and through competition thereby prevent other clusters from learning. Figure 10.16 shows the trajectories of three cluster centers in response to a sequence of patterns chosen randomly from the set shown. If we let k denote a stopping criterion, the algorithm is as follows:

■ **Algorithm 6. (Competitive Learning)**

1 **begin initialize** $\eta, n, c, k, \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_c$
2 $\quad\quad \mathbf{x}_i \leftarrow \{1, \mathbf{x}_i\},\ i = 1, \ldots, n$ (augment all patterns)

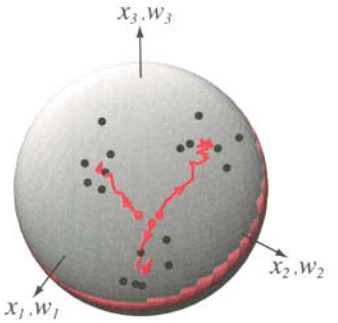


FIGURE 10.16. All of the two-dimensional patterns have been augmented and normalized and hence lie on a two-dimensional sphere in three dimensions. Likewise, the weights of the three cluster centers have been normalized. The red curves show the trajectory of the weight vectors, which start at the red points and end at the center of a cluster.

```
3        x_i ← x_i/‖x_i‖, i = 1, ..., n    (normalize all patterns)
4        do randomly select a pattern x
5           j ← arg max_j' w_j'^t x        (classify x)
6           w_j ← w_j + ηx                 (weight update)
7           w_j ← w_j/‖w_j‖                (weight normalization)
8        until no significant change in w in k attempts
9     return w_1, w_2, ..., w_c
10 end
```

A drawback of Algorithm 6 is that there is no guarantee that it will terminate, even for a finite, nonpathological data set. For instance, the termination condition in line 8 may never be satisfied, and thus the weights may vary forever. A simple heuristic is to let the learning rate in line 6 decrease over time, for instance by letting $\eta(t) = \eta(0)\alpha^t$ for $0 < \alpha < 1$, where t is an iteration number. If the initial cluster centers are representative of the full data set, and the rate of decay is set so that the full data set is presented at least several times before the learning is reduced to very small values, then good results can be expected. However, if a novel pattern is added, it cannot be learned, because η is too small. Likewise, such a learning decay scheme is inappropriate if we seek to track gradual but ongoing changes in the data.

10.11.1 Unknown Number of Clusters

We have mentioned the problem of unknown number c of cluster centers. When c is unknown, we can proceed in one of two general ways. In the first, we solve the problem repeatedly for many different values of c, and compare some criterion for each clustering. If there is a large gap in the criterion values, it suggests a "natural" number of clusters. A second approach is to state a threshold for the creation of a new cluster. This latter approach is particularly useful in on-line cases. The drawback is that it depends more strongly on the order of data presentation.

Whereas clustering algorithms such as k-means and hierarchical clustering typically have all data present before clustering begins (i.e., are off-line), there are occasionally situations in which clustering must be performed on-line as the data streams

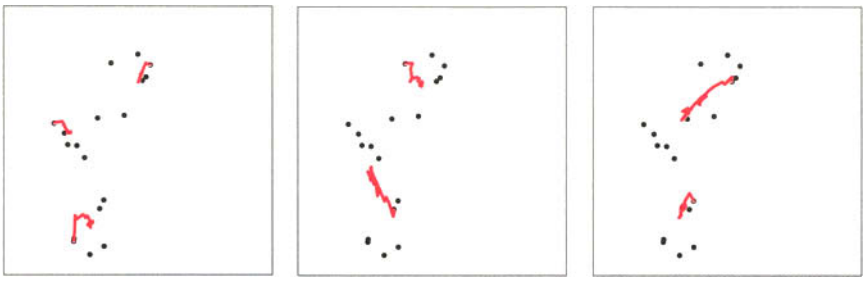


FIGURE 10.17. In leader-follower clustering, the number of clusters and their centers depend upon the random sequence of presentations of the points. The three simulations shown employed the same learning rate η, threshold θ, and number of presentations of each point (50), but differ in the random sequence of presentations. Notice that in the simulation on the left, three clusters are generated, whereas only two are generated in the other simulations.

in—for instance, when there is inadequate memory to store all the patterns themselves, or in a time-critical situation where the clusters need to be used even before the full data are present. Our graph-theoretic methods can be performed on-line—one merely links the new pattern to an existing cluster based on some similarity measure.

In order to make on-line versions of methods such as k-means, we will have to be a bit more careful. Under these conditions, the best approach generally is to represent clusters by their "centers" (e.g., means) and update a center based solely on its current value and the incoming pattern.

Suppose we currently have c cluster centers; they may have been placed initially at random positions, or at positions corresponding to the first c patterns presented, or in the current state after any number of patterns have been presented. The simplest approach is to alter only the cluster center most similar to a new pattern being presented, and the cluster center is changed to be somewhat more like the pattern (Fig. 10.17), a generic approach called *leader-follower clustering*.

LEADER-FOLLOWER CLUSTERING

If we let $\mathbf{w}_i$ represent the current center for cluster i, let η represent a learning rate, and let θ represent a threshold, then a Basic leader-follower clustering algorithm is as follows:

Algorithm 7. (Basic Leader-Follower Clustering)

```
1  begin initialize η, θ
2          w_1 ← x
3          do accept new x
4              j ← arg min_j' ||x − w_j'||      (find nearest cluster)
5              if ||x − w_j|| < θ
6                then w_j ← w_j + ηx
7                 else add new w ← x
8                w ← w/||w||                    (normalize weight)
9          until no more patterns
10   return w_1, w_2, ...
11 end
```

Naturally, for a given problem the threshold θ determines implicitly the number of clusters that will be formed. A large threshold leads to a small number of large clusters, while a small threshold leads to a large number of small clusters. In the absence of information about the data, there are no firm guidelines to setting the threshold so that the "proper" number of clusters are found. It should be noted that Algorithm 7 has no provision for *reducing* the number of clusters, by, for instance, merging two clusters that are sufficiently similar.

Before we discuss further properties of such a leader-follower clustering algorithm, let us consider one popular neural approach based on it.

10.11.2 Adaptive Resonance

Leader-follower clustering is at the heart of a general approach to designing self-organizing neural networks that is known as adaptive resonance theory or ART. This theory was developed primarily to model how biological neural networks might be able to recognize unexpected patterns and to remember them for future use. If we think of one of these unexpected patterns as being a new cluster center, then one of the goals of ART is to make sure that even if slightly different new instances of the pattern called for some adjustments to this new cluster center, it would retain its basic characteristics in a stable fashion (Fig. 10.18). Another goal is to show how *expectations* can influence the response of a network. This leads to feedback connections and interesting dynamic behavior.

Adaptive resonance theory can be applied to a number of different network structures. For simplicity, we shall consider only the simple two-layer structure in Fig. 10.19. We shall provide only a general outline of its structure and behavior, suppressing many details that would be needed for a complete working implementation.

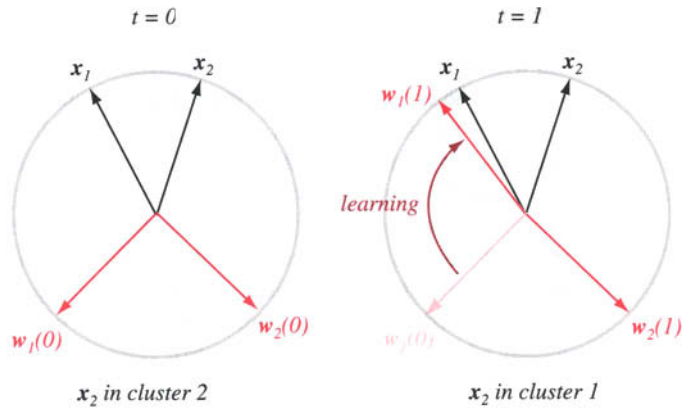


FIGURE 10.18. Instability and recoding can occur during competitive learning, as illustrated in this simple case of two patterns and two cluster centers. Two patterns, $\mathbf{x}_1$ and $\mathbf{x}_2$, are presented to a 2-2 network of Fig. 10.15 represented by two weight vectors. At $t = 0$, $\mathbf{w}_1$ happens to be most aligned with $\mathbf{x}_1$ and hence this pattern belongs to cluster 1; likewise, $\mathbf{x}_2$ is most aligned with $\mathbf{w}_2$ and hence it belongs to cluster 2, as shown at the left. Next, suppose pattern $\mathbf{x}_1$ is presented several times; through the competitive learning weight update rule, $\mathbf{w}_1$ moves to become closer to $\mathbf{x}_1$. Now $\mathbf{x}_2$ is most aligned with $\mathbf{w}_1$, and thus it has changed from class 2 to class 1. Surprisingly, this recoding of $\mathbf{x}_2$ occurs even though $\mathbf{x}_2$ was not used for weight update. It is theoretically possible that such recoding will occur numerous times in response to particular sequences of pattern presentations.

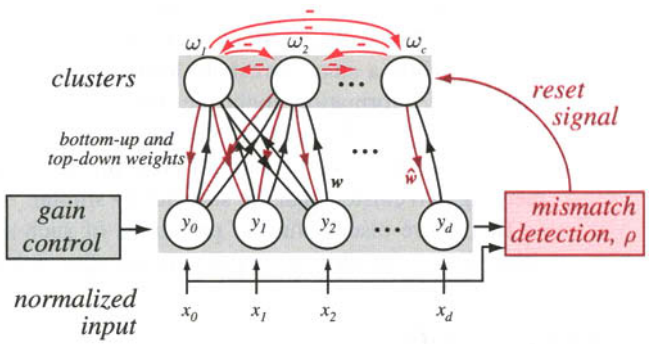


FIGURE 10.19. A generic adaptive resonance network has inputs and cluster units, much like a network for performing competitive learning. However, the input and the category layers are fully interconnected by *both* bottom-up and top-down connections with weights. The bottom-up weights, denoted $\mathbf{w}$, learn the cluster centers while the top-down weights, $\hat{\mathbf{w}}$, learn expected input patterns. If a match between input and a learned cluster is poor (where the quality of the match is specified by a user-specified vigilance parameter ρ), then the active cluster unit is suppressed by a reset signal, and a new cluster center can be recruited.

As Fig. 10.19 shows, the network has an input layer, fully interconnected by bottom-up weights to units in a cluster layer, much like a network implementation of competitive learning. In addition, an ART network has top-down weights, denoted $\hat{\mathbf{w}}$ in the figure; these will give "priming signals" back to units in the input layer. The units in the bottom layer of the network receive three kinds of inputs: (a) the usual input pattern $\mathbf{x}$, (b) feedback signals from the top layer through the top-down weights, and (c) a time-varying bias signal from the so-called gain-control unit. The bottom-up weights $\mathbf{w}_i$ going into the ith output unit are normalized to unit length. As with the basic leader-follower algorithm, the number of clusters can increase over time, so the number c of units in the top layer is variable.

When the ART network is working properly, if the inputs to the bottom layer are close to a familiar pattern, the outputs of the bottom layer are supposed to be a "cleaned-up" version of the input $\mathbf{x}$. To be more precise, if the input vector $\mathbf{x}$ is close to a cluster center $\mathbf{w}_i$, then the output $\mathbf{y}$ of the bottom layer would ideally be $\mathbf{w}_i$ itself. What is it that drives the bottom-layer responses toward a cluster center? Two things: (a) feedback from the top layer, and (b) a "gain-control" signal. In the absense of either feedback or gain control, the outputs of the bottom layer would merely be a copy of the inputs. The gain-control system is just a mechanism to keep a constant level of activity in the bottom layer, i.e., to hold $\|\mathbf{y}\|$ constant. The feedback signals come from the most active cluster unit, through top-down weights $\hat{\mathbf{w}}$.

The weights going to each unit in the top layer represent the long-term memory of a cluster center. The top layer employs a "winner-take-all" competition, so only the unit i for which $\mathbf{x}^t\mathbf{w}_i$ is large will have a strong reponse. In clustering terms, the output unit that responds most strongly identifies the cluster center $\mathbf{w}_i$ that is nearest to the cleaned-up input $\mathbf{y}$ emerging from the bottom layer.

Clearly, for all vectors $\mathbf{y}$ of a given length, the input that will cause output unit i to have the strongest response is proportional to $\mathbf{w}_i$, just as in standard competitive learning. The top-down weights leading from that output unit provide an *expectation* for what the response that it most wants to see from the bottom layer. When an input $\mathbf{x}$ is first presented, the output $\mathbf{y}$ of the bottom layer is just the input vector $\mathbf{x}$, which

may or may not be particularly close to the weight vector $\mathbf{w}_i$ for the most strongly responding output unit. However, the network uses top-down signals to provide *additional input* to the bottom layer through the feedback connections from the top layer. After some time delay and adjustment by the gain control, this has the effect of pushing the responses of the bottom-layer units closer to $\mathbf{w}_i$, which in turn stimulates the top unit more strongly, which in turn stimulates the input, and so on. In ART terminology, this convergence of the feedback sequence is called a "resonance," although the behavior is unrelated to the resonant response of a driven oscillator.

But what might this network actually do? One possibility is that it might actually arrive at a stable state in which (a) the input $\mathbf{x}$ is close to $\mathbf{w}_i$, (b) the output of the bottom-layer units is very close to $\mathbf{w}_i$, and (c) only the ith top-layer unit is responding strongly. That represents the desired result when the input is indeed close to $\mathbf{w}_i$. As with the basic leader-follower algorithm, the value of $\mathbf{w}_i$ can be adjusted slightly to be closer to the new input.

However, another possibility is that the feedback is so powerful that even though the input vector $\mathbf{x}$ is very different from all of the cluster centers, one of the output units grabs control, and the output of the first layer is $\mathbf{w}_i$ anyway. This is not a desirable situation. In the leader-follower algorithm, it corresponds to the situation in which a new cluster center should be introduced. In ART terminology, the fact that there is a large difference between $\mathbf{x}$ and $\mathbf{w}_i$ is detected by the so-called *orienting subsystem*, which allows a new cluster unit to form or be recruited and initialized to $\mathbf{x}$ whenever the input and the activations at the input layer differ such that $\mathbf{x}^t\mathbf{y} < \rho$.

ORIENTING SUBSYSTEM

VIGILANCE

Here ρ is a user-determined parameter called the *vigilance*. The role of the vigilance is analogous to the role of the threshold θ in the leader-follower algorithm. If the vigilance is low, there can be a poor "match" between the input and the closest learned cluster and the network will accept it anyway. However, if the vigilance is high, the network will frequently generate new cluster centers. In ART networks this is implemented by a "reset wave," whose biologically inspired properties need not concern us. For the same input data set, a low vigilance leads to a small number of large coarse clusters, while a high vigilance leads to a large number of fine clusters.

This description of adaptive resonance is incomplete, and needs to be more precisely stated before the behavior of an actual network could be simulated. In many ways, one can view ART as being merely a neural-network implementation of leader-follower clustering. However, the network realization suggests interesting generalizations, such as using multi-layer architectures and allowing higher-level or "cross-modality" expectations to influence the activation of lower-level units, which are explored in the references in the Bibliography.

10.11.3 Learning with a Critic

Most of this text has addressed the problem of supervised learning, where a teacher provides the category labels of each training pattern. This chapter concerns unsupervised where no such label information is available. There is an intermediate case, in which the teacher knows only that the category provided by the classifier is correct or incorrect. Thus, the teacher serves as a *critic* of the system as it learns.

It is simple to incorporate learning with a critic in competitive learning and adaptive resonance networks. For instance, if the tentative cluster assignment for a pattern is judged to be correct by the critic, then standard learning is permitted. If, however, the cluster is judged to be incorrect by the critic, then no learning (weight update) is permitted.

*10.12 GRAPH-THEORETIC METHODS

Where the mathematics of normal mixtures and minimum-variance partitions leads us to picture clusters as isolated clumps, the language and concepts of graph theory lead us to consider much more intricate structures. Unfortunately, there is no uniform way of posing clustering problems as problems in graph theory. Thus, the effective use of these ideas is still largely an art, and the reader who wants to explore the possibilities should be prepared to be creative.

We begin our brief look into graph-theoretic methods by reconsidering the simple procedures that produce the graphs shown in Fig. 10.7. Here a threshold distance d_0 was selected, and two points are placed in the same cluster if the distance between them is less than d_0. This procedure can easily be generalized to apply to arbitrary similarity measures. Suppose that we pick a threshold value s_0 and say that $\mathbf{x}_i$ is similar to $\mathbf{x}_j$ if $s(\mathbf{x}_i, \mathbf{x}_j) > s_0$. This defines an $n \times n$ *similarity matrix* $\mathbf{S} = [s_{ij}]$, with binary element

SIMILARITY MATRIX

$$s_{ij} = \begin{cases} 1 & \text{if } s(\mathbf{x}_i, \mathbf{x}_j) > s_0 \\ 0 & \text{otherwise.} \end{cases} \tag{93}$$

SIMILARITY GRAPH

Furthermore, this matrix induces a *similarity graph*, dual to $\mathbf{S}$, in which nodes correspond to points and an edge joins node i and node j if and only if $s_{ij} = 1$.

The clusterings produced by the single-linkage algorithm and by a modified version of the complete-linkage algorithm are readily described in terms of this graph. With the single-linkage algorithm, two samples $\mathbf{x}$ and $\mathbf{x}'$ are in the same cluster if and only if there exists a chain $\mathbf{x}, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k, \mathbf{x}'$ such that $\mathbf{x}$ is similar to $\mathbf{x}_1$, $\mathbf{x}_1$ is similar to $\mathbf{x}_2$, and so on for the whole chain. Thus, this clustering corresponds to the *connected components* of the similarity graph. With the complete-linkage algorithm, all samples in a given cluster must be similar to one another, and no sample can be in more than one cluster. If we drop this second requirement, then this clustering corresponds to the *maximal complete subgraphs* of the similarity graph—the "largest" subgraphs with edges joining all pairs of nodes. (In general, the clusters of the complete-linkage algorithm will be found among the maximal complete subgraphs, but they cannot be determined without knowing the unquantized similarity values.)

CONNECTED COMPONENT

MAXIMAL COMPLETE SUBGRAPH

We have noted above that the nearest-neighbor algorithm could be viewed as an algorithm for finding a minimal spanning tree. Conversely, given a minimal spanning tree we can find the clusterings produced by the nearest-neighbor algorithm. Removal of the longest edge produces the two-cluster grouping, removal of the next longest edge produces the three-cluster grouping, and so on. This amounts to a divisive hierarchical procedure, and it suggests other ways of dividing the graph into subgraphs. For example, in selecting an edge to remove, we can compare its length to the lengths of other edges incident upon its nodes. Let us say that an edge is *inconsistent* if its length l is significantly larger than $\bar{l}$, the average length of all other edges incident on its nodes. Figure 10.20 shows a minimal spanning tree for a two-dimensional point set and the clusters obtained by systematically removing all edges for which $l > 2\bar{l}$ in this way. This criterion is sensitive to local conditions, giving results that are quite different from merely removing the two longest edges.

INCONSISTENT EDGE

When the data points are strung out into long chains, a minimal spanning tree forms a natural skeleton for the chain. If we define the *diameter path* as the longest path through the tree, then a chain will be characterized by the shallow depth of the branching off the diameter path. In contrast, for a large, uniform cloud of data

DIAMETER PATH

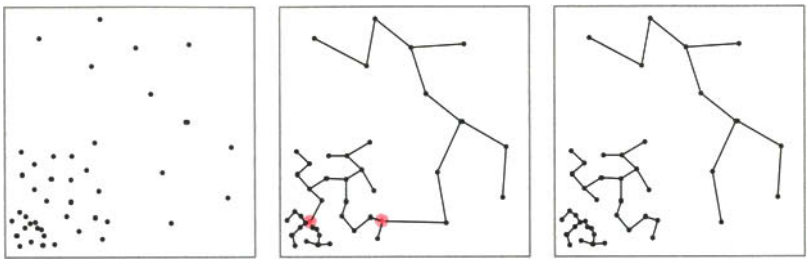


FIGURE 10.20. The removal of inconsistent edges—ones with length significantly larger than the average incident upon a node—may yield natural clusters. The original data are shown at the left, and its minimal spanning tree is shown in the middle. At virtually every node, incident edges are of nearly the same length. Each of the two nodes shown in red are exceptions: their incident edges are of very different lengths. When the two such inconsistent edges are removed, three clusters are produced, as shown at the right.

points, the tree will usually not have an obvious diameter path, but rather several distinct, near-diameter paths. For any of these, an appreciable number of nodes will be off the path. While slight changes in the locations of the data points can cause major rerouting of a minimal spanning tree, they typically have little effect on such statistics.

One of the useful statistics that can be obtained from a minimal spanning tree is the edge length distribution. Figure 10.21 shows a situation in which two dense clusters are embedded in a sparse set of points; the lengths of the edges of the minimal spanning tree exhibit two distinct clusters that would easily be detected by a minimum-variance procedure. By deleting all edges longer than some intermediate value, we can extract the dense cluster as the largest connected component of the remaining graph. While more complicated configurations cannot be disposed of this easily, the flexibility of the graph-theoretic approach suggests that it is applicable to a wide variety of clustering problems.

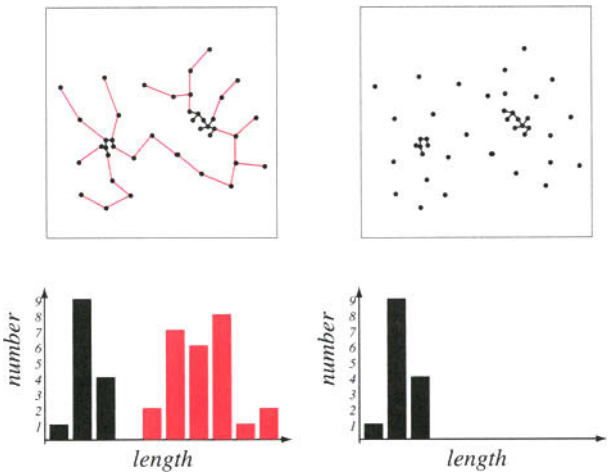


FIGURE 10.21. A minimal spanning tree is shown at the left; its bimodal edge length distribution is evident in the histogram below. If all links of intermediate or high length are removed (red), the two natural clusters are revealed (right).

10.13 COMPONENT ANALYSIS

Component analysis is an unsupervised approach to finding the "right" features from the data. We shall discuss several leading methods, each having a somewhat different goal. We have seen in Chapter 3 how principal component analysis (PCA) projects d-dimensional data onto a lower-dimensional subspace in a way that is optimal in a sum-squared error sense. Nonlinear component analysis (NLCA), typically implemented by neural techniques, is a nonlinear generalization of PCA. In independent component analysis (ICA) we seek those directions in feature space that show the independence of signals. This method is particularly helpful for segmenting signals from multiple sources.

10.13.1 Principal Component Analysis (PCA)

KARHUNEN-LOÉVE TRANSFORM

For completeness, we reiterate the basic approach in principal components or *Karhunen-Loéve transform* presented in Chapter 3. First, the d-dimensional mean vector $\boldsymbol{\mu}$ and $d \times d$ covariance matrix $\boldsymbol{\Sigma}$ are computed for the full data set. Next, the eigenvectors and eigenvalues are computed (cf. Section A.2.7 of the Appendix), and sorted according to decreasing eigenvalue. Call these eigenvectors $\mathbf{e}_1$ with eigenvalue λ_1, $\mathbf{e}_2$ with eigenvalue λ_2, and so on, and choose the k eigenvectors having the largest eigenvalues. Often there will be just a few large eigenvalues, and this implies that k is the inherent dimensionality of the subspace governing the "signal" while the remaining $d - k$ dimensions generally contain noise. Next we form a $d \times k$ matrix $\mathbf{A}$ whose columns consist of the k eigenvectors. The representation of data by principal components consists of projecting the data onto the k-dimensional subspace according to

$$\mathbf{x}' = \mathbf{F}_1(\mathbf{x}) = \mathbf{A}^t(\mathbf{x} - \boldsymbol{\mu}). \tag{94}$$

AUTO-ENCODER

A simple three-layer linear neural network, trained as an auto-encoder can form such a representation, as shown in Fig. 10.22. Each pattern of the data set is presented to *both* the input and output layers and the full network trained by gradient descent on a sum-squared-error criterion, for instance by backpropagation. It can be shown

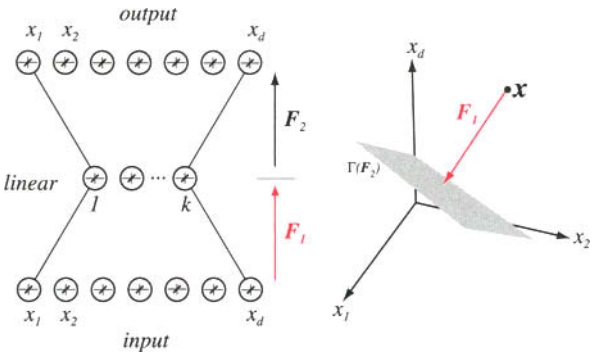


FIGURE 10.22. A three-layer neural network with linear hidden units, trained to be an auto-encoder, develops an internal representation that corresponds to the principal components of the full data set. The transformation $\mathbf{F}_1$ is a linear projection onto a k-dimensional subspace denoted $\Gamma(\mathbf{F}_2)$.

that this representation minimizes a squared error criterion (Problem 44). After the network is trained, the top layer is discarded, and the linear hidden layer provides the principal components.

10.13.2 Nonlinear Component Analysis (NLCA)

Principal component analysis yields a k-dimensional linear subspace of feature space that best represents the full data according to a minimum-square-error criterion. If the data represent complicated interactions of features, then the linear subspace may be a poor representation and nonlinear components may be needed.

A neural network approach to such nonlinear component analysis employs a network with five layers of units, as shown in Fig. 10.23. The middle layer consists of $k < d$ linear units, and it is here that the nonlinear components will be revealed. It is important that the two other internal layers have nonlinear units (Problem 46). The entire network is trained using the techniques of Chapter 6 as an auto-encoder or auto-associator. That is, each d-dimensional pattern is presented as both the input and as the target or desired output. When trained using a sum-squared error criterion, such a network readily learns the auto-encoder problem. The top two layers of the trained network are discarded, and the rest used for nonlinear component analysis. For each input pattern $\mathbf{x}$, the outputs of the k units of the three-layer network correspond to the nonlinear components.

We can understand the function of the full five-layer network in terms of two successive mappings, $\mathbf{F}_1$ followed by $\mathbf{F}_2$. As Fig. 10.23 illustrates, $\mathbf{F}_1$ is a projection from the d-dimensional input onto a k-dimensional nonlinear subspace and $\mathbf{F}_2$ is a mapping from that subspace back to the full d-dimensional space.

There are often multiple local minima in the error surface associated with the five-layer network, and we must take care to set an appropriate number k of units. Recall that in (linear) principal component analysis, the number of components k could be chosen based on the spectrum of eigenvectors. If the eigenvalues are ordered by magnitude, any significant drop between successive values indicates a "natural" number dimension to the subspace. Likewise, suppose five-layer networks are trained, with

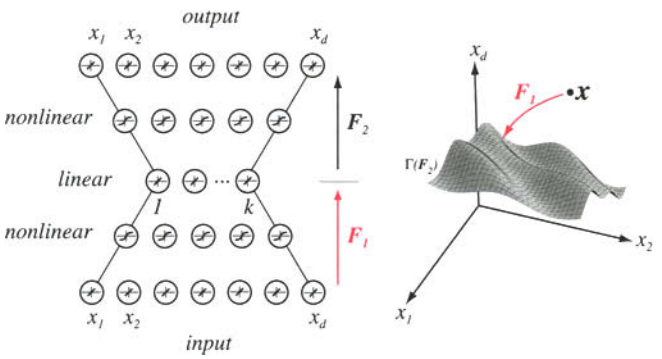


FIGURE 10.23. A five-layer neural network with two layers of nonlinear units (e.g., sigmoidal), trained to be an auto-encoder, develops an internal representation that corresponds to the nonlinear components of the full data set. The process can be viewed in feature space (at the right). The transformation $\mathbf{F}_1$ is a nonlinear projection onto a k-dimensional subspace, denoted $\Gamma(\mathbf{F}_2)$. Points in $\Gamma(\mathbf{F}_2)$ are mapped via $\mathbf{F}_2$ back to the the d-dimensional space of the original data. After training, the top two layers of the net are removed and the remaining three-layer network maps inputs $\mathbf{x}$ to the space $\Gamma(\mathbf{F}_2)$.

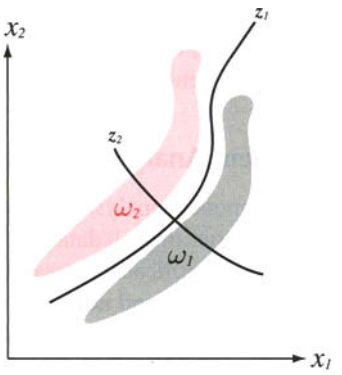


FIGURE 10.24. Features from two classes are as shown, along with nonlinear components of the full data set. Apparently, these classes are well-separated along the line marked z_2, but the large noise gives the largest nonlinear component to be along z_1. Preprocessing by keeping merely the largest nonlinear component would retain the "noise" and discard the "signal," giving poor recognition. The same defect can arise in linear principal components, where the coordinates are linear and orthogonal.

different numbers k of units in the middle layer. Assuming poor local minima have been avoided, the training error will surely decrease for successively larger values of k. If the improvement $k + 1$ over k is small, this may indicate that k is the "natural" dimension of the subspace at the network's middle layer.

We should not conclude that principal component analysis or nonlinear component analysis is always beneficial for classification. If the noise is large compared to the difference between categories, then component analysis will find the directions of the noise, rather than the signal, as illustrated in Fig. 10.24. In such cases, we seek to ignore the noise, and instead we extract the directions that are indicative of the categories—a technique we consider next.

*10.13.3 Independent Component Analysis (ICA)

BLIND SOURCE SEPARATION

While principal component analysis and nonlinear component analysis seek directions in feature space that best represent the data in a sum-squared error sense, independent component analysis instead seeks directions that are most *independent* from each other. This goal of ICA can be understood in the domain of *blind source separation*. Suppose there are d independent scalar source signals $x_i(t)$ for $i = 1, \ldots, d$, where we can consider t to be a time index $1 \leq t \leq T$. For notational convenience we group the d values at an instant into a vector $\mathbf{x}(t)$ and assume for simplicity that averaged over the full data set, the mean of $\mathbf{x}$ vanishes. Because of our assumptions of source independence and the absense of noise, we can write the multivariate density function as

$$p[\mathbf{x}(t)] = \prod_{i=1}^{d} p[x_i(t)]. \tag{95}$$

Suppose that a k-dimensional data (or sensor) vector is observed at each moment,

$$\mathbf{s}(t) = \mathbf{A}\mathbf{x}(t), \tag{96}$$

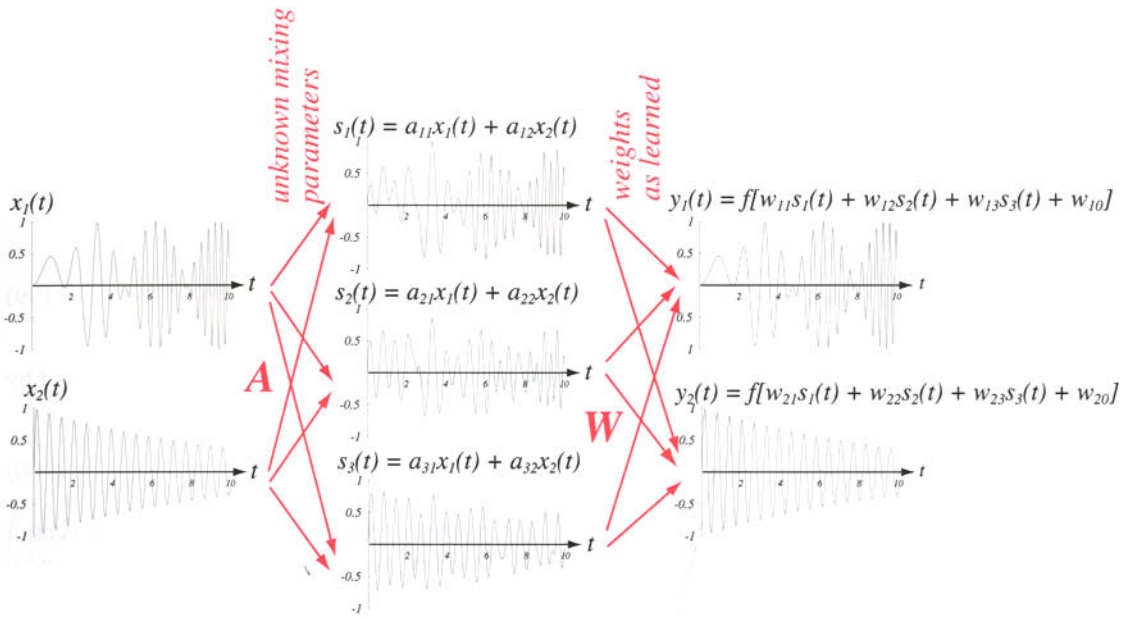


FIGURE 10.25. Independent component analysis (ICA) is an unsupervised method that can be applied to the problem of blind source separation. In such problems, two or more source signals (assumed independent) $x_1(t)$, $x_2(t), \cdots, x_d(t)$ are mixed linearly to yield sum signals $s_1(t), s_2(t), \cdots, s_k(t)$, where $k \geq d$. (This figure illustrates the case $d = 2$ and $k = 3$.) Given merely the sensed signals $\mathbf{x}(t)$ and an assumed number of components, d, the task of ICA is to find independent components in $\mathbf{s}$. In a blind source separation application, these are merely the source signals.

where $\mathbf{A}$ is a $k \times d$ matrix. If $\mathbf{x}$ represents acoustic sources, and $\mathbf{s}$ the signals in k microphones, then the matrix $\mathbf{A}$ depends upon the attenuations due to the source-sensor separations. The goal of ICA is to extract d components in $\mathbf{s}$ that are independent. If the components of $\mathbf{x}$ are independent as described by Eq. 95, then the components found by ICA will allow the discovery of the source signals themselves (Fig. 10.25). (We shall ignore any effects of delays or room echoes which greatly complicate the practical application of this particular example.)

For simplicity, we will treat the case where the number of sensor signals is the same as the number of desired independent components, i.e., we let $k = d$ in the below. (Problem 49 asks you to generalize the following results to the case where $k > d$.) The distribution in the outputs is related to the distribution

$$p_{\mathbf{y}}(\mathbf{y}) = \frac{p_{\mathbf{s}}(\mathbf{s})}{|\mathbf{J}|}, \tag{97}$$

where $\mathbf{J}$ is the Jacobean matrix

$$\mathbf{J} = \begin{pmatrix} \frac{\partial y_1}{\partial s_1} & \cdots & \frac{\partial y_d}{\partial s_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial s_d} & \cdots & \frac{\partial y_d}{\partial s_d} \end{pmatrix} \tag{98}$$

and

$$|\mathbf{J}| = \left| |\mathbf{W}| \prod_{i=1}^{d} \frac{\partial y_i}{\partial s_i} \right|. \tag{99}$$

We model the final stage as a linear transform of the source signals, followed by a static nonlinearity, i.e.,

$$\mathbf{y} = f[\mathbf{W}\mathbf{s} + \mathbf{w}_0] \tag{100}$$

where $\mathbf{w}_0$ is a bias vector and $f[\cdot]$ is typically chosen to be a sigmoid. The central goal in ICA is to find the parameters $\mathbf{W}$ and $\mathbf{w}_0$ so as to make the outputs y_i as independent from one another as possible. The natural measure of such independence—and hence the criterion to be maximized—is the *joint entropy*

JOINT ENTROPY

$$\begin{aligned} H(\mathbf{y}) &= -\mathcal{E}[\ln p_{\mathbf{y}}(\mathbf{y})] \\ &= \mathcal{E}[\ln |\mathbf{J}|] - \underbrace{\mathcal{E}[\ln p_{\mathbf{s}}(\mathbf{s})]}_{\substack{\text{independent} \\ \text{of weights}}}, \end{aligned} \tag{101}$$

where the expectation is over the the full samples $t = 1, \ldots, T$.

Thus the learning rule for the weight matrix, based on gradient descent of the criterion in Eq 101, is

$$\Delta \mathbf{W} \propto \frac{\partial H(\mathbf{y})}{\partial \mathbf{W}} = \frac{\partial}{\partial \mathbf{W}} \ln |\mathbf{J}| = \frac{\partial}{\partial \mathbf{W}} \ln |\mathbf{W}| + \frac{\partial}{\partial \mathbf{W}} \ln \prod_{i=1}^{d} \left| \frac{\partial y_i}{\partial s_i} \right|, \tag{102}$$

where we used Eq. 99. In component form, the first term on the right-hand side of Eq. 102 is

$$\frac{\partial}{\partial W_{ij}} \ln |\mathbf{W}| = \frac{\text{cof}[W_{ij}]}{|\mathbf{W}|} \tag{103}$$

where the cofactor of W_{ij}, that is, $(-1)^{i+j}$ times the determinant of $(d-1)$-by-$(k-1)$-dimensional matrix obtained by deleting the ith row and jth column of $\mathbf{W}$ (cf. Section A.2.6 of the Appendix). Thus we have

$$\frac{\partial}{\partial \mathbf{W}} \ln |\mathbf{W}| = [\mathbf{W}^t]^{-1} \tag{104}$$

Assuming a sigmoidal nonlinearity, Eq. 103 then gives our weight update rule for the matrix $\mathbf{W}$

$$\Delta \mathbf{W} \propto [\mathbf{W}^t]^{-1} + (\mathbf{1} - 2\mathbf{y})\mathbf{s}_g^t, \tag{105}$$

where $\mathbf{1}$ is a d-component vector of 1s.

Problem 48 asks you to make the same assumptions as above, and follow similar arguments, and show that the learning rule for the bias weights is

$$\Delta \mathbf{w}_0 \propto \mathbf{1} - 2\mathbf{y}. \tag{106}$$

Equations 105 and 106 give the learning rule for ICA. It is often difficult to know ahead of time the proper number of components to seek. If ICA is being used in a pattern recognition application with known number of categories, then in the absence of other information d should be set to equal to the number of categories. If, however, this number is fairly large, then ICA may be overly sensitive to numerical simulation and give unreliable components.

Generally speaking, when used as preprocessing for *classification*, ICA has several characteristics that make it more desirable than linear or nonlinear PCA. As we saw in Fig. 10.24, such principal components need not be effective in separating classes. If the different sources arise from different models, then we can expect them to be independent, and ICA should be able to extract them.

10.14 LOW-DIMENSIONAL REPRESENTATIONS AND MULTIDIMENSIONAL SCALING (MDS)

Part of the problem of deciding whether or not a given clustering means anything stems from our inability to visualize the structure of multidimensional data. This problem is further aggravated when similarity or dissimilarity measures are used that lack the familiar properties of distance. One way to attack this problem is to try to represent the data points as points in some lower-dimensional space in such a way that the distances between points in that space correspond to the dissimilarities between points in the original space. If acceptably accurate representations can be found in two or perhaps three dimensions, this can be an extremely valuable way to gain insight into the structure of the data. The general process of finding a configuration of points whose interpoint distances correspond to similarities or dissimilarities is often called *multidimensional scaling*.

Let us begin with the simpler case where it is meaningful to talk about the distances between the n samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$. Let $\mathbf{y}_i$ be the lower-dimensional *image* of $\mathbf{x}_i$, let δ_{ij} be the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$, and let d_{ij} be the distance between $\mathbf{y}_i$ and $\mathbf{y}_j$ (Fig. 10.26). Then we are looking for a *configuration* of image points $\mathbf{y}_1, \ldots, \mathbf{y}_n$ for which the $n(n-1)/2$ distances d_{ij} between image points are as close as possible to the corresponding original distances δ_{ij}. Because it will usually not be possible to find a configuration for which $d_{ij} = \delta_{ij}$ for all i and j, we need some criterion for deciding whether or not one configuration is better than another. The following sum-of-squared-error functions are all reasonable candidates:

$$J_{ee} = \frac{\sum_{i<j}(d_{ij} - \delta_{ij})^2}{\sum_{i<j}\delta_{ij}^2} \tag{107}$$

$$J_{ff} = \sum_{i<j}\left(\frac{d_{ij} - \delta_{ij}}{\delta_{ij}}\right)^2 \tag{108}$$

$$J_{ef} = \frac{1}{\sum_{i<j}\delta_{ij}} \sum_{i<j} \frac{(d_{ij} - \delta_{ij})^2}{\delta_{ij}}. \tag{109}$$

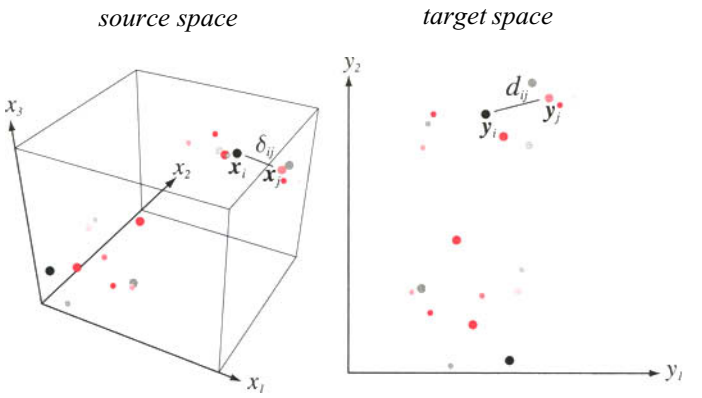


FIGURE 10.26. The figure shows an example of points in a three-dimensional space being mapped to a two-dimensional space. The size and color of each point $\mathbf{x}_i$ matches that of its image, $\mathbf{y}_i$. Here we use simple Euclidean distance, that is, $\delta_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$ and $d_{ij} = \|\mathbf{y}_i - \mathbf{y}_j\|$. In typical applications, the source space usually has high dimensionality, but to allow easy visualization the target space is only two- or three-dimensional.

Because these criterion functions involve only the distances between points, they are invariant to rigid-body motions of the configurations. Moreover, they have all been normalized so that their minimum values are invariant to dilations of the sample points. While J_{ee} emphasizes large errors (regardless of whether the distances δ_{ij} are large or small), J_{ff} emphasizes the large fractional errors (regardless of whether the errors $|d_{ij} - \delta_{ij}|$ are large or small). A useful compromise is J_{ef}, which emphasizes large products of error and fractional error.

Once a criterion function has been selected, an optimal configuration $\mathbf{y}_1, \ldots, \mathbf{y}_n$ is defined as one that minimizes that criterion function. Such a configuration can be sought by a standard gradient-descent procedure, starting with some initial configuration and changing the $\mathbf{y}_i$'s in the direction of greatest rate of decrease in the criterion function. Because the distance in the lower-dimensional space is given by $d_{ij} = \|\mathbf{y}_i - \mathbf{y}_j\|$, the gradient of d_{ij} with respect to $\mathbf{y}_i$ is merely a unit vector in the direction of $\mathbf{y}_i - \mathbf{y}_j$. Thus, the gradients of the criterion functions are easy to compute:

$$\nabla_{\mathbf{y}_k} J_{ee} = \frac{2}{\sum_{i<j} \delta_{ij}^2} \sum_{j \neq k} (d_{kj} - \delta_{kj}) \frac{\mathbf{y}_k - \mathbf{y}_j}{d_{kj}}$$

$$\nabla_{\mathbf{y}_k} J_{ff} = 2 \sum_{j \neq k} \frac{d_{kj} - \delta_{kj}}{\delta_{kj}^2} \frac{\mathbf{y}_k - \mathbf{y}_j}{d_{kj}}$$

$$\nabla_{\mathbf{y}_k} J_{ef} = \frac{2}{\sum_{i<j} \delta_{ij}} \sum_{j \neq k} \frac{d_{kj} - \delta_{kj}}{\delta_{kj}} \frac{\mathbf{y}_k - \mathbf{y}_j}{d_{kj}}.$$

The starting configuration can be chosen randomly, or in any convenient way that spreads the image points about. If the image points lie in a $\hat{d}$-dimensional space, then a simple and effective starting configuration can be found by selecting those $\hat{d}$ coordinates of the samples that have the largest variance.

The following example illustrates the kind of results that can be obtained by these techniques. The data consist of 30 points spaced at unit intervals along a spiral in three dimensions:

$$x_1(k) = \cos(k/\sqrt{2})$$
$$x_2(k) = \sin(k/\sqrt{2})$$
$$x_3(k) = k/\sqrt{2}, \qquad k = 0, 1, \ldots, 29.$$

Figure 10.27 shows the three-dimensional data. When the J_{ef} criterion was used, 20 iterations of a gradient descent procedure produced the two-dimensional configuration shown at the right. Of course, translations, rotations, and reflections of this configuration would be equally good solutions.

In nonmetric multidimensional scaling problems, the quantities δ_{ij} are dissimilarities whose numerical values are not as important as their rank order. An ideal configuration would be one for which the rank order of the distances d_{ij} is the same as the rank order of the dissimilarities δ_{ij}. Let us order the $m = n(n-1)/2$ dissimilarities so that $\delta_{i_1 j_1} \leq \cdots \leq \delta_{i_m j_m}$, and let $\hat{d}_{ij}$ be *any* m numbers satisfying the *monotonicity constraint*

MONOTONICITY CONSTRAINT

$$\hat{d}_{i_1 j_1} \leq \hat{d}_{i_2 j_2} \leq \cdots \leq \hat{d}_{i_m j_m}. \tag{110}$$

In general, the distances d_{ij} will not satisfy this constraint, and the numbers $\hat{d}_{ij}$ will not be distances. However, the degree to which the d_{ij} satisfy this constraint is measured by

$$\hat{J}_{mon} = \min_{\hat{d}_{ij}} \sum_{i<j} (d_{ij} - \hat{d}_{ij})^2, \tag{111}$$

where it is always to be understood that the $\hat{d}_{ij}$ must satisfy the monotonicity constraint. Thus, $\hat{J}_{mon}$ measures the degree to which the configuration of points $\mathbf{y}_1, \ldots, \mathbf{y}_n$ represents the original dissimilarities. Unfortunately, $\hat{J}_{mon}$ cannot be used to define an optimal configuration because it can be made to vanish by collapsing

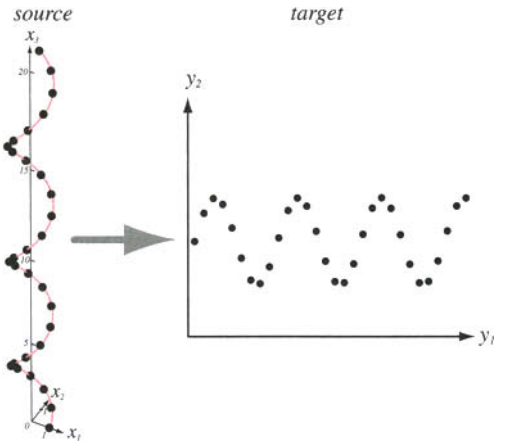


FIGURE 10.27. Thirty points of the form $\mathbf{x} = (\cos(k/\sqrt{2}), \sin(k/\sqrt{2}), k/\sqrt{2})^t$ for $k = 0, 1, \ldots, 29$ are shown at the left. Multidimensional scaling using the J_{ef} criterion (Eq. 109) and a two-dimensional target space leads to the image points shown at the right. This lower-dimensional representation shows clearly the fundamental sequential nature of the points in the original source space.

the configuration to a single point. However, this defect is easily removed by a normalization such as the following:

$$J_{mon} = \frac{\hat{J}_{mon}}{\sum_{i<j} d_{ij}^2}. \tag{112}$$

Thus, J_{mon} is invariant to translations, rotations, and dilations of the configuration, and an optimal configuration can be defined as one that minimizes this criterion function. It has been observed experimentally that when the number of points is larger than dimensionality of the image space, the monotonicity constraint is actually quite confining. This might be expected from the fact that the number of constraints grows as the square of the number of points, and it is the basis for the frequently encountered statement that this procedure allows the recovery of metric information from nonmetric data. The quality of the representation generally improves as the dimensionality of the image space is increased, and it may be necessary to go beyond three dimensions to obtain an acceptably small value of J_{mon}. However, this may be a small price to pay to allow the use of the many clustering procedures available for data points in metric spaces.

10.14.1 Self-Organizing Feature Maps

KOHONEN MAPS

A method closely related to multidimensional scaling is that of self-organizing feature maps, sometimes called topologically ordered maps or *Kohonen self-organizing feature maps*. As before, the goal is to represent all points in the source space by points in a target space, such that distance and proximity relationships are preserved as much as possible. The self-organizing map algorithm we shall discuss does not require the storage of a large number of samples, and thus it has much lower space complexity than multidimensional scaling. (In practice, both methods have high time complexities.) Moreover, the method is particularly useful when there is a nonlinear mapping inherent in the problem itself, as we shall see.

It is simplest to explain self-organizing maps by means of an example. Suppose we seek to learn a mapping from a circular disk region (the source space) to a linear target space, as shown in Fig. 10.28. The source space is sensed by a movable two-joint arm of fixed segment lengths; thus each point (x_1, x_2) in the disk area leads to a pair of angles (ϕ_1, ϕ_2), which we denote as a two-component vector $\boldsymbol{\phi}$. The algorithm uses a sequence of $\boldsymbol{\phi}$ values—but not the (x_1, x_2) values themselves, because they and their nonlinear transformation are not directly accessible. In our illustration the nonlinearity involves inverse trigonometric functions, but in most applications it is more complicated and not even known.

The task is this: Given a sequence of $\boldsymbol{\phi}$'s (corresponding to points sampled in the source space), create a mapping from $\boldsymbol{\phi}$ to $\mathbf{y}$ such that points neighboring in the source space are mapped to points that are neighboring in the target space. It is this goal of preserving neighborhoods that gives the resulting "topologically ordered maps" their name.

The mapping is learned by a simple two-layer neural network, here with two inputs (ϕ_1 and ϕ_2), fully connected to a large number of outputs, corresponding to points along the target line. When a pattern $\boldsymbol{\phi}$ is presented, each node in the target space computes its net activation, $net_k = \boldsymbol{\phi}^t \mathbf{w}_k$. One of the units is most activated; call it $\mathbf{y}^*$. The weights to this unit and those in its immediate neighborhood are updated according to:

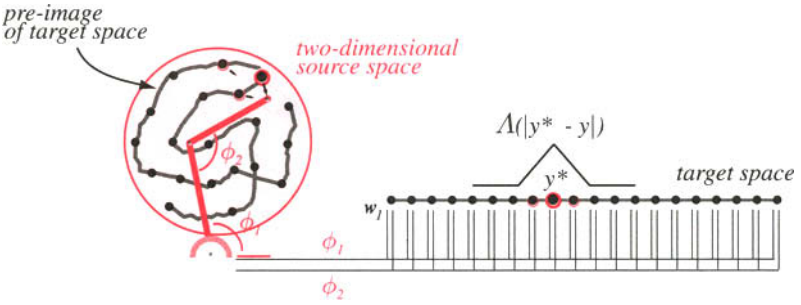


FIGURE 10.28. A self-organizing map from the (two-dimensional) disk source space to the (one-dimensional) line of the target space can be learned as follows. For each point y in the target line, there exists a corresponding point in the source space that, if sensed, would lead to y being most active. For clarity, then, we can link theses points in the source; it is as if the image line is placed in the source space. We call this the pre-image of the target space. At the state shown, the particular sensed point leads to y^* begin most active. The learning rule (Eq. 113) makes its source point move toward the sensed point, as shown by the small arrow. Because of the window function $\Lambda(|y^* - y|)$, the pre-image of points adjacent to y^* are also moved toward the sensed point, thought not as much. If such learning is repeated many times as the arm randomly senses the whole source space, a topologically correct map is learned.

$$w_{ki}(t+1) = w_{ki}(t) + \eta(t)\Lambda(|\mathbf{y} - \mathbf{y}^*|)(\boldsymbol{\phi}_i - w_{ki}(t)), \tag{113}$$

WINDOW FUNCTION

where $\eta(t)$ is a learning rate which depends upon the iteration number t. The function $\Lambda(|\mathbf{y} - \mathbf{y}^*|)$ is called the "window function" and has value 1.0 for $\mathbf{y} = \mathbf{y}^*$ and smaller for large values of $|\mathbf{y} - \mathbf{y}^*|$. The window function is vital to the success of the algorithm: It ensures that neighboring points in the target space have weights that are similar, and thus correspond to neighboring points in the source space, thereby ensuring topological neighborhoods (Fig. 10.29). Next, every weight vector is normalized such that $\|\mathbf{w}\| = 1$. (Naturally, only those weight vectors that have been altered during the learning trial need be renormalized.) The learning rate $\eta(t)$ decreases slowly as a function of iteration number (i.e., as patterns are presented) to ensure that learning will ultimately stop.

Equation 113 has a particularly straightforward interpretation. For each pattern presentation, the "winning" unit in the target space is adjusted so that it is more like the particular pattern. Others in the neighborhood of $\mathbf{y}^*$ are also adjusted so that their

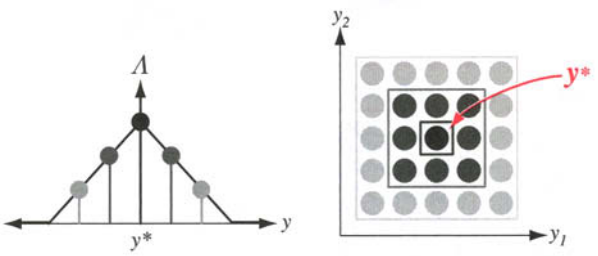


FIGURE 10.29. Typical window functions for self-organizing maps for target spaces in one dimension (left) and two dimensions (right). In each case, the weights at the maximally active unit, $\mathbf{y}^*$, in the target space get the largest weight update while units more distant get smaller update.

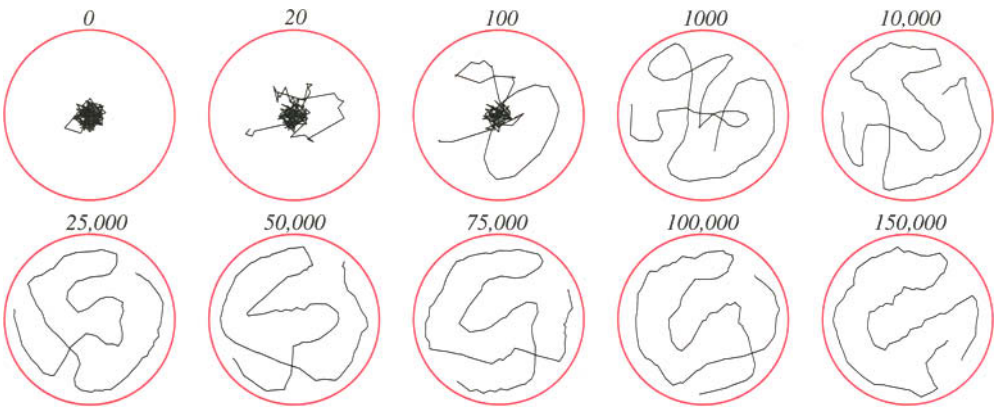


FIGURE 10.30. If a large number of pattern presentations are made using the setup of Fig. 10.28, a topologically ordered map develops. The number of pattern presentations is listed.

weights more nearly match that of the input pattern (though not quite as much as for $\mathbf{y}^*$, as given by the window function). In this way, neighboring points in the input space lead to neighboring points being active.

After a large number of pattern presentations, learning according to Eq. 113 ensures that neighboring points in the source space lead to neighboring points in the target space. Informally speaking, it is as if the target space line has been placed on the source space; learning pulls and stretches the line to fill the source space, as illustrated in Fig. 10.30, which shows the development of the map. After 150,000 training presentations, a topological map has been learned.

The learning of such self-organizing maps is very general and can be applied to virtually any source space, target space, and continuous nonlinear mapping. Figure 10.31 shows the development of a self-organizing map from a square source space to a square (grid) target space.

There are generally inherent ambiguities in the maps learned by this algorithm. For instance, a mapping from a square to a square could have eight possible orientations, corresponding to the four rotation and two flip symmetries. Such ambiguity

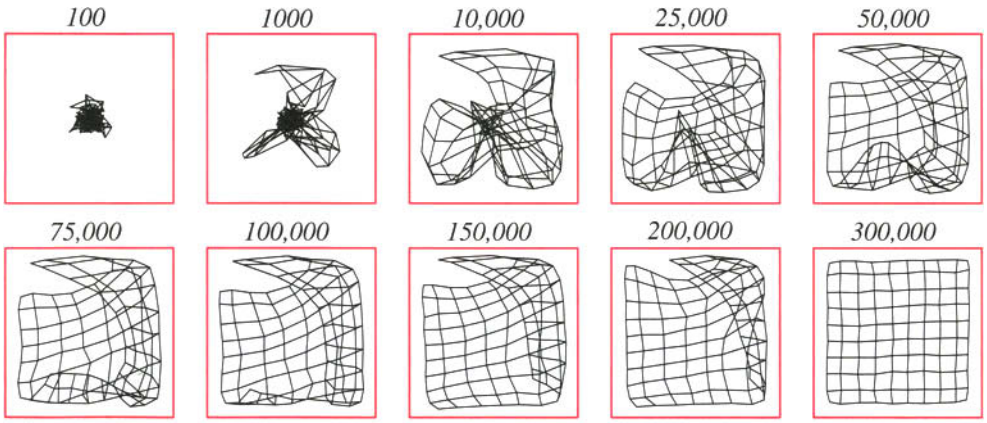


FIGURE 10.31. A self-organizing feature map from a square source space to a square (grid) target space. As in Fig. 10.28, each grid point of the target space is shown atop the point in the source space that maximally excites that target point.

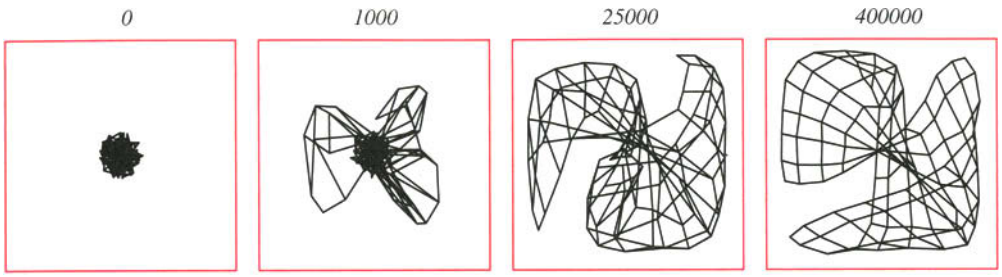


FIGURE 10.32. Some initial (random) weights and the particular sequence of patterns (randomly chosen) lead to kinks in the map; even extensive further training does not eliminate the kink. In such cases, learning should be restarted with randomized weights and possibly a wider window function and slower decay in learning.

is generally irrelevant for subsequent clustering or classification in the target space. Nevertheless, the mapping ambiguities are related to a more significant drawback—the possibility of "kinks" in the map. A particular initial condition can lead to *part* of the map learning one of the orientations, while a different part learns another one, as shown in Fig. 10.32. When this occurs, it is generally best to reinitialize the weights randomly and restart the learning with perhaps a wider window function or slower decay in the learning rate.

One of the benefits of this learning algorithm is that it naturally takes account of the probability of sampling in the source space, that is, $p(\mathbf{x})$. Regions of high such probability attract more of the points in the target space, and this means that more of the image points will be mapped there, as shown in Fig. 10.33.

Such self-organizing feature maps can be used in a number of systems. For instance, in a signal-processing application we can use the outputs of a bank of filters to map a waveform to a two-dimensional target space. When such an approach is applied to spoken vowel sounds, similar utterances such as /ee/ and /eh/ will be close together, while others, such as /ee/ and /oo/ will be far apart—just as we had in multidimensional scaling. Subsequent supervised learning can label regions in this target space and thus lead to a full classifier, but one formed using only a small amount of supervised training.

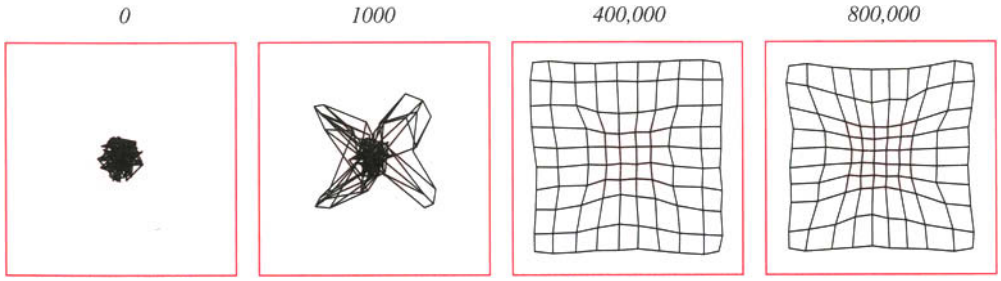


FIGURE 10.33. As in Fig. 10.31 except that the sampling of the input space was not uniform. In particular, the probability density for sampling a point in the central square region (pink) was 20 times greater than elsewhere. Notice that the final map devotes more nodes to this central region than in Fig. 10.31.

10.14.2 Clustering and Dimensionality Reduction

Because the curse of dimensionality plagues so many pattern recognition procedures, a variety of methods for dimensionality reduction have been proposed. Unlike the procedures that we have just examined, most of these methods provide a functional mapping, so that one can determine the image of an arbitrary feature vector. The classical procedures of statistics are principal components and *factor analysis*, both of which reduce dimensionality by forming linear combinations of the features. As we saw in Chapter 3 and Section 10.13.1, the object of principal components analysis is to find a lower-dimensional representation that accounts for the *variance* of the features. The object of factor analysis is to find a lower-dimensional representation that accounts for the *correlations* among the features. If we think of the problem as one of removing or combining (i.e., grouping) highly correlated features, then it becomes clear that the techniques of clustering are applicable to this problem. In terms of the *data matrix*, whose n rows are the d-dimensional samples, ordinary clustering can be thought of as a grouping of the rows, with a smaller number of *cluster centers* being used to represent the data, whereas dimensionality reduction can be thought of as a grouping of the columns, with *combined features* being used to represent the data.

FACTOR ANALYSIS

DATA MATRIX

Let us consider a simple modification of hierarchical clustering to reduce dimensionality. In place of an n-by-n matrix of distances between samples, we consider a d-by-d *correlation matrix* $\mathbf{R} = [\rho_{ij}]$, where the correlation coefficient ρ_{ij} is related to the covariances (or sample covariances) by

CORRELATION MATRIX

$$\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}. \tag{114}$$

Because $0 \leq \rho_{ij}^2 \leq 1$, with $\rho_{ij}^2 = 0$ for uncorrelated features and $\rho_{ij}^2 = 1$ for completely correlated features, ρ_{ij}^2 plays the role of a similarity function for features. Two features for which ρ_{ij}^2 is large are clearly good candidates to be merged into one feature, thereby reducing the dimensionality by one. Repetition of this process leads to the following hierarchical procedure:

■ **Algorithm 8. (Hierarchical Dimensionality Reduction)**

```
1  begin initialize d', D_i ← {x_i}, i = 1, ..., d
2        d̂ ← d + 1
3        do d̂ ← d̂ − 1
4           compute R by Eq. 114
5           find most correlated distinct clusters, say D_i and D_j
6           D_i ← D_i ∪ D_j merge
7           delete D_j
8        until d̂ = d'
9     return d' clusters
10 end
```

Probably the simplest way to merge two groups of features is just to average them. (This tacitly assumes that the features have been scaled so that their numerical ranges are comparable.) With this definition of a new feature, there is no problem in defining the correlation matrix for groups of features. It is not hard to think of variations on this general theme, but we shall not pursue this topic further.

For the purposes of pattern *classification*, the most serious criticism of all of the approaches to dimensionality reduction that we have mentioned is that they are overly concerned with faithful *representation* of the data. Greatest emphasis is usually placed on those features or groups of features that have the greatest variability. But for classification, we are interested in *discrimination*—not representation. While it is a truism that the ideal representation is the one that makes classification easy, it is not always so clear that clustering without explicitly incorporating classification criteria will find such a representation. Of course, even if an algorithm finds clear, isolated clusters, there is no guarantee that these will be useful for classification; after all, each isolated cluster might consist of points from the different categories highly interspersed. Roughly speaking, the most interesting features are the ones for which the difference in the class means is large relative to the standard deviations, not the ones for which merely the standard deviations are large. In short, for classification we are interested in something more like the method of multiple discriminant analysis described in Chapter 3.

There is a large body of theory on methods of dimensionality reduction for pattern classification. Some of these methods seek to form new features out of linear combinations of old ones. Others seek merely a smaller subset of the original features. A major problem confronting this theory is that the division of pattern recognition into feature extraction followed by classification is theoretically artificial. A completely optimal feature extractor can never be anything but an optimal classifier. It is only when constraints are placed on the classifier or limitations are placed on the size of the set of samples that one can formulate nontrivial problems. Various ways of circumventing this problem that may be useful under the proper circumstances can be found in the literature. When it is possible to exploit knowledge of the problem domain to obtain more informative features, that is usually the most profitable course of action.

SUMMARY

Unsupervised learning and clustering seek to extract information from unlabeled samples. If the underlying distribution comes from a mixture of component densities described by a set of unknown parameters $\boldsymbol{\theta}$, then $\boldsymbol{\theta}$ can be estimated by Bayesian or maximum-likelihood methods. A more general approach is to define some measure of similarity between two clusters, as well as a global criterion such as a sum-squared-error or trace of a scatter matrix. Because there are only occasionally analytic methods for computing the clustering which optimizes the criterion, a number of greedy (*locally* stepwise optimal) iterative algorithms can be used, such as k-means and fuzzy k-means clustering.

If we seek to reveal structure in the data at many levels—that is, clusters with subclusters and subsubclusters—then hierarchical methods are needed. Agglomerative or bottom-up methods start with each sample as a singleton cluster and iteratively merge clusters that are "most similar" according to some chosen similarity or distance measure. Conversely, divisive or top-down methods start with a single cluster representing the full data set and iteratively split them into smaller clusters, each time seeking the subclusters that are most dissimilar. The resulting hierarchical structure is revealed in a dendrogram. A large disparity in the similarity measure for successive cluster levels in a dendrogram usually indicates the "natural" number of clusters. Alternatively, the problem of cluster validity—knowing the proper number of clusters—can also be addressed by hypothesis testing. In that case the null hypoth-

esis is that there are some number c of clusters; we then determine if the reduction of the cluster criterion due to an additional cluster is statistically significant.

Competitive learning is a neural network clustering algorithm in which the cluster center most similar to an input pattern is modified to become more like that pattern. In order to guarantee that learning stops, the learning rate must decay. Competitive learning can be modified to allow for the creation of new cluster centers if no center is sufficiently similar to a particular input pattern, as in leader-follower clustering and adaptive resonance. While these methods have many advantages, such as computational ease and ability to track gradual variations in the data, they rarely optimize an easily specified global criterion such as sum-of-squared error. Unsupervised learning and clustering algorithms are often sensitive to a number of critical user-selected parameters.

Graph-theoretic methods in clustering treat the data as points, to be linked based on a number of heuristics and distance measures. The clusters produced by these methods can exhibit chaining or other intricate structures, but they rarely optimize an easily specified global cost function. Graph methods are, moreover, generally more sensitive to details of the data.

Component analysis seeks directions or axes in feature space that provide an improved, lower-dimensional representation for the full data space. In principal component analysis, which is a linear process, such directions are merely the eigenvectors of the covariance matrix having the largest eigenvalues; this projection optimizes a sum-squared-error criterion. Nonlinear component analysis, for instance as learned in an internal layer an auto-encoder neural network, yields curved surfaces embedded in the full d-dimensional feature space, onto which an arbitrary pattern $\mathbf{x}$ is projected. The goal in independent component analysis—which is developed through gradient descent in an entropy criterion—is to determine the directions in feature space that are statistically most independent. Such directions may reveal the true sources (if they are indeed independent) and can be used for segmentation and blind source separation.

Self-organizing feature maps and multidimensional scaling are two general methods for dimensionality reduction. Self-organizing feature maps can be highly nonlinear, and represents points close in the source space by points close in the lower-dimensional target space. In preserving neighborhoods in this way, such maps are also said to be "topologically correct." The source and target spaces can be of very general shapes, and the mapping will depend upon the the distribution of samples within the source space. Multidimensional scaling similarly learns a correspondence between points that preserves neighborhoods, and is often used for data visualization. Because the basic method requires all the inter-point distances for minimizing a global criterion function, its space complexity limits the usefulness of multidimensional scaling to problems of moderate size.

BIBLIOGRAPHICAL AND HISTORICAL REMARKS

Historically, the literature on unsupervised learning and clustering can be traced back to Karl Pearson, who in 1894 used sample moments to determine the parameters in a mixture of two univariate Gaussians. While most books on pattern classification address unsupervised learning, there are several modern books and review articles on unsupervised learning that go into great detail, such as references [1] and [23]. Much of the mathematical analysis of unsupervised methods comes from the signal compression community, where vector quantization (VQ) seeks to represent an arbi-

trary vector by one of c "prototype" vectors, which correspond to our cluster centers, as described in reference [18].

A clear book on mixture models is reference [34] and the issue of identifiability in unsupervised learning is covered in reference [46]. In reference [20], Hasselblad showed how the parameters of one-dimensional normals could be learned in an unsupervised environment. The k-means algorithm was introduced in a paper by Lloyd [32], which inspired many variations, such as the use of the Mahalanobis distance [33] or use of "fuzzy" measures [5, 6]. Efficient agglomerative methods for hierarchical clustering are summarized in reference [12]. Cladistics, the classificatory foundation of biology (from the Greek *klados*, branch), provides useful background for the use of classification in all scientific fields [27]. A good introduction to methods of hierarchical clustering is reference [26].

The key mathematical concepts underlying principal component analysis appear in references [9, 24, 13] and [30]. Independent component analysis was introduced by Jutten and Herault [25], and the maximum-likelihood approach was introduced by Gaeta and Lacoume [16]. Generalizations and a maximum-likelihood approach are given in references [39] and [40]. Bell and Sejnowski [4] showed a neural network implementation of independent component analysis, and in reference [36] explained the very close relationship between ICA and information maximization. A good compendium on ICA is reference [48]; several studies have shown the benefits of the technique for classification, such as reference [15]. Multidimensional scaling, occassionally called nonlinear projection in distinction to linear projection, is discussed in references [8] and [43] and its relationship to clustering is explored in reference [31].

Kohonen's long series of papers on self-organizing feature maps began in the early 1980s [28], and good compendia can be found in references [29] and [42]; convergence properties of algorithms for self-organizing feature maps are proved in reference [49]. A good comparison of self-organizing maps with other methods such as principal component analysis and discriminant analysis is reference [33]. There have been numerous applications of the method, from speech to finding patterns of poverty in the world.

The main emphasis of research on Adaptive Resonance has been to explore pattern recognition and clustering in biological systems, as discussed in reference [10]. A wonderfully clear exposition of the central algorithmic ideas is reference [35]; an attempt to translate the ideas and terminology of adaptive resonance to standard engineering terminology, complete with a glossary, is reference [45].

PROBLEMS

Section 10.2

1. Suppose that x can assume the values $0, 1, \ldots, m$ and that $P(x|\boldsymbol{\theta})$ is a mixture of c binomial distributions

$$P(x|\boldsymbol{\theta}) = \sum_{j=1}^{c} \binom{m}{x} \theta_j^m (1-\theta_j)^{m-x} P(\omega_j),$$

where $\boldsymbol{\theta}$ is a vector of length c representing the parameters in the distributions.

(a) Assuming that the prior probabilities $P(\omega_j)$ are known, explain why this mixture is not identifiable if $m < c$.

(b) Under these conditions, is the mixture *completely* unidentifiable?

(c) How do your answers above change if the prior probabilities are also unknown?

2. Consider a mixture distribution of two triangle distributions, where component density ω_i is centered on μ_i and has "halfwidth" w_i, according to

$$p(x|\omega_i) \sim T(\mu_i, w_i) = \begin{cases} (w_i - |x - \mu_i|)/w_i^2 & \text{for } |x - \mu_i| < w_i \\ 0 & \text{otherwise.} \end{cases}$$

(a) Assume $P(\omega_1) = P(\omega_2) = 0.5$ and derive the equations for the maximum-likelihood values $\hat{\mu}_i$ and $\hat{w}_i$, $i = 1, 2$.

(b) Under the conditions in part (a), is the distribution identifiable?

(c) Assume that both widths w_i are known, but the centers are not. Assume, too, that there exist values for the centers that give nonzero probability to each of the samples. Derive a formula for the maximum-likelihood value of the centers.

(d) Under the conditions in part (c), is the distribution identifiable?

3. Suppose there is a one-dimensional mixture density consisting of two Gaussian components, each centered on the origin:

$$p(x|\boldsymbol{\theta}) = P(\omega_1)\frac{1}{\sqrt{2\pi}\sigma_1}e^{-x^2/(2\sigma_1^2)} + (1 - P(\omega_1))\frac{1}{\sqrt{2\pi}\sigma_2}e^{-x^2/(2\sigma_2^2)},$$

and $\boldsymbol{\theta} = (P(\omega_1), \sigma_1, \sigma_2)^t$ describes the parameters.

(a) Show that under these conditions this density is completely unidentifiable.

(b) Suppose the value $P(\omega_1)$ is fixed and known. Is the model identifiable?

(c) Suppose σ_1 and σ_2 are known, but $P(\omega_1)$ is unknown. Is this resulting model identifiable? That is, can $P(\omega_1)$ be identified using data?

Section 10.3

4. Let $\mathbf{x}$ be a d-component binary vector (0,1) and let $P(\mathbf{x}|\boldsymbol{\theta})$ be a mixture of c multivariate Bernoulli distributions,

$$P(\mathbf{x}|\boldsymbol{\theta}) = \sum_{i=1}^{c} P(\mathbf{x}|\omega_i, \boldsymbol{\theta}_i)P(\omega_i)$$

where

$$P(\mathbf{x}|\omega_i, \boldsymbol{\theta}_i) = \prod_{j=1}^{d} \theta_{ij}^{x_j}(1 - \theta_{ij})^{1-x_j}.$$

(a) Derive the formula for the partial derivative:

$$\frac{\partial \ln P(\mathbf{x}|\omega_i, \boldsymbol{\theta}_i)}{\partial \theta_{ij}} = \frac{\mathbf{x}_i - \theta_{ij}}{\theta_{ij}(1 - \theta_{ij})}.$$

(b) Using the general equations for maximum-likelihood estimates, show that the maximum-likelihood estimate $\hat{\boldsymbol{\theta}}_i$ for $\boldsymbol{\theta}_i$ must satisfy

$$\hat{\boldsymbol{\theta}}_i = \frac{\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})\mathbf{x}_k}{\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})}.$$

(c) Interpret your answer to part (b) in words.

5. Let $p(\mathbf{x}|\boldsymbol{\theta})$ be a c-component normal mixture with $p(\mathbf{x}|\omega_i, \boldsymbol{\theta}_i) \sim N(\boldsymbol{\mu}_i, \sigma_i^2\mathbf{I})$. Using the results of Section 10.3, show that the maximum-likelihood estimate for σ_i^2 must satisfy

$$\hat{\sigma}_i^2 = \frac{1/d \sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}_i)\|\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i\|^2}{\sum_{k=1}^{m} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}_i)},$$

where $\hat{\boldsymbol{\mu}}_i$ and $\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}_i)$ are given by Eqs. 25 and 27, respectively.

6. Consider a c-component mixture model with unknown parameter vector $\boldsymbol{\theta}$ and unknown prior probabilities $P(\omega_i)$. Let $\hat{P}(\omega_i)$ be the maximum-likelihood estimate for $P(\omega_i)$, and let $\hat{\boldsymbol{\theta}}_i$ be the maximum-likelihood estimate for $\boldsymbol{\theta}_i$. Show that if the likelihood function is differentiable and if $\hat{P}(\omega_i) \neq 0$ for any i, then $\hat{P}(\omega_i)$ and $\hat{\boldsymbol{\theta}}_i$ must satisfy Eqs. 11 and 12, that is,

$$\hat{P}(\omega_i) = \frac{1}{n} \sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})$$

and

$$\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) \nabla_{\boldsymbol{\theta}_i} \ln p(\mathbf{x}_k|\omega_i, \hat{\boldsymbol{\theta}}_i) = 0,$$

where

$$\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) = \frac{p(\mathbf{x}_k|\omega_i, \hat{\boldsymbol{\theta}}_i)\hat{P}(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}_k|\omega_j, \hat{\boldsymbol{\theta}}_j)\hat{P}(\omega_j)}.$$

7. The derivation of the equations for maximum-likelihood estimation of parameters of a mixture density was made under the assumption that the parameters in each component density are functionally independent. Suppose instead that

$$p(\mathbf{x}|\alpha) = \sum_{j=1}^{c} p(\mathbf{x}|\omega_j, \alpha)P(\omega_j),$$

where α is a parameter that appears in *several* (and possibly all) of the component densities. Let l be the n-sample log-likelihood function. Show that the derivative of l with respect to that parameter is

$$\frac{\partial l}{\partial \alpha} = \sum_{k=1}^{n} \sum_{j=1}^{c} P(\omega_j|\mathbf{x}_k, \alpha) \frac{\partial \ln p(\mathbf{x}_k|\omega_j, \alpha)}{\partial \alpha},$$

where

$$P(\omega_j|\mathbf{x}_k, \alpha) = \frac{p(\mathbf{x}_k|\omega_j, \alpha)P(\omega_j)}{p(\mathbf{x}_k|\alpha)}.$$

8. Let θ_1 and θ_2 be unknown parameters for the component densities $p(x|\omega_1, \theta_1)$ and $p(x|\omega_2, \theta_2)$, respectively. Assume that θ_1 and θ_2 are initially statistically independent, so that $p(\theta_1, \theta_2) = p_1(\theta_1)p_2(\theta_2)$.

   (a) Show that after one sample x_1 from the mixture density is observed, $p(\theta_1, \theta_2|x_1)$ can no longer be factored as $p_1(\theta_1|x_1)p_2(\theta_2|x_1)$ if

$$\frac{\partial p(x|\omega_i, \theta_i)}{\partial \theta_i} \neq 0, \qquad i = 1, 2.$$

   (b) What does this imply in general about the statistical dependence of parameters in unsupervised learning?

9. Assume that a mixture density $p(\mathbf{x}|\boldsymbol{\theta})$ is identifiable. Prove that under very general conditions, $p(\boldsymbol{\theta}|\mathcal{D}^n)$ converges (in probability) to a Dirac delta function centered at the true value of $\boldsymbol{\theta}$ as the number of samples becomes very large.

10. Assume that the likelihood function of Eq. 3 is differentiable and derive the maximum-likelihood conditions of Eqs. 11–13.

Section 10.4

11. Let $p(\mathbf{x}|\omega_i, \boldsymbol{\theta}_i) \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is a common covariance matrix for the c component densities. Let σ_{pq} be the pqth element of $\boldsymbol{\Sigma}$, let σ^{pq} be the pqth element of $\boldsymbol{\Sigma}^{-1}$, let $x_p(k)$ be the pth element of $\mathbf{x}_k$, and let $\mu_p(i)$ be the pth element of $\boldsymbol{\mu}_i$.

   (a) Show that

$$\frac{\partial \ln p(\mathbf{x}_k|\omega_i, \boldsymbol{\theta}_i)}{\partial \sigma^{pq}} = \left(1 - \frac{\delta_{pq}}{2}\right)\left[\sigma_{pq} - (x_p(k) - \mu_p(i))(x_q(k) - \mu_q(i))\right],$$

   where

$$\delta_{pq} = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$$

   is the Kroneker delta symbol.

   (b) Use this result and the results of Problem 7 to show that the maximum-likelihood estimate for $\boldsymbol{\Sigma}$ must satisfy

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{k=1}^{n} \mathbf{x}_k \mathbf{x}_k^t - \sum_{i=1}^{c} \hat{P}(\omega_i)\hat{\boldsymbol{\mu}}_i \hat{\boldsymbol{\mu}}_i^t,$$

   where $\hat{P}(\omega_i)$ and $\hat{\boldsymbol{\mu}}_i$ are the maximum-likelihood estimates given by Eqs. 24 and 25.

12. Show that the maximum-likelihood estimate of a prior probability can be zero by considering the following special case. Let $p(x|\omega_1) \sim N(0, 1)$ and $p(x|\omega_2) \sim N(0, 1/2)$, so that $P(\omega_1)$ is the only unknown parameter in the mixture

$$p(x) = \frac{P(\omega_1)}{\sqrt{2\pi}} e^{-x^2/2} + \frac{(1 - P(\omega_1))}{\sqrt{\pi}} e^{-x^2}.$$

   (a) Show that the maximum-likelihood estimate $\hat{P}(\omega_1)$ of $P(\omega_1)$ is zero if one sample x_1 is observed and if $x_1^2 < \ln 2$.

(b) What is the value of $\hat{P}(\omega_1)$ if $x_1^2 > \ln 2$?

(c) Summarize and interpret your answer in words.

13. Consider the univariate normal mixture

$$p(x|\mu_1, \ldots, \mu_c) = \sum_{j=1}^{c} \frac{P(\omega_j)}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu_j}{\sigma}\right)^2\right]$$

in which all of the c components have the same known variance σ^2. Suppose that the means are so far apart compared to σ that for any observed x all but one of the terms in this sum are negligible. Use a heuristic argument to show that the value of

$$\max_{\mu_1,\ldots,\mu_c} \left\{\frac{1}{n} \ln p(x_1, \ldots, x_n|\mu_1, \ldots, \mu_c)\right\}$$

ought to be approximately

$$\sum_{j=1}^{c} P(\omega_j) \ln P(\omega_j) - \frac{1}{2} \ln[2\pi\sigma e]$$

when the number n of independently drawn samples is large. (Here e is the base of the natural logarithms.)

14. Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be n d-dimensional samples and $\mathbf{\Sigma}$ be any nonsingular d-by-d matrix. Show that the vector $\mathbf{x}$ that minimizes

$$\sum_{k=1}^{m} (\mathbf{x}_k - \mathbf{x})^t \mathbf{\Sigma}^{-1} (\mathbf{x}_k - \mathbf{x})$$

is the sample mean, $\bar{\mathbf{x}} = 1/n \sum_{k=1}^{n} \mathbf{x}_k$.

15. Perform the differentiation in Eq. 31 to derive Eqs. 32 and 33.

16. Show that the computational complexity of Algorithm 1 is $O(ndcT)$, where n is the number of d-dimensional patterns, c is the assumed number of clusters, and T is the number of iterations.

17. Fill in the steps of the derivation of Eqs. 24–26. Be sure to state any assumptions you invoke.

Section 10.5

18. Consider the combinatorics of exhaustive inspection of clusters of n samples into c clusters.

(a) Show that there are exactly

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

such distinct clusterings.

(b) How many clusters are there for $n = 100$ and $c = 5$?

(c) Find an approximation for your answer to (a) for the case $n \gg c$. Use your answer to estimate the number of clusterings of 1000 points into 10 clusters.

Section 10.6

19. Prove that the ranking of distances between samples discussed in Section 10.6 is invariant to any monotonic transformation of the dissimilarity values. Do this as follows:

(a) Define the *value* v_k for the clustering at level k, and for level 1 let $v_1 = 0$. For all higher levels, v_k is the minimum dissimilarity between pairs of distinct clusters at level $k-1$. Explain why with both δ_{min} and δ_{max} the value v_k either stays the same or increases as k increases.

(b) Assume that no two of the n samples are identical, so that $v_2 > 0$. Use this to prove monotonicity—that is, that $0 = v_1 \leq v_2 \leq v_3 \leq \cdots \leq v_n$.

Section 10.7

20. Derive Eq. 55 from Eq. 54 using the definition given in Eq. 56.

21. If a set of n samples $\mathcal{D}$ is partitioned into c disjoint subsets $\mathcal{D}_1, \ldots, \mathcal{D}_c$, the sample mean $\mathbf{m}_i$ for samples in $\mathcal{D}_i$ is undefined if $\mathcal{D}_i$ is empty. In such a case, the sum-of-squared errors involves only the nonempty subsets:

$$J_e = \sum_{\mathcal{D}_i \neq \emptyset} \sum_{\mathbf{x} \in \mathcal{D}_i} \|\mathbf{x} - \mathbf{m}_i\|^2.$$

Assuming that $n \geq c$, show there are no empty subsets in a partition that minimizes J_e. Explain your answer in words.

22. Consider a set of $n = 2k + 1$ samples, k of which coincide at $x = -2$, k at $x = 0$, and one at $x = a > 0$.

(a) Show that the two-cluster partitioning that minimizes J_e groups the k samples at $x = 0$ with the one at $x = a$ if $a^2 < 2(k+1)$.

(b) What is the optimal grouping if $a^2 > 2(k+1)$?

23. Let $\mathbf{x}_1 = \binom{4}{5}$, $\mathbf{x}_2 = \binom{1}{4}$, $\mathbf{x}_3 = \binom{0}{1}$, and $\mathbf{x}_4 = \binom{5}{0}$, and consider the following three partitions:

1. $\mathcal{D}_1 = \{\mathbf{x}_1, \mathbf{x}_2\}, \mathcal{D}_2 = \{\mathbf{x}_3, \mathbf{x}_4\}$
2. $\mathcal{D}_1 = \{\mathbf{x}_1, \mathbf{x}_4\}, \mathcal{D}_2 = \{\mathbf{x}_2, \mathbf{x}_3\}$
3. $\mathcal{D}_1 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}, \mathcal{D}_2 = \{\mathbf{x}_4\}$

Show that by the sum-of-square error J_e criterion (Eq. 54), the third partition is favored, whereas by the invariant J_d (Eq. 68) criterion the first two partitions are favored.

24. Let $\mathbf{x}_1 = \binom{0}{0}$, $\mathbf{x}_2 = \binom{1}{1}$, $\mathbf{x}_3 = \binom{1}{0}$, and $\mathbf{x}_4 = \binom{2}{0.5}$, and consider the following three partitions:

1. $\mathcal{D}_1 = \{\mathbf{x}_1, \mathbf{x}_2\}, \mathcal{D}_2 = \{\mathbf{x}_3, \mathbf{x}_4\}$
2. $\mathcal{D}_1 = \{\mathbf{x}_1, \mathbf{x}_4\}, \mathcal{D}_2 = \{\mathbf{x}_2, \mathbf{x}_3\}$
3. $\mathcal{D}_1 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}, \mathcal{D}_2 = \{\mathbf{x}_4\}$

(a) Find the clustering that minimizes the sum-of-squared error criterion, J_e (Eq. 54).

(b) Find the clustering that minimizes the determinant criterion, J_d (Eq. 68).

25. Consider the problem of invariance to transformation of the feature space.

(a) Show that the eigenvalues $\lambda_1, \ldots, \lambda_d$ of $\mathbf{S}_W^{-1}\mathbf{S}_B$ are invariant to nonsingular linear transformations of the data.

(b) Show that the eigenvalues $\nu_1, \ldots, \nu_d$ of $\mathbf{S}_T^{-1}\mathbf{S}_W$ are related to those of $\mathbf{S}_W^{-1}\mathbf{S}_B$ by $\nu_i = 1/(1+\lambda_i)$.

(c) Use your above results to show that $J_d = |\mathbf{S}_W|/|\mathbf{S}_T|$ is invariant to nonsingular linear transformations of the data.

26. Recall the definitions of the within-cluster and the between-cluster scatter matrices (Eqs. 62 and 63). Define the total scatter matrix to be $\mathbf{S}_T = \mathbf{S}_W + \mathbf{S}_B$. Show that the following measures (Eqs. 70 and 71) are invariant to linear transformations of the data.

(a) $\mathrm{tr}[\mathbf{S}_T^{-1}\mathbf{S}_W] = \sum_{i=1}^{d} \frac{1}{1+\lambda_i}$

(b) $|\mathbf{S}_W|/|\mathbf{S}_T| = \prod_{i=1}^{d} \frac{1}{1+\lambda_i}$

(c) $|\mathbf{S}_W^{-1}\mathbf{S}_B| = \prod_{i=1}^{d} \lambda_i$

(d) What is the typical value of the criterion in (c)? Why, therefore, is that criterion not very useful?

27. Show that the clustering criterion J_d in Eq. 68 is invariant to linear transformations of the space as follows. Let $\mathbf{T}$ be a nonsingular matrix and consider the change of variables $\mathbf{x}' = \mathbf{T}\mathbf{x}$.

(a) Write the new mean vectors $\mathbf{m}_i'$ and scatter matrices $\mathbf{S}_i'$ in terms of the old values and $\mathbf{T}$.

(b) Calculate J_d' in terms of the (old) J_d and show that they differ solely by an overall scalar factor.

(c) Because this factor is the same for all partitions, argue that J_d and J_d' rank the partitions in the same way, and hence that the optimal clustering based on J_d is invariant to nonsingular linear transformations of the data.

28. The eigenvalues $\lambda_1, \ldots, \lambda_d$ of $\mathbf{S}_W^{-1}\mathbf{S}_B$ are the basic linear invariants of the scatter matrices. Show that the eigenvalues are indeed invariant under nonsingular linear transformations of the data.

29. Consider the problems that might arise when using the determinant criterion for clustering.

(a) Show that the rank of the within-cluster scatter matrix $\mathbf{S}_i$ cannot exceed $n_i - 1$, and thus the rank of $\mathbf{S}_W$ cannot exceed $\sum_{i=1}^{c}(n_i - 1) = n - c$.

(b) Use your answer to explain why the between-cluster scatter matrix $\mathbf{S}_B$ may become singular. (Of course, if the samples are confined to a lower-dimensional subspace, it is possible to have $\mathbf{S}_W$ be singular even though $n - c \geq d$.)

Section 10.8

30. One way to generalize the basic-minimum-squared-error procedure is to define the criterion function

$$J_T = \sum_{i=1}^{c} \sum_{\mathbf{x} \in \mathcal{D}_i} (\mathbf{x} - \mathbf{m}_i)^t \mathbf{S}_T^{-1} (\mathbf{x} - \mathbf{m}_i),$$

where $\mathbf{m}_i$ is the mean of the n_i samples in $\mathcal{D}_i$ and $\mathbf{S}_T$ is the total scatter matrix.

(a) Show that J_T is invariant to nonsingular linear transformations of the data.

(b) Show that the transfer of a sample $\hat{\mathbf{x}}$ from $\mathcal{D}_i$ to $\mathcal{D}_j$ causes J_T to change to

$$J_T^* = J_T + \left[\frac{n_j}{n_j + 1} (\hat{\mathbf{x}} - \mathbf{m}_j)^t \mathbf{S}_T^{-1} (\hat{\mathbf{x}} - \mathbf{m}_j) - \frac{n_i}{n_i - 1} (\hat{\mathbf{x}} - \mathbf{m}_i)^t \mathbf{S}_T^{-1} (\hat{\mathbf{x}} - \mathbf{m}_i) \right].$$

(c) Using this result, write pseudocode for an iterative procedure for minimizing J_T.

31. Consider how the transfer of a single point from one cluster to another affects the mean and sum-squared error, and thereby derive Eqs. 76 and 77.

Section 10.9

32. Let a similarity measure be defined as $s(\mathbf{x}, \mathbf{x}') = \mathbf{x}^t \mathbf{x}' / (\|\mathbf{x}\| \|\mathbf{x}'\|)$.

(a) Interpret this similarity measure if the d features have binary values, where $x_i = 1$ if $\mathbf{x}$ possesses the ith feature and $x_i = -1$ if it does not.

(b) Show that for this case the squared Euclidean distance satisfies

$$\|\mathbf{x} - \mathbf{x}'\|^2 = 2d(1 - s(\mathbf{x}, \mathbf{x}')).$$

33. Let patterns $\mathbf{x}$ and $\mathbf{x}'$ be arbitrary points in a d-dimensional space, and let q be a scalar parameter ($q > 1$). For each of the measures shown, state whether or not it represents a metric, and whether or not it represents an ultrametric.

(a) $s(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2$ (squared Euclidean)

(b) $s(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|$ (Euclidean)

(c) $s(\mathbf{x}, \mathbf{x}') = \left(\sum_{k=1}^{d} |x_k - x_k'|^q \right)^{1/q}$ (Minkowski)

(d) $s(\mathbf{x}, \mathbf{x}') = \mathbf{x}^t \mathbf{x}' / (\|\mathbf{x}\| \|\mathbf{x}'\|)$ (cosine)

(e) $s(\mathbf{x}, \mathbf{x}') = \mathbf{x}^t \mathbf{x}'$ (dot product)

34. Let cluster $\mathcal{D}_i$ contain n_i samples, and let d_{ij} be some measure of the distance between two clusters $\mathcal{D}_i$ and $\mathcal{D}_j$. In general, one might expect that if $\mathcal{D}_i$ and $\mathcal{D}_j$ are merged to form a new cluster $\mathcal{D}_k$, then the distance from $\mathcal{D}_k$ to some other cluster $\mathcal{D}_h$ is not simply related to d_{hi} and d_{hj}. However, consider the equation

$$d_{hk} = \alpha d_{hi} + \alpha_i d_{hj} + \beta d_{ij} + \gamma |d_{hi} - d_{hj}|.$$

Show that the following choices for the coefficients α_i, α_j, β, and γ lead to the distance functions indicated.

(a) $d_{min} : \alpha_i = \alpha_j = 0.5, \beta = 0, \gamma = -0.5.$

(b) $d_{max} : \alpha_i = \alpha_j = 0.5, \beta = 0, \gamma = +0.5.$

(c) $d_{avg} : \alpha_i = \frac{n_i}{n_i + n_j}, \alpha_j = \frac{n_j}{n_i + n_j}, \beta = \gamma = 0.$

(d) $d^2_{mean} : \alpha_i = \frac{n_i}{n_i + n_j}, \alpha_j = \frac{n_j}{n_i + n_j}, \beta = -\alpha_i\alpha_j, \gamma = 0.$

35. Consider a hierarchical clustering procedure in which clusters are merged so as to produce the smallest increase in the sum-of-squared error at each step. If the ith cluster contains n_i samples with sample mean $\mathbf{m}_i$, show that the smallest increase results from merging the pair of clusters for which

$$\frac{n_i n_j}{n_i + n_j} \|\mathbf{m}_i - \mathbf{m}_j\|^2$$

is minimum.

36. Assume we are clustering using the sum-of-squared error criterion J_e (Eq. 54). Show that a "distance" measure between clusters can be derived, Eq. 83, such that merging the "closest" such clusters increases J_e as little as possible.

37. Create by hand a dendrogram for the following eight points in one dimension: $\{-5.5, -4.1, -3.0, -2.6, 10.1, 11.9, 12.3, 13.6\}$. Define the similarity between two clusters to be $20 - d_{min}(\mathcal{D}_i, \mathcal{D}_j)$, where $d_{min}(\mathcal{D}_i, \mathcal{D}_j)$ is given in Eq. 79. Based on your dendrogram, argue that two is the natural number of clusters.

38. Create by hand a dendrogram for the following 10 points in one dimension: $\{-2.2, -2.0, -0.3, 0.1, 0.2, 0.4, 1.6, 1.7, 1.9, 2.0\}$. Define the similarity between to clusters to be $20 - d_{min}(\mathcal{D}_i, \mathcal{D}_j)$, where $d_{min}(\mathcal{D}_i, \mathcal{D}_j)$ is given in Eq. 79. Based on your dendrogram, argue that three is the natural number of clusters.

39. Assume that the nearest-neighbor cluster algorithm has been allowed to continue fully, thereby giving a tree with a path from any node to any other node. Show that the sum of the edge lengths of this resulting tree will not exceed the sum of the edge lengths for any other spanning tree for that set of samples.

Section 10.10

40. Assume that a large number n of d-dimensional samples has been chosen from a multidimensional Gaussian—that is, $p(\mathbf{x}) \sim N(\mathbf{m}, \mathbf{\Sigma})$, where $\mathbf{\Sigma}$ is an arbitrary positive-definite covariance matrix.

(a) Prove that the distribution of the criterion function $J_e(1)$ given in Eq. 87 is normal with mean $nd\sigma^2$. Express σ in terms of $\mathbf{\Sigma}$.

(b) Prove that the variance of this distribution is $2nd\sigma^4$.

(c) Consider a suboptimal partition of the Gaussian by a hyperplane through the sample mean. Show that for large n, the sum of squared error for this partition is approximately normal with mean $n(d - 2/\pi)\sigma^2$ and variance $2n(d - 8/\pi^2)\sigma^4$, where σ is given in part (a).

41. Derive Eqs. 90 and 91 used in rejecting the null hypothesis associated with determining the number of clusters for a data set. Be sure to state any conditions you need to impose.

Section 10.11

42. Consider a simple greedy algorithm for creating a spanning tree based on the Euclidean distance between points.

(a) Write pseudocode for creating a minimal spanning tree linking n points in d dimension.

(b) Let k denote the average linkage per node. What is the average space complexity of your algorithm?

(c) What is the average time complexity?

Section 10.12

43. Consider the adaptive resonance clustering algorithm.

(a) Show that the standard ART algorithm of Fig. 10.19 cannot learn the XOR problem.

(b) Explain how the number of clusters generated by the adaptive resonance algorithm depends upon the order of presentation of the samples.

(c) Discuss the benefits and drawbacks of adaptive resonance in stationary and in nonstationary environments.

Section 10.13

44. Show that minimizing a mean-squared error criterion for d-dimensional data leads to the k-dimensional representation ($k < d$) of the Karhunen-Loéve transform (Eq. 94) as follows. For simplicity, assume that the data set has zero mean. (If the mean is not zero, we can always subtract off the mean from each vector to define new vectors.)

(a) The (scalar) projection of a vector $\mathbf{x}$ onto a unit vector $\mathbf{e}$, $a(\mathbf{e}) = \mathbf{x}^t\mathbf{e}$, is, of course, a random variable. Define the variance of a to be $\sigma^2 = \mathcal{E}_{\mathbf{x}}[a^2]$. Show that $\sigma^2 = \mathbf{e}^t\boldsymbol{\Sigma}\mathbf{e}$, where $\boldsymbol{\Sigma} = \mathcal{E}_{\mathbf{x}}[\mathbf{x}\mathbf{x}^t]$ is the correlation matrix.

(b) A vector $\mathbf{e}$ that yields an extremal or stationary value of this variance must obey $\sigma^2(\mathbf{e} + \delta\mathbf{e}) = \sigma^2(\mathbf{e})$, where $\delta\mathbf{e}$ is a small perturbation. Show that this condition implies $(\delta\mathbf{e})^t\boldsymbol{\Sigma}\mathbf{e} = 0$ at such a stationary point.

(c) Consider small variations $\delta\mathbf{e}$ that do not change the length of the vector, that is, ones in which $\delta\mathbf{e}$ is perpendicular to $\mathbf{e}$. Use this condition and your above results to show that $(\delta\mathbf{e})^t\boldsymbol{\Sigma}\mathbf{e} - \lambda(\delta\mathbf{e})^t\mathbf{e} = 0$, where λ is a scalar. Show that the necessary and sufficient solution is $\boldsymbol{\Sigma}\mathbf{e} = \lambda\mathbf{e}$—as based on Eq. 94.

(d) Define a sum-squared-error criterion for a set of points in d-dimensional space and their projections onto a k-dimensional linear subspace ($k < d$). Use your results above to show that in order to minimize your criterion, the subspace shoud be spanned by the k largest eigenvectors of the correlation matrix.

45. Consider a linear neural net network consisting of $d - k - d$ units ($k < d$). Show that such a neural net, when trained to autoassociate patterns, performs principal component analysis by considering the minimization it solves.

46. Consider the use of neural networks for nonlinear principal component analysis.

(a) Prove that if all units in the five-layer network of Fig. 10.23 are linear, and the network trained to serve as an autoencoder, then the representation learned at the middle layer corresponds to the linear principal component of the data.

(b) State briefly why this also implies that a three-layer network (input, hidden, output) cannot be used for nonlinear principal component analysis, even if the middle layer consists of nonlinear units.

47. Use the fact that the sum of samples from two Gaussians is again a Gaussian to show why independent component analysis cannot isolate sources perfectly if two or more components are Gaussian.

48. Follow arguments similar to those leading to Eq. 102 to show that the learning rule for bias weights in independent component analysis using sigmoidal nonlinearity is the one given by Eq. 106.

49. Generalize the learning rules of Eqs. 102 and 106 to the case where the number of sensed signals k is strictly greater than the number of independent components d, as is illustrated in Fig. 10.25. Be sure to note that $\mathbf{W}$ is not square.

Section 10.14

50. Consider the use of multidimensional scaling for representing the points $\mathbf{x}_1 = (1, 0)^t$, $\mathbf{x}_2 = (0, 0)^t$, and $\mathbf{x}_3 = (0, 1)^t$ in one dimensions. To obtain a unique solution, assume that the image points satisfy $0 = y_1 < y_2 < y_3$.

(a) Show that the criterion function J_{ee} is minimized by the configuration with $y_2 = (1 + \sqrt{2})/3$ and $y_3 = 2y_2$.

(b) Show that the criterion function J_{ff} is minimized by the configuration with $y_2 = (2 + \sqrt{2})/4$ and $y_3 = 2y_2$.

COMPUTER EXERCISES

Several exercises make use of the data in the following table.

Sample	x_1	x_2	x_3	Sample	x_1	x_2	x_3
1	−7.82	−4.58	−3.97	11	6.18	2.81	5.82
2	−6.68	3.16	2.71	12	6.72	−0.93	−4.04
3	4.36	−2.19	2.09	13	−6.25	−0.26	0.56
4	6.72	0.88	2.80	14	−6.94	−1.22	1.13
5	−8.64	3.06	3.50	15	8.09	0.20	2.25
6	−6.87	0.57	−5.45	16	6.81	0.17	−4.15
7	4.47	−2.62	5.76	17	−5.19	4.24	4.04
8	6.73	−2.01	4.18	18	−6.38	−1.74	1.43
9	−7.71	2.34	−6.33	19	4.08	1.30	5.33
10	−6.91	−0.49	−5.68	20	6.27	0.93	−2.78

Section 10.4

1. Consider the univariate normal mixture

$$p(\mathbf{x}|\boldsymbol{\theta}) = \frac{P(\omega_1)}{\sqrt{2\pi}\sigma_1} \exp\left[-\frac{1}{2}\left(\frac{x-\mu_1}{\sigma_1}\right)^2\right] + \frac{1-P(\omega_1)}{\sqrt{2\pi}\sigma_2} \exp\left[-\frac{1}{2}\left(\frac{x-\mu_2}{\sigma_2}\right)^2\right].$$

Write a general program for computing the maximum-likelihood values of the parameters, and apply it to the 20 x_1 points in the table above under the following assumptions of what is known and what is unknown:

(a) Known: $P(\omega_1) = 0.5, \sigma_1 = \sigma_2 = 1$; Unknown: μ_1 and μ_2.

(b) Known: $P(\omega_1) = 0.5$; Unknown: $\sigma_1 = \sigma_2 = \sigma$, μ_1 and μ_2.

(c) Known: $P(\omega_1) = 0.5$; Unknown: σ_1, σ_2, μ_1 and μ_2.

(d) Unknown: $P(\omega_1)$, σ_1, σ_2, μ_1 and μ_2.

2. Write a program to implement k-means clustering (Algorithm 1), and apply it to the three-dimensional data in the table for the following assumed numbers of clusters, and starting points.

   (a) Let $c = 2$, $\mathbf{m}_1(0) = (1, 1, 1)^t$, and $\mathbf{m}_2(0) = (-1, 1, -1)^t$.

   (b) Let $c = 2$, $\mathbf{m}_1(0) = (0, 0, 0)^t$, and $\mathbf{m}_2(0) = (1, 1, -1)^t$. Compare your final solution with that from part (a), and explain any differences, including the number of iterations for convergence.

   (c) Let $c = 3$, $\mathbf{m}_1(0) = (0, 0, 0)^t$, $\mathbf{m}_2(0) = (1, 1, 1)^t$, and $\mathbf{m}_3(0) = (-1, 0, 2)^t$.

   (d) Let $c = 3$, $\mathbf{m}_1(0) = (-0.1, 0, 0.1)^t$, $\mathbf{m}_2(0) = (0, -0.1, 0.1)^t$, and $\mathbf{m}_3(0) = (-0.1, -0.1, .1)^t$. Compare your final solution with that from part (c), and explain any differences, including the number of iterations for convergence.

3. Repeat Computer exercise 2, but use instead a fuzzy k-means algorithm with the "blending" set by $b = 2$ (Eqs. 32 and 33).

4. Explore the problems that can come with misspecifying the number of clusters in the fuzzy k-means algorithm (Algorithm 2) using the following one-dimensional data: $\mathcal{D} = \{-5.0, -4.5, -4.1, -3.9, 2.5, 2.8, 3.1, 3.9, 4.5\}$.

   (a) Use your program in the four conditions defined by $c = 2$, $c = 3$, $b = 1$, and $b = 4$. In each case initialize the cluster centers to distinct values, but ones near $x = 0$.

   (b) Compare your solutions to the $c = 3$, $b = 4$ case to the $c = 3$, $b = 1$ case, and discuss any sources of the differences.

5. Show how a few labeled samples in a k-means algorithm can improve clustering of unlabeled samples in the following somewhat extreme case.

   (a) Generate 50 two-dimensional samples for each of four spherical Gaussians, $p(\mathbf{x}|\omega_i) \sim N(\boldsymbol{\mu}_i, \mathbf{I})$, where $\boldsymbol{\mu}_1 = \binom{-2}{-2}$, $\boldsymbol{\mu}_2 = \binom{-2}{2}$, $\boldsymbol{\mu}_3 = \binom{2}{2}$, and $\boldsymbol{\mu}_4 = \binom{2}{-2}$.

   (b) Choose $c = 4$ initial positions for the cluster means randomly from the full 200 samples. What is the probability that your random selection yields exactly one cluster center for each component density? (Make the simplifying assumption that the component densities do not overlap significantly.)

   (c) Using the four samples selected in part (b), run a k-means clusterer on the full 200 points. (If the four points in fact come from different components, reselect samples to ensure that at least two come from the same component density before using your clusterer.)

   (d) Now assume you have some label information, in particular four samples known to come from distinct component densities. Using these as your initial cluster centers, run a k-means clusterer on the full 200 points.

   (e) Discuss the value of a few labeled samples for clustering in light of the final clusters given in parts (c) and (d).

Section 10.5

6. Explore unsupervised Bayesian learning of the mean of a Gaussian distribution in the following way.

(a) Generate a data set $\mathcal{D}$ of 30 points, uniformly distributed in the interval $-10 \leq x \leq +10$.

(b) Assume (incorrectly) that the data in $\mathcal{D}$ were drawn from a normal distribution with known variance but unknown mean, namely, $p(x) \sim N(\mu, 2)$. Thus the unknown parameter $\boldsymbol{\theta}$ in Eq. 42 is simply the scalar μ. Assume a wide prior for the parameter: $p(\mu)$ is uniform in the range $-10 \leq \mu \leq +10$. Plot posterior probabilities for $k = 0, 1, 2, 3, 4, 5, 10, 15, 20, 25$, and 30 points from $\mathcal{D}$.

(c) Now assume instead a narrow prior—that is, $p(\mu)$ uniform in the range $-1 \leq \mu \leq +1$—and repeat part (b) using the same order of data presentation.

(d) Are your curves for part (b) and part (c) the same for small number of points? For large number of points? Explain.

7. Write a decision-directed clusterer related to k-means in the following way.

(a) First, generate a set $\mathcal{D}$ of $n = 1000$ three-dimensional points in the unit cube, $0 \leq x_i \leq 1, i = 1, 2, 3$.

(b) Randomly choose $c = 4$ of these points as the initial cluster centers $\mathbf{m}_j$, $j = 1, 2, 3, 4$.

(c) The core of the algorithm operates as follows: First, each sample $\mathbf{x}_i$ is classified by the nearest cluster center $\mathbf{m}_j$. Next, each mean $\mathbf{m}_j$ is calculated to be the mean of the samples in ω_j. If there is no change in the centers after n presentations, halt.

(d) Use your algorithm to plot four trajectories of the position of the cluster centers.

(e) What is the space and the time complexities of this algorithm? State any assumptions you invoke.

Section 10.6

8. Explore the role of metrics, similarity measures, and thresholds on cluster formation in the following way.

(a) First, generate a two-dimensional data set consisting of two parts: $\mathcal{D}_1$ contains 1000 points whose distance from the origin is chosen uniformly in the range $3 \leq r \leq 5$, and angular position uniform in the range $0 \leq \phi < 2\pi$; likewise, $\mathcal{D}_2$ consists of 50 points of distance $0 \leq r \leq 2$ and angle $0 \leq \phi < 2\pi$. The full data set used below is $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$.

(b) Write a simple clustering algorithm that links any two points $\mathbf{x}$ and $\mathbf{x}'$ if $d(\mathbf{x}, \mathbf{x}') < \theta$, where θ is a threshold selected by the user, and distance is calculated by means of a general Minkowski metric (Eq. 49),

$$d(\mathbf{x}, \mathbf{x}') = \left(\sum_{k=1}^{d} |x_k - x'_k|^q \right)^{1/q}.$$

Let $q = 2$ (Euclidean distance) and apply your algorithm to the data $\mathcal{D}$ for the following thresholds: $\theta = 0.01, 0.05, 0.1, 0.5, 1, 5$. In each case, plot all 150 points and differentiate the clusters by linking their points or other plotting convention.

(c) Repeat part (b) with $q = 1$ (city block distance).

(d) Repeat part (b) with $q = 4$.

(e) Discuss how the metric affects the "natural" number of clusters implied by your results.

Section 10.7

9. Explore different clustering criteria by exhaustive search in the following way. Let $\mathcal{D}$ be the first seven three-dimensional points in the table above.

(a) If we assume that any cluster must have at least one point, how many cluster configurations are possible for the seven points?

(b) Write a program to search through each of the cluster configurations, and for each compute the following criteria: J_e (Eq. 54), J_d (Eq. 68), $\sum_{i=1}^{d} \lambda_i$ (Eq. 69), $J_f = \text{tr}[\mathbf{S}_T^{-1}\mathbf{S}_W]$ (Eq. 70) and $|\mathbf{S}|/|\mathbf{S}_T|$ (Eq. 71). Show the optimal clusters for each of your four criteria.

(c) Perform a whitening transformation on your points and repeat part (b).

(d) In light of your results, discuss which of the criteria are invariant to the whitening transformation.

Section 10.8

10. Show that the Basic Iterative Least-Squares clustering algorithm gives solutions and final criterion values that depend upon starting conditions in the following way. Implement Algorithm 3 for $c = 3$ clusters and apply it to the data in the table above. For each simulation, list the final clusters as sets of points (identified by their index in the table), along with the corresponding value of the criterion function.

(a) $\mathbf{m}_1(0) = (1, 1, 1)^t$, $\mathbf{m}_2(0) = (-1, -1, -1)^t$, and $\mathbf{m}_3(0) = (0, 0, 0)^t$.

(b) $\mathbf{m}_1(0) = (0.1, 0.1, 0.1)^t$, $\mathbf{m}_2(0) = (-0.1, -0.1, -0.1)^t$, and $\mathbf{m}_3(0) = (0, 0, 0)^t$.

(c) $\mathbf{m}_1(0) = (2, 0, 2)^t$, $\mathbf{m}_2(0) = (-2, 0, -2)^t$, and $\mathbf{m}_3(0) = (1, 1, 1)^t$.

(d) $\mathbf{m}_1(0) = (0.5, 1, 0.2)^t$, $\mathbf{m}_2(0) = (0.2, -1, 0.5)^t$, and $\mathbf{m}_3(0) = (0.2, 0.4, 0.6)^t$.

(e) Explain why your final answers differ.

Section 10.9

11. Implement the basic hierarchical agglomerative clustering algorithm (Algorithm 4), as well as a method for drawing dendrograms based on its results. Apply your algorithm and draw dendrograms to the data in the table above using the distance measures indicated below. Define the similarity between two clusters to be linear in distance, with similarity = 100 for singleton clusters ($c = 20$) and similarity = 0 for the single cluster ($c = 1$).

(a) d_{min} (Eq. 79)

(b) d_{max} (Eq. 80)

(c) d_{avg} (Eq. 81)

(d) d_{mean} (Eq. 82)

12. Explore the use of cluster dendrograms for selecting the "most natural" number of clusters.

(a) Write a program to perform hierarchical clustering and display a dendrogram, using measure of distance to be selected from Eqs. 79–82.

(b) Write a program to generate n/c points from each of c one-dimensional Gaussians, $p(x|\omega_i) \sim N(\mu_i, \sigma_i^2)$, $i = 1, \ldots, c$. Use your program to generate $n = 50$ points, 25 in each of two clusters, with $\mu_1 = 0$, $\mu_2 = 1$, and $\sigma_1^2 = \sigma_2^2 = 1$. Repeat with $\mu_2 = 4$.

(c) Use your program from (a) to generate dendrograms for each of the two data sets generated in (b).

(d) The difference in similarity values for successive levels is a random variable, which we can model as a normal distribution with some mean and variance. Suppose we define the "most natural" number of clusters according to the largest gap in similarity values, and that this largest gap is *significant* if it differs "significantly" from the distribution. State your criterion analytically, and show that one of the cases in (b) indeed has two clusters.

Section 10.11

13. Implement a basic Competitive Learning clustering algorithm (Algorithm 6) and apply it to the three-dimensional data in the table above as follows.

(a) First, preprocess the data by augmenting each vector with $x_0 = 1$ and normalizing to unit length. In this way, each point lies on the surface of a hypersphere.

(b) Set $c = 2$, and let the initial (normalized) weght vectors correspond to patterns 1 and 2. Let the learning rate be $\eta = 0.1$. Present the patterns in cyclic order, $1, 2, \ldots, 20, 1, 2, \ldots, 20, 1, 2, \ldots$.

(c) Modify your program so as to reduce the learning rate by multiplying by the constant factor $\alpha < 1$ after each pattern presentation, so the learning rate approaches zero exponentially. Repeat your simulation of part (b) with such decay, where $\alpha = 0.99$. Compare your final clusterings with those from using $\alpha = 0.5$.

(d) Repeat part (c) but with the patterns chosen in a random order—that is, with the probability of presenting any given pattern being 1/20 per trial. Discuss the role of random versus sequenced pattern presentation on the final clusterings.

Section 10.12

14. Consider the application of graph-theoretic clustering methods applied to the 20 points in the table above.

(a) Write a program to form the similarity matrix $\mathbf{S} = [s_{ij}]$, of Eq. 93 where two points are similar if their Euclidean distance is less than a threshold d_0.

(b) Apply your program to the x_1x_2-components of the data in the table. Specifically, for $d_0 = 0.01, 0.05, 0.1, 0.5, 1.0$ show the number of distinct clusters, including singleton clusters.

(c) Repeat part (b) for the full three-dimensional data in the table above.

Section 10.13

15. Use principal component analysis to represent all the three-dimensional data in the table above in two dimensions. What are the eigenvectors and eigenvalues?

16. Explore the use of independent component analysis for blind source separation in the following example.

(a) Generate 100 points for $t = 1, \ldots, 100$ for $x_1(t) = \cos(t)$, and $x_2(t) = e^{-t} - 5e^{-t/5}$ and two sensor signals:

$$s_1(t) = 0.5x_1(t) + 0.2x_2(t)$$
$$s_2(t) = 0.1x_1(t) + 0.4x_2(t).$$

(b) Write a program for independent component analysis, based on Eqs. 105 and 48 to find $\mathbf{W}$ and $\mathbf{w}_0$, as defined in the text. Let your starting matrix be $\mathbf{W} = \begin{pmatrix} 0.1 & 0.3 \\ 1.0 & 0.2 \end{pmatrix}$ and bias vector be $\mathbf{w}_0 = \begin{pmatrix} 0.01 \\ -0.02 \end{pmatrix}$.

Section 10.14

17. Write a program to implement multidimensional scaling.

(a) Use your program to represent the three-dimensional data in the table above into a two-dimensional space minimizing the J_{ee} criterion of Eq. 107. Use a numeral, 1–20, to label each of your points.

(b) Repeat with J_{ff} of Eq. 108.

(c) Repeat with J_{ef} of Eq. 109.

(d) In light of your above three plots, discuss the relationships among your three different clustering criteria.

BIBLIOGRAPHY

[1] Phipps Arabie, Lawrence J. Hubert, and Geert De Soete, editors. *Clustering and Classification*. World Scientific, River Edge, NJ, 1998.

[2] Thomas A. Bailey and Richard C. Dubes. Cluster validity profiles. *Pattern Recognition*, 15:61–83, 1982.

[3] Geoffrey H. Ball and David J. Hall. Some implications of interactive graphic computer systems for data analysis and statistics. *Technometrics*, 12:17–31, 1970.

[4] Anthony J. Bell and Terrence J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1996.

[5] James C. Bezdek. *Fuzzy Mathematics in Pattern Classification*. Ph.D. thesis, Cornell University, Applied Mathematics Center, Ithaca, NY, 1973.

[6] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981.

[7] Christopher M. Bishop. Bayesian PCA. In Michael S. Kearns, Sara A. Solla, and David A. Cohn, editors, *Advances in Neural Information Processing Systems 3*, pages 382–387. MIT Press, Cambridge, MA, 1999.

[8] Ingwer Borg and Patrick J. F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer-Verlag, New York, 1997.

[9] Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294, 1988.

[10] Gail A. Carpenter and Stephen Grossberg, editors. *Pattern Recognition by Self-Organizing Neural Networks*. MIT Press, Cambridge, MA, 1991.

[11] Michael A. Cohen, Stephen Grossberg, and David G. Stork. Speech perception and production by a self-organizing neural network. In Gail A. Carpenter and Stephen Grossberg, editors, *Pattern Recognition by Self-Organizing Neural Networks*, pages 615–633. MIT Press, Cambridge, MA, 1991.

[12] William H. E. Day and Herbert Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1(1):7–24, 1984.

[13] Konstantinos I. Diamantaras and Sun-Yuang Kung. *Principal Component Neural Networks: Theory and Applications*. Wiley-Interscience, New York, 1996.

[14] Thomas Eckes. An error variance approach to 2-mode hierarchical-clustering. *Journal of Classification*, 10(1):51–74, 1993.

[15] Ze'ev Roth and Yoram Baram. Multidmensional density shaping by sigmoids. *IEEE Transactions on Neural Networks*, TNN-7(5):1291–1298, 1996.

[16] Michel Gaeta and Jean-Louis Lacoume. Sources separation without *a priori* knowledge: The maximum likelihood solution. In Luis Torres, Enrique Masgrau, and Miguel A. Lagunas, editors, *European Association for Signal Processing, Eusipco 90*, pages 621–624, Barcelona, Spain, 1990. Elsevier.

[17] Selvanayagam Ganesalingam. Classification and mixture approaches to clustering via maximum likelihood. *Applied Statistics*, 38(3):455–466, 1989.

[18] Allen Gersho and Robert M. Gray. *Vector Quantization and Signal Processing*. Kluwer Academic Publishers, Boston, MA, 1992.

[19] John C. Gower and Gavin J. S. Ross. Minimum spanning trees and single-linkage cluster analysis. *Applied Statistics*, 18:54–64, 1969.

[20] Victor Hasselblad. Estimation of parameters for a mixture of normal distributions. *Technometrics*, 8:431–444, 1966.

[21] Geoffrey Hinton and Terrence J. Sejnowski, editors. *Unsupervised Learning: Foundations of Neural Computation*. MIT Press, Cambridge, MA, 1999.

[22] Lawrence J. Hubert. Min and max hierarchical clustering using asymmetric similarity measures. *Psychometrika*, 38(1):63–72, 1973.

[23] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs, NJ, 1988.

[24] Ian T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[25] Christian Jutten and Jeanny Herault. Blind separation of sources 1: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24(1):1–10, 1991.

[26] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York, 1990.

[27] Ian J. Kitching, Peter L. Forey, Christopher J. Humphries, and David M. Williams. *Cladistics: A Practical Course in Systematics*. Clarendon Press, Oxford, UK, second edition, 1998.

[28] Teuvo Kohonen. Self-organizing formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

[29] Tuevo Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, third edition, 1989.

[30] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991.

[31] Joseph B. Kruskal. The relationship between multidimensional scaling and clustering. In John Van Ryzin, editor, *Classification and Clustering: Proceedings of an Advanced Seminar Conducted by the Mathematics Research Center, the University of Wisconsin–Madison, May 3-5, 1976*, pages 7–44. Academic Press, New York, 1977.

[32] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, IT-2:129–137, 1982.

[33] Jianchang Mao and Anil K. Jain. A self-organizing network for hyperellipsoidal clustering (HEC). *IEEE Transactions on Neural Networks*, TNN-7(1):16–29, 1996.

[34] Geoffrey J. McLachlan and Kaye E. Basford. *Mixture Models*. Dekker, New York, 1988.

[35] Barbara Moore. ART1 and pattern clustering. In David Touretzky, Geoffrey Hinton, and Terrence Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 174–185. Morgan Kaufmann, San Mateo, CA, 1988.

[36] Dragan Obradovic and Gustavo Deco. Information maximization and independent component analysis: Is there a difference? *Neural Computation*, 10(8):2085–2101, 1998.

[37] Erkki Oja. *Subspace Methods of Pattern Recognition*. John Wiley, New York, 1983.

[38] Erkki Oja and Samuel Kaski, editors. *Kohonen Maps*. Elsevier, Amsterdam, The Netherlands, 1999.

[39] Barak A. Pearlmutter and Lucas C. Parra. A context-sensitive generalization of ICA. In *International Conference on Neural Information Processing*, pages 151–157. Springer-Verlag, Hong Kong, 1996.

[40] Barak Perlmutter and Lucas C. Parra. Maximum-likelihood blind source separation: A context-sensitive generalization of ICA. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 613–619. MIT Press, Cambridge, MA, 1997.

[41] James O. Ramsay. Maximum-likelihood estimation in multidimensional scaling. *Psychometrika*, 42(2):241–266, 1977.

[42] Helge Ritter, Thomas Martinetz, and Klaus Schulten. *Neural Computation and Self-Organizing Maps: An Introduction*. Addison-Wesley, New York, English translation from the German edition, 1992.

[43] Susan S. Schiffman, Mark Lance Reynolds, and Forrest W. Young. *Introduction to Multidimensional Scaling: Theory, Methods and Applications*. Academic Press, New York, 1981.

[44] Stephen P. Smith and Anil K. Jain. Testing for uniformity in multidimensional data. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, PAMI-6(1):73–81, 1984.

[45] David G. Stork. Self-organization, pattern recognition and adaptive resonance networks. *Journal of Neural Network Computing*, 1(1):26–42, 1989.

[46] Henry Teicher. Identifiability of mixtures. *Annals of Mathematical Statistics*, 32(1):244–248, 1961.

[47] John H. Wolfe. Pattern clustering by multivariate mixture analysis. *Multivariate Behavioral Research*, 5:329–350, 1970.

[48] Te Won Lee. *Independent Component Analysis: Theory and Applications*. Kluwer Academic Publishers, Dordrecht, 1998.

[49] Hujun Yin and Nigel M. Allinson. On the distribution and convergence of feature space in self-organizing maps. *Neural Computation*, 7(6):1178–1187, 1998.