Pattern Classification and Scene Analysis (2nd ed.)
Part 1: Pattern Classification

Richard O. Duda, Peter E. Hart and David G. Stork

February 27, 1995

Contact: Dr. David G. Stork
Ricoh California Research Center
2882 Sand Hill Road, Suite 115
Menlo Park, CA 94025-7022 USA
stork@crc.ricoh.com

Contents

Chapter 11

Unsupervised Learning and Clustering

11.1 Introduction

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

One might wonder why anyone is interested in such an unpromising problem, and whether or not it is even possible in principle to learn anything of value from unlabelled samples. There are at least five basic reasons for interest in unsupervised procedures. First, the collection and labelling of a large set of sample patterns can be surprisingly costly. For instance, recorded speech is virtually free, but *labelling* 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, labelled set of samples, and then "tuned up" by allowing it to run without supervision on a large, unlabelled 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) unlabelled 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 represent a sort of data-dependent "smart preprocessing" or "smart feature extraction." Lastly, in the early stages of an investigation it may be valuable to gain some insight into the nature or structure of the data. The discovery of distinct subclasses or similarities among 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 unlabelled data depends upon the assumptions one is willing to accept — theorems can not be proved without premises. We shall begin with the very restrictive assumption that the functional forms for the underlying probability densitites 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 **??**. 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, there are still among the more useful tools for pattern recognition problems.

11.2 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 a priori probabilities $P(\omega_j)$ for each class are known, $j = 1, ..., c$.
3. The forms for the class-conditional probability densities $p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)$ are known, $j = 1, ..., c$.
4. All that is unknown are the values for the c parameter vectors $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_c$.

Samples are assumed to be 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

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

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_c)$. 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 a priori 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.

COMPONENT DENSITIES

MIXING PARAMETERS

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, use a Bayesian or maximum likelihood 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 nonparameteric methods of Chapter **??** 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.

COMPLETELY UNIDENTI-FIABLE

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

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

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 32).

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. Thus, while it can be shown that mixtures of normal densities are usually identifiable, the parameters in the simple mixture density

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

can not be uniquely identified if $P(\omega_1) = P(\omega_2)$, for 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.

[*] Technically speaking, a distribution is not identifiable if we cannot determine the parameters *without bias.* We might guess their correct value but such a guess would have to be biased in some way.

11.3 Maximum Likelihood Estimates

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

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

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

$$p(\mathcal{H}|\boldsymbol{\theta}) \equiv \prod_{k=1}^{n} p(\mathbf{x}_k|\boldsymbol{\theta}). \tag{2}$$

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

If we assume that $p(\mathcal{H}|\boldsymbol{\theta})$ 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

$$l = \sum_{k=1}^{n} \log p(\mathbf{x}_k|\boldsymbol{\theta}) \tag{3}$$

and

$$\nabla_{\boldsymbol{\theta}_i} l = \sum_{k=1}^{n} \frac{1}{p(\mathbf{x}_k|\boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}_i} \left[\sum_{j=1}^{c} p(\mathbf{x}_k|\omega_j, \boldsymbol{\theta}_j)P(\omega_j) \right]. \tag{4}$$

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 a posteriori probability

$$P(\omega_i|\mathbf{x}_k, \boldsymbol{\theta}) = \frac{p(\mathbf{x}_k|\omega_i, \boldsymbol{\theta}_i)P(\omega_i)}{p(\mathbf{x}_k|\boldsymbol{\theta})}, \tag{5}$$

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

$$\nabla_{\boldsymbol{\theta}_i} l = \sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}_i} \log p(\mathbf{x}_k|\omega_i, \boldsymbol{\theta}_i). \tag{6}$$

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

$$\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) \nabla_{\boldsymbol{\theta}_i} \log p(\mathbf{x}_k|\omega_i, \hat{\boldsymbol{\theta}}_i) = 0, \quad i = 1, ..., c. \tag{7}$$

Conversely, 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 a priori probabilities $P(\omega_i)$ among the unknown quantities. In this case the search for the maximum value of $p(\mathcal{H}|\boldsymbol{\theta})$ extends over $\boldsymbol{\theta}$ and $P(\omega_i)$, subject to the constraints

$$P(\omega_i) \geq 0 \quad i = 1, ..., c$$

and

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

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 **??**) 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

$$\hat{P}(\omega_i) = \frac{1}{n} \sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) \tag{8}$$

and

$$\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) \nabla_{\boldsymbol{\theta}_i} \log p(\mathbf{x}_k|\omega_i, \hat{\boldsymbol{\theta}}_i) = 0, \tag{9}$$

where

$$\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) = \frac{p(\mathbf{x}_k|\omega_i, \hat{\boldsymbol{\theta}}_i)\hat{P}(\omega_i)}{\sum\limits_{j=1}^{c} p(\mathbf{x}_k|\omega_i, \hat{\boldsymbol{\theta}}_i)\hat{P}(\omega_i)}. \tag{10}$$

These equations have the following interpretation. Equation 8 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. Note especially that each sample is weighted equally. Equation 10 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. 9 is a bit subtle, we can understand it clearly in the trivial $k = 1$ case. Since $\hat{P} \neq 0$, 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.

11.4 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 ($\times$) and which are unknown (?):

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

Case 1 is the simplest, and 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 Sect. **??**.

11.4.1 Case 1: Unknown Mean Vectors

If the only unknown quantities are the mean vectors $\boldsymbol{\mu}_i$, then $\boldsymbol{\theta}_i$ can be identified with $\boldsymbol{\mu}_i$ and Eq. 7 can be used to obtain necessary conditions on the maximum likelihood estimate for $\boldsymbol{\mu}_i$. Since

$$\log p(\mathbf{x}|\omega_i, \boldsymbol{\mu}_i) = \Leftrightarrow \log[(2\pi)^{d/2}|\boldsymbol{\Sigma}_i|^{1/2}] \Leftrightarrow \frac{1}{2}(\mathbf{x} \Leftrightarrow \boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} \Leftrightarrow \boldsymbol{\mu}_i), \qquad (11)$$

and

$$\nabla_{\boldsymbol{\mu}_i} \log p(\mathbf{x}|\omega_i, \boldsymbol{\mu}_i) = \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} \Leftrightarrow \boldsymbol{\mu}_i). \qquad (12)$$

Thus, Eq. 7 for the maximum-likelihood estimate $\hat{\boldsymbol{\mu}}_i$ yields

$$\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}}) \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}_k \Leftrightarrow \hat{\boldsymbol{\mu}}_i) = 0, \quad \text{where } \hat{\boldsymbol{\mu}} = (\hat{\boldsymbol{\mu}}_1, ..., \hat{\boldsymbol{\mu}}_c). \qquad (13)$$

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

$$\hat{\boldsymbol{\mu}}_i = \frac{\sum\limits_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}})\mathbf{x}_k}{\sum\limits_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}})}. \qquad (14)$$

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 a posteriori 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. 14 essentially gives $\hat{\boldsymbol{\mu}}_i$ as the average of the samples coming from the ith class.

Unfortunately, Eq. 14 does not give $\hat{\boldsymbol{\mu}}_i$ explicitly, and if we substitute

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

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. 14 suggests the following iterative scheme for improving the estimates:

$$\hat{\boldsymbol{\mu}}_i(j+1) = \frac{\sum\limits_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}}(j))\mathbf{x}_k}{\sum\limits_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}}(j))} \qquad (15)$$

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 (Computer Exercise 4).

Example 1: Mixtures of two 1D Gaussians

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

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

The 25 samples shown in the Table were drawn from this mixture with $\mu_1 = \Leftrightarrow 2$ and $\mu_2 = 2$. Let us use these samples to compute the log-likelihood function

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

for various values of μ_1 and μ_2. Figure 11.1 is a plot showing how l varies with μ_1 and μ_2. The maximum value of l occurs at $\hat{\mu}_1 = \Leftrightarrow 2.130$ and $\hat{\mu}_2 = 1.668$, which is in the rough vicinity of the true values $\mu_1 = \Leftrightarrow 2$ and $\mu_2 = 2$. However, l reaches another peak of comparable height at $\hat{\mu}_1 = 2.085$ and $\hat{\mu}_2 = \Leftrightarrow 1.257$. Roughly speaking, this solution corresponds to interchanging μ_1 and μ_2. Note that had the a priori 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	Class	
		ω_1	ω_2
1	0.608		$\times$
2	-1.590	$\times$	
3	0.235		$\times$
4	3.949		$\times$
5	-2.249	$\times$	
6	2.704		$\times$
7	-2.473	$\times$	
8	0.672		$\times$
9	0.262		$\times$
10	1.072		$\times$
11	-1.773	$\times$	
12	0.537		$\times$
13	3.240		$\times$
14	2.400		$\times$
15	-2.499	$\times$	
16	2.608		$\times$
17	-3.458	$\times$	
18	0.257		$\times$
19	2.569		$\times$
20	1.415		$\times$
21	1.410		$\times$
22	-2.653	$\times$	
23	1.396		$\times$
24	3.286		$\times$
25	-0.712	$\times$	

Additional insight into the nature of these multiple solutions can be obtained by examining the resulting estimates for the mixture density. Figure 11.2 shows the true mixture density and the estimates obtained by using the maximum likelihood

Figure 11.1: Log-likelihood of a mixture model consisting of two univariate Gaussians as a function of their means, for the data in the Table.

Figure 11.2: The mixture density used to generate sample data, and two maximum likelihood estimates based on the data in the Table. The data themselves are shown below.

estimates as if they were the true parameter values. The 25 sample values are shown as a scatter of points along the abscissa. Note that the peaks of both the true mixture density and the maximum likelihood solution 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. 15 is used to determine solutions to Eq. 14 iteratively, the results depend on the starting values $\hat{\mu}_1(0)$ and $\hat{\mu}_2(0)$. Figure 11.3 shows how different starting points lead to different solutions, and gives some indication of rates of convergence. Note that 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 $P(\omega_i|x_k, \hat{\mu}_i(0), \hat{\mu}_i(0)) = P(\omega_i)$. Thus, Eq. 15 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.

Figure 11.3: Trajectories for the iterative maximum likelihood estimation of the means of a two-Gaussian mixture model based on the data in the Table.

11.4.2 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:

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

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

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

Clearly, for the rest of the samples

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

so that

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

Thus, 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. 8 – 10 to obtain estimates for $\boldsymbol{\mu}_i$, $\mathbf{\Sigma}_i$, and $P(\omega_i)$. When we include the elements of $\mathbf{\Sigma}_i$ 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 $\mathbf{\Sigma}_i^{-1}$ rather than $\mathbf{\Sigma}_i$ be the unknown parameters. With these observations, the actual differentiation of

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

with respect to the elements of $\boldsymbol{\mu}_i$ and $\mathbf{\Sigma}_i^{-1}$ 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 $\mathbf{\Sigma}_i$, and $\sigma^{pq}(i)$ be the pqth element of $\mathbf{\Sigma}_i^{-1}$. Then

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

and

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

where δ_{pq} is the Kronecker delta. Substituting these results in Eq. 9 and doing a small amount of algebraic manipulation (Problem 27), we obtain the following equations for the local-maximum-likelihood estimate $\hat{\boldsymbol{\mu}}_i$, $\hat{\mathbf{\Sigma}}_i$, and $\hat{P}(\omega_i)$:

$$\hat{P}(\omega_i) = \frac{1}{n}\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) \tag{16}$$

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

$$\hat{\mathbf{\Sigma}}_i = \frac{\sum\limits_{k=1}^{n} \hat{p}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})(\mathbf{x}_k \Leftrightarrow \boldsymbol{\mu}_i)(\mathbf{x}_k \Leftrightarrow \boldsymbol{\mu}_i)^t}{\sum\limits_{k=1}^{n} \hat{p}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})} \tag{18}$$

where

$$\begin{aligned} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) &= \frac{p(\mathbf{x}_k|\omega_i, \hat{\boldsymbol{\theta}}_i)\hat{P}(\omega_i)}{\sum\limits_{j=1}^{c} p(\mathbf{x}_k|\omega_j, \hat{\boldsymbol{\theta}}_j)\hat{P}(\omega_j)} \\ &= \frac{|\hat{\mathbf{\Sigma}}_i|^{-1/2} \exp\left[\Leftrightarrow \frac{1}{2}(\mathbf{x}_k \Leftrightarrow \hat{\boldsymbol{\mu}}_i)^t \hat{\mathbf{\Sigma}}_i^{-1}(\mathbf{x}_k \Leftrightarrow \hat{\boldsymbol{\mu}}_i)\right]\hat{P}(\omega_i)}{\sum\limits_{j=1}^{c} |\hat{\mathbf{\Sigma}}_i|^{-1/2} \exp\left[\Leftrightarrow \frac{1}{2}(\mathbf{x}_k \Leftrightarrow \hat{\boldsymbol{\mu}}_i)^t \hat{\mathbf{\Sigma}}_i^{-1}(\mathbf{x}_k \Leftrightarrow \hat{\boldsymbol{\mu}}_i)\right]\hat{P}(\omega_i)}. \end{aligned} \tag{19}$$

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 Sect. **??**, 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. 19 for $\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})$ and then to use Eqs. 16 – 18 to update these estimates. If the initial estimates are very good, having perhaps been obtained from a fairly large set of labelled 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 27).

11.4.3 K-means clustering

Of the various techniques that can be used to simplify the computation and accelerate convergence, we shall briefly consider one elementary, approximate method. From Eq. 19, it is clear that the probability $\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})$ is large when the squared Mahalanobis distance $(\mathbf{x}_k \Leftrightarrow \hat{\boldsymbol{\mu}}_i)^t \hat{\boldsymbol{\Sigma}}_i^{-1} (\mathbf{x}_k \Leftrightarrow \hat{\boldsymbol{\mu}}_i)$ is small. Suppose that we merely compute the squared Euclidean distance $\|\mathbf{x}_k \Leftrightarrow \hat{\boldsymbol{\mu}}_i\|^2$, 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

$$\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) \approx \begin{cases} 1 & \text{if } i = m \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

Then the iterative application of Eq. 17 leads to the following procedure for finding $\hat{\boldsymbol{\mu}}_1, ..., \hat{\boldsymbol{\mu}}_c$:

```
                    K-means clustering

Initialize $c$ means: $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_c$          initialize
Do classify $n$ samples by closest mean     classify samples
  recompute means from category samples    recompute mean
Until no change in means                   stopping criterion
End
```

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, since the means tend to move so as to minimize a squared-error criterion function. At

[*] This algorithm is historically called K-means, where K (our c) is the assumed number of clusters.

Figure 11.4: Trajectories for the K-means clustering procedure. The final Voronoi tesselation (for classification) is also shown — the means correspond to the "centers" of the Voronoi cells.

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 before. Figure 11.4 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 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 $\hat{\mu}_1 = \Leftrightarrow 2.176, \hat{\mu}_2 = 1.684$ (or to its image). This is close to the solution found by the maximum likelihood method (viz., $\hat{\mu}_1 = \Leftrightarrow 2.130$ and $\hat{\mu}_2 = 1.688$), and the trajectories show a general resemblance to those shown in Fig. 11.3. 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.

11.5 Unsupervised Bayesian Learning

11.5.1 The Bayes Classifier

Maximum likelihood methods do not consider the parameter vector $\boldsymbol{\theta}$ to be random — it is just unknown. Prior knowledge about the likely values for $\boldsymbol{\theta}$ is irrelevant, 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 a priori distribution $p(\boldsymbol{\theta})$, and we shall use the samples to compute the a posteriori density $p(\boldsymbol{\theta}|\mathcal{H})$. Interestingly enough, the analysis will closely parallel the analysis of supervised Bayesian learning (Sect. **??**.**??**), showing that the two problems are formally very similar.

We begin with an explicit statement of our basic assumptions. We assume that

1. The number of classes c is known.
2. The a priori probabilities $P(\omega_j)$ for each class are known, $j = 1, ..., c$.

3. The forms for the class-conditional probability densities $p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)$ are known, $j = 1, ..., c$, but the full parameter vector $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_c)$ is not known.

4. Part of our knowledge about $\boldsymbol{\theta}$ is contained in a known a priori density $p(\boldsymbol{\theta})$.

5. The rest of our knowledge about $\boldsymbol{\theta}$ is contained in a set $\mathcal{H}$ of n samples $\mathbf{x}_1, ..., \mathbf{x}_n$ drawn independently from the familiar mixture density

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

At this point we could go directly to the calculation of $p(\boldsymbol{\theta}|\mathcal{H})$. 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 a posteriori probability $P(\omega_i|\mathbf{x})$. We exhibit the role of the samples explicitly by writing this as $P(\omega_i|\mathbf{x}, \mathcal{H})$. By Bayes formula, we have

$$P(\omega_i|\mathbf{x}, \mathcal{H}) = \frac{p(\mathbf{x}|\omega_i, \mathcal{H})P(\omega_i|\mathcal{H})}{\sum\limits_{j=1}^{c} p(\mathbf{x}|\omega_i, \mathcal{H})P(\omega_i|\mathcal{H})}. \tag{21}$$

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

$$P(\omega_i|\mathbf{x}, \mathcal{H}) = \frac{p(\mathbf{x}|\omega_i, \mathcal{H})P(\omega_i)}{\sum\limits_{j=1}^{c} p(\mathbf{x}|\omega_i, \mathcal{H})P(\omega_i)}. \tag{22}$$

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

$$\begin{aligned} p(\mathbf{x}|\omega_i, \mathcal{H}) &= \int p(\mathbf{x}, \boldsymbol{\theta}|\omega_i, \mathcal{H})\; d\boldsymbol{\theta} \\ &= \int p(\mathbf{x}|\boldsymbol{\theta}, \omega_i, \mathcal{H})p(\boldsymbol{\theta}|\omega_i, \mathcal{H})\; d\boldsymbol{\theta}. \end{aligned} \tag{23}$$

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

$$P(\mathbf{x}|\omega_i, \mathcal{H}) = \int p(\mathbf{x}|\omega_i, \boldsymbol{\theta}_i)p(\boldsymbol{\theta}|\mathcal{H})\; d\boldsymbol{\theta}. \tag{24}$$

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 $p(\boldsymbol{\theta}|\mathcal{H})$, and thus our attention turns at last to that density.

11.5.2 Learning the Parameter Vector

Using Bayes formula, we can write

$$\boxed{p(\boldsymbol{\theta}|\mathcal{H}) = \frac{p(\mathcal{H}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\mathcal{H}|\boldsymbol{\theta})p(\boldsymbol{\theta})\ d\boldsymbol{\theta}}} \tag{25}$$

where the independence of the samples yields

$$p(\mathcal{H}|\boldsymbol{\theta}) = \prod_{k=1}^{n} p(\mathbf{x}_k|\boldsymbol{\theta}). \tag{26}$$

Alternatively, letting $\mathcal{H}^n$ denote the set of n samples, we can write Eq. 25 in the recursive form

$$p(\boldsymbol{\theta}|\mathcal{H}^n) = \frac{p(\mathbf{x}_n|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{H}^{n-1})}{\int p(\mathbf{x}_n|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{H}^{n-1})\ d\boldsymbol{\theta}}. \tag{27}$$

These are the basic equations for unsupervised Bayesian learning. Equation 25 emphasizes the relation between the Bayesian and the maximum likelihood solutions. If $p(\boldsymbol{\theta})$ is essentially uniform over the region where $p(\mathcal{H}|\boldsymbol{\theta})$ peaks, then $p(\boldsymbol{\theta}|\mathcal{H})$ 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. 22 & 24 yield

$$p(\mathbf{x}|\omega_i, \mathcal{H}) \approx p(\mathbf{x}|\omega_i, \hat{\boldsymbol{\theta}}) \tag{28}$$

and

$$P(\omega_i|\mathbf{x}, \mathcal{H}) \approx \frac{p(\mathbf{x}|\omega_i, \hat{\boldsymbol{\theta}}_i)P(\omega_i)}{\sum\limits_{j=1}^{c} p(\mathbf{x}|\omega_j, \hat{\boldsymbol{\theta}}_j)P(\omega_j)}. \tag{29}$$

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

As we saw in Sect. **??**.**??**, in the limit of large amounts of data, maximum likelihood and the Bayes methods will agree (or nearly agree). In fact, even in many *small* sample size problems they will agree. Nevertheless there exist distributions where the approximations are poor (Fig. 11.5). As we saw in the analogous case in supervised learning (Sect. **??**), 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 however are often easier to implement.

Of course, if $p(\boldsymbol{\theta})$ has been obtained by supervised learning using a large set of labelled samples, it will be far from uniform, and it will have a dominant influence on $p(\boldsymbol{\theta}|\mathcal{H}^n)$ when n is small. Equation 27 shows how the observation of an additional unlabelled sample modifies our opinion about the true value of $\boldsymbol{\theta}$, and emphasizes the ideas of updating and learning. If the mixture density $p(\mathbf{x}|\boldsymbol{\theta})$ is identifiable, then each additional sample tends to sharpen $p(\boldsymbol{\theta}|\mathcal{H}^n)$, and under fairly general conditions $p(\boldsymbol{\theta}|\mathcal{H}^n)$ can be shown to converge (in probability) to a Dirac delta function centered at the true value of $\boldsymbol{\theta}$ (Problem 34). 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 $p(\mathbf{x}|\omega_i, \boldsymbol{\theta})$.

Figure 11.5: Figure which agree and where disagree.

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 problem 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. (For instance, in multilayer neural networks, there may be a large number of total weight vectors that lead to the same classification boundaries.) However, since all of these yield the same component density, lack of identifiability presents no theoretical difficulty. With unsupervised learning, lack of identifiability is much more serious. When $\boldsymbol{\theta}$ cannot be determined uniquely, the mixture cannot be decomposed into its true components. Thus, while $p(\mathbf{x}|\mathcal{H}^n)$ may still converge to $p(\mathbf{x})$, $p(\mathbf{x}|\omega_i, \mathcal{H}^n)$ given by Eq. 24 will not in general converge to $p(\mathbf{x}|\omega_i)$, and a theoretical barrier to learning exists. It is here that a few *labelled* 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 every finding simple exact solutions for $p(\boldsymbol{\theta}|\mathcal{H})$. Such solutions are tied to the existence of a simple sufficient statistic (Sect. **??**.**??**), and the factorization theorem requires the ability to factor $p(\mathcal{H}|\boldsymbol{\theta})$ as

$$p(\mathcal{H}|\boldsymbol{\theta}) = g(\mathbf{s}, \boldsymbol{\theta}) h(\mathcal{H}). \tag{30}$$

But from Eqs. 26 & 1,

$$p(\mathcal{H}|\boldsymbol{\theta}) = \prod_{k=1}^{n} \Big[\sum_{j=1}^{c} p(\mathbf{x}_k|\omega_j, \boldsymbol{\theta}_j)P(\omega_j) \Big]. \tag{31}$$

Thus, $p(\mathcal{H}|\boldsymbol{\theta})$ 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 $\mathbf{x}_1, ..., \mathbf{x}_n$ bearing a particular labelling, with the sum extending over all of the ways that the samples could be labelled. 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, so that as $\boldsymbol{\theta}$ varies only one term in the mixture density is non-zero. In that case, $p(\mathcal{H}|\boldsymbol{\theta})$ is the product of the n nonzero terms, and may possess a simple sufficient statistic. However, since 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. 27 and obtain

$$p(\boldsymbol{\theta}|\mathcal{H}^n) = \frac{\sum\limits_{j=1}^{c} p(\mathbf{x}_n|\omega_j, \boldsymbol{\theta}_j)P(\omega_j)}{\sum\limits_{j=1}^{c} \int p(\mathbf{x}_n|\omega_j, \boldsymbol{\theta}_j)P(\omega_j)p(\boldsymbol{\theta}|\mathcal{H}^{n-1})\ d\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{H}^{n-1}). \tag{32}$$

If we consider the special case where $P(\omega_1) = 1$ and all the other a priori probabilities are zero, corresponding to the supervised case in which all samples come from Class ω_1, then Eq. 32 simplifies to

$$p(\boldsymbol{\theta}|\mathcal{H}^n) = \frac{p(\mathbf{x}_n|\omega_1, \boldsymbol{\theta}_1)}{\int p(\mathbf{x}_n|\omega_1, \boldsymbol{\theta}_1)p(\boldsymbol{\theta}|\mathcal{H}^{n-1})\ d\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{H}^{n-1}). \tag{33}$$

Let us compare Eqs. 32 & 33 to see how observing an additional sample changes our estimate of $\boldsymbol{\theta}$. In each case we can ignore the denominator, which is independent of $\boldsymbol{\theta}$. Thus, the only significant difference is that in the supervised case we multiply the "a priori" 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 $\boldsymbol{\theta}$. Since $\mathbf{x}_n$ 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 2: 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 $\mu, P(\omega_1)$ and $P(\omega_2)$ are known. Here

$$p(x|\theta) = \frac{P(\omega_1)}{\sqrt{2\pi}} \exp\ [\Leftrightarrow \frac{1}{2}(x \Leftrightarrow \mu)^2] + \frac{P(\omega_2)}{\sqrt{2\pi}} \exp\ [\Leftrightarrow \frac{1}{2}(x \Leftrightarrow \theta)^2].$$

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 a priori density $p(\theta)$ is uniform from a to b. Then after one observation $(x = x_1)$ we have

$$\begin{aligned} p(\theta|x_1) &= \alpha p(x_1|\theta)p(\theta) \\ &= \left\{ \begin{array}{ll} \alpha'\{P(\omega_1)\exp[\Leftrightarrow\frac{1}{2}(x_1 \Leftrightarrow\mu)^2]+ & \\ \qquad P(\omega_2)\exp\ [\Leftrightarrow\frac{1}{2}(x_1 \Leftrightarrow\theta)^2]\} & a \leq \theta \leq b \\ 0 \qquad\quad \text{otherwise} & \end{array} \right\}, \end{aligned}$$

where α and α' are normalizing constants, independent of θ. If the sample x_1 is in the range $a \leq x \leq b$, 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 $[\Leftrightarrow(1/2)(x_1 \Leftrightarrow\mu)^2]$ 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

$$\begin{aligned} p(\theta|x_1, x_2) &= \beta p(x_2|\theta)p(\theta|x_1) \\ &= \left\{ \begin{array}{l} \beta'\{P(\omega_1)P(\omega_1)\exp\ [\Leftrightarrow\frac{1}{2}(x_1 \Leftrightarrow\mu)^2 \Leftrightarrow\frac{1}{2}(x_2 \Leftrightarrow\mu)^2] \\ \ +[P(\omega_1)P(\omega_2)\exp\ [\Leftrightarrow\frac{1}{2}(x_1 \Leftrightarrow\mu)^2 \Leftrightarrow\frac{1}{2}(x_2 \Leftrightarrow\theta)^2] \\ \ +[P(\omega_2)P(\omega_1)\exp\ [\Leftrightarrow\frac{1}{2}(x_1 \Leftrightarrow\theta)^2 \Leftrightarrow\frac{1}{2}(x_2 \Leftrightarrow\mu)^2] \\ \ +[P(\omega_2)P(\omega_2)\exp\ [\Leftrightarrow\frac{1}{2}(x_1 \Leftrightarrow\theta)^2 \Leftrightarrow\frac{1}{2}(x_2 \Leftrightarrow\theta)^2]\} \\ \qquad\qquad\qquad\qquad\qquad a \leq \theta \leq b \\ 0 \qquad\qquad\qquad\qquad\quad \text{otherwise.} \end{array} \right. \end{aligned}$$

Unfortunately, the primary thing we learn from this expression is that $p(\theta|\mathcal{H}^n)$ 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

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

and numerical integration to obtain an approximate numerical solution for $p(\theta|\mathcal{H}^n)$. This was done for the data we have seen using the values $\mu = 2$, $P(\omega_1) = 1/3$, and $P(\omega_2) = 2/3$. An a priori density $p(\theta)$ uniform from -4 to +4 encompasses the data in the table. When this was used to start the recursive computation of $p(\theta|\mathcal{H}^n)$, the results shown in Fig. 11.6 were obtained. As n goes to infinity we can confidently expect $p(\theta|\mathcal{H}^n)$ 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 appears in the presence of the a priori density $p(\theta)$. Figure 11.7 shows how $p(\theta|\mathcal{H}^n)$ 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 analog of Chapter **??**) that the differences between the Bayesian and the maximum

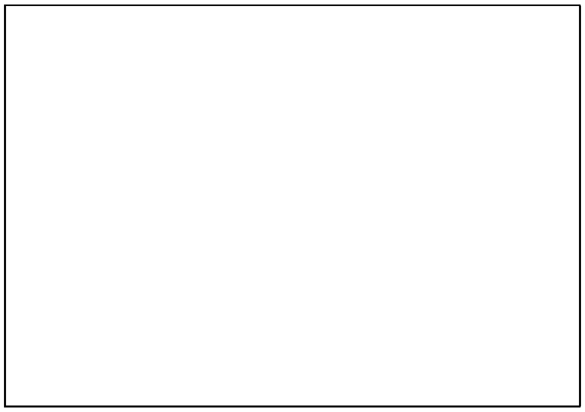

Figure 11.6: Contours of a log-likelihood function for the data used before.

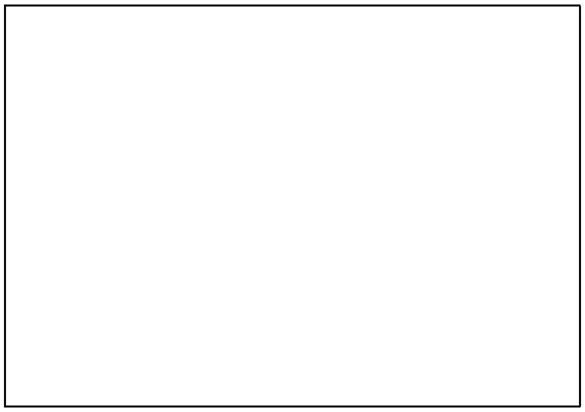

Figure 11.7: Contours of a log-likelihood function for the data used before.

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 unlabelled samples is several times the effective number of labelled samples used to determine $p(\theta)$.

11.5.3 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 (Problem 28). 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.

Since the basic 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 a priori 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 unlabelled 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 labelled. 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 unlabelled 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, the resulting labelling will not in general 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. The literature contains a few rather complicated analyses of particular decison-directed procedures, and numerous reports of experimental results. The basic conclusions are 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.

11.6 Data Description and Clustering

Let us reconsider our original problem of learning something of use from a set of unlabelled samples. Viewed geometrically, these samples may form clouds of points in a d-dimensional space. Suppose that we knew 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 $\mathbf{x}$ that best represents all of the data in the sense of minimizing the sum of squared distances from $\mathbf{x}$ to the samples. The sample covariance matrix tells us how well the sample mean describes the data in terms of the amount of scatter that exists in various directions. 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 11.8 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.

By assuming that the samples come from a mixture of c normal distributions, we can approximate a greater variety of situations. In essence, this corresponds to

Figure 11.8: Data sets having identical second-order statistics, i.e., the same mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.

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 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 a priori 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 **??** 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.

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. The more formal 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 ground that they are easy to apply and often yield interesting results that may guide the application of more rigorous procedures.

11.7 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 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 acutally involves two separate issues:

- How should one measure the similarity between samples?

Figure 11.9: The effect of distance threshold on clustering — lines are drawn between points closer than a distance d_0 apart. a) Large d_0. b) Intermediate d_0. c) Small d_0.

- 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 distance function 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 signifiantly 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 the samples will be assigned to one cluster. If d_0 is very small, each sample will form an isolated cluster. To obtain "natural" clusters, d_0 will have to be greater than the typical within-cluster distances and less than typical between-cluster distances (Fig. 11.9).

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 justified if the feature space is isotropic; consequently 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. 11.10 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. 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* can be preceded and/or followed by normalization for scale.

Figure 11.10: The effect of scaling on the apparent clustering.

Figure 11.11: The undesirable effects of normalization.

However, the reader 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 (Sect. **??**.**??**). 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. 11.11). Thus, this routine normalization may be less than helpful in the cases of greatest interest.[*] Section **??** describes some better ways to obtain invariance to scaling.

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

[*] In backpropagation, one of the goals for such preprocessing and scaling of data was to increase learning speed; in contrast, such preprocessing does not significantly affect the speed of these clustering algorithms.

Figure 11.12: The set of points equidistant from the origin for the Minkowski metric with different values of q. For $q = 2$ (i.e., the Euclidean metric) the set is a d-dimensional hypersphere; for $q = 1$ the city block metric.

$$d(\mathbf{x}, \mathbf{x}') = \left(\sum_{k=1}^{d} |x_k \Leftrightarrow x'_k|^q \right)^{1/q}, \tag{34}$$

where $q \geq 1$ is a selectable parameter — the general *Minkowski metric.* Setting $q = 2$ gives the familiar Euclidean metric while setting $q = 1$ the Manhattan or *city block* metric — the sum of the absolute distances along each of the d coordinate axes (Fig. 11.12). 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.

MINKOWSKI METRIC

CITY BLOCK

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 functions 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

SIMILARITY FUNCTION

$$s(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^t \mathbf{x}'}{\|\mathbf{x}\| \; \|\mathbf{x}'\|} \tag{35}$$

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), this similarity functions has a simple non-geometrical interpretation in terms of measuring 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 variation are

$$s(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^t \mathbf{x}'}{d}, \tag{36}$$

the fraction of attributes shared, and

$$s(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^t\mathbf{x}'}{\mathbf{x}^t\mathbf{x} + \mathbf{x}'^t\mathbf{x}' \Leftrightarrow \mathbf{x}^t\mathbf{x}'}, \tag{37}$$

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. Other measures of similarity arise in other applications, the variety of measures testifying to the diversity of problem domains.

One might wish to insure invariance transformations in the *feature* space, and thus tangent distance (Sect. **??**.**??**) would be appropriate. Rewritten in our current terminology, the (two-sided) tangent distance is

$$d(\mathbf{x}, \mathbf{x}') = \min_{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2} ||\mathbf{x} + \boldsymbol{\alpha}_1\mathbf{T}_1(\mathbf{x}) \Leftrightarrow (\mathbf{x}' + \boldsymbol{\alpha}_2\mathbf{T}_2(\mathbf{x}'))||^2, \tag{38}$$

where the columns of $\mathbf{T}_1$ and $\mathbf{T}_2$ represent vectors corresponding to different transformations of $\mathbf{x}$ and $\mathbf{x}'$, respectively. That is, the tangent distance is the minimum (squared) Euclidean distance after the optimal linear transformation have been performed on the patterns (Sect. **??**.**??**). In the case of 2D spatial patterns could represent transformations such as rotation, translation, scale, and so on.

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 one 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 does one 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 user 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 labelled (e.g., by resorting to a teacher or small number of labelled 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 6).

11.8 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 $\mathcal{H}$ of n samples $\mathbf{x}_1, ..., \mathbf{x}_n$ that we want to partition into exactly c disjoint subsets $\mathcal{H}_1, ..., \mathcal{H}_c$. Each subset is to represent a cluster, with samples in the same cluster begin somehow more similar than samples in different

* These fundamental considerations are by no means unique to clustering. They appear, for example, whenever one chooses a parametric form for an unknown probability density function, a metric for non-parametric density estimation, or scale factors for linear discriminant functions. Clustering problems merely expose them more clearly.

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.

11.8.1 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 $\mathcal{H}_i$ and let $\mathbf{m}_i$ be the mean of those samples,

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{H}_i} \mathbf{x}. \tag{39}$$

Then the sum of squared errors is defined by

$$J_e = \sum_{i=1}^{c} \sum_{\mathbf{x} \in \mathcal{H}_i} \|\mathbf{x} \Leftrightarrow \mathbf{m}_i\|^2. \tag{40}$$

This criterion function has a simple interpretation: for a given cluster $\mathcal{H}_i$, the mean vector $\mathbf{m}_i$ is the best representative of the samples in $\mathcal{H}_i$ in the sense that it minimizes the sum of the squared lengths of the "error" vectors $\mathbf{x} \Leftrightarrow \mathbf{m}_i$ in $\mathcal{H}_i$. Thus, J_e measures the total squared error incurred in representing the n samples $\mathbf{x}_1, ..., \mathbf{x}_n$ by the c cluster centers $\mathbf{m}_1, ..., \mathbf{m}_c$. The value of J_e depends on how the samples are grouped into clusters, and an 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 essentially compact clouds that are rather well separated from one another. It should work well for the two or three clusters in Fig. 11.13, but one would not expect reasonable results for the data in Fig. 11.14. A less obvious problem arises when there are great differences in the number of samples in different clusters. In that case it can happen that a partition that splits a large cluster is favored over one that maintains the integrity of the clusters merely because the slight reduction in squared error achieved is multiplied by many terms in the sum (Fig. 11.15). 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. Since 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.

11.8.2 Related Minimum Variance Criteria

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

$$J_e = \frac{1}{2} \sum_{i=1}^{c} n_i \bar{s}_i, \tag{41}$$

where

Figure 11.13: A two-dimensional section of the Anderson iris data.

Figure 11.14: The Herzsprung-Russell Diagram.

Figure 11.15: The problem of splitting large clusters: The sum of squared error is smaller for a) than for b).

$$\bar{s}_i = \frac{1}{n^2} \sum_{\mathbf{x} \in \mathcal{H}_i} \sum_{\mathbf{x}' \in \mathcal{H}_i} \|\mathbf{x} \Leftrightarrow \mathbf{x}'\|^2. \tag{42}$$

Equation 42 leads us to interpret $\bar{s}_i$ as the average squared distance between points in the ith cluster, and 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

$$\bar{s}_i = \frac{1}{n_i^2} \sum_{\mathbf{x} \in \mathcal{H}_i} \sum_{\mathbf{x}' \in \mathcal{H}_i} s(\mathbf{x}, \mathbf{x}') \tag{43}$$

or

$$\bar{s}_i = \min_{\mathbf{x}, \mathbf{x}' \in \mathcal{H}_i} s(\mathbf{x}, \mathbf{x}'). \tag{44}$$

As before, we define an optimal partioning 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.

11.8.3 Scattering Criteria

The scatter matrices

Another interesting class of criterion functions can be derived from the scatter matrices used in multiple discriminant analysis. The following definitions directly parallel the definitions given in Sect.??.??.

	Depend on cluster center?		
	Yes	No	
Mean vector for the ith cluster		$\times$	$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{H}_i} \mathbf{x}$ (45)
Total mean vector		$\times$	$\mathbf{m} = \frac{1}{n} \sum_{\mathcal{H}} \mathbf{x} = \frac{1}{n} \sum_{i=1}^{c} n_i \mathbf{m}_i$ (46)
Scatter matrix for the ith cluster	$\times$		$\mathbf{S}_i = \sum_{\mathbf{x} \in \mathcal{H}_i} (\mathbf{x} \Leftrightarrow \mathbf{m}_i)(\mathbf{x} \Leftrightarrow \mathbf{m}_i)^t$ (47)
Within-cluster scatter matrix	$\times$		$\mathbf{S}_W = \sum_{i=1}^{c} \mathbf{S}_i$ (48)
Between-cluster scatter matrix	$\times$		$\mathbf{S}_B = \sum_{i=1}^{c} n_i (\mathbf{m}_i \Leftrightarrow \mathbf{m})(\mathbf{m}_i \Leftrightarrow \mathbf{m})^t$ (49)
Total scatter matrix		$\times$	$\mathbf{S}_T = \sum_{\mathbf{x} \in \mathcal{H}} (\mathbf{x} \Leftrightarrow \mathbf{m})(\mathbf{x} \Leftrightarrow \mathbf{m})^t$ (50)

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

$$\mathbf{S}_T = \mathbf{S}_W + \mathbf{S}_B. \tag{51}$$

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 do depend on the partitioning, however. Roughly speaking, there is an exchange between these two matrices, the between-cluster scatter going up as the within-cluster scatter goes down. This is fortunate, since 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, since 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, since the definitions of scatter matrices (Eqs. 47 & 48) yield

$$tr\ \mathbf{S}_W = \sum_{i=1}^{c} tr\ \mathbf{S}_i = \sum_{i=1}^{c} \sum_{\mathbf{x} \in \mathcal{H}_i} \|\mathbf{x} \Leftrightarrow \mathbf{m}_i\|^2 = J_e. \tag{52}$$

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

$$trS_B = \sum_{i=1}^{c} n_i \|\mathbf{m}_i \Leftrightarrow \mathbf{m}\|^2. \tag{53}$$

The Determinant Criterion

In Sect. **??** we used the determinant of the scatter matrix to obtain a scalar measure of scatter. Roughly speaking, this measures the square of the scattering volume, since it is proportional to the product of the variances in the directions of the principal axes. Since $\mathbf{S}_B$ 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 $n \Leftrightarrow c$ is less than the dimensionality d.* However, if we assume that $\mathbf{S}_W$ is nonsingular, we are led to consider the criterion

* This follows from the fact that the rank of $\mathbf{S}_i$ can not exceed $n_i - 1$, and thus the rank of $\mathbf{S}_W$ can not exceed $\sum(n_i - 1) = n - c$. Of course, if the samples are confined to a lower dimensional subspace it is possible to have $\mathbf{S}_W$ be singular even though $n - c \geq d$. In such cases, some kind of dimensionality-reduction procedure must be used before the determinant criterion can be applied (see Sect.**??**).

function

$$J_d = |\mathbf{S}_W| = \Big|\sum_{i=1}^{c} \mathbf{S}_i\Big|. \tag{54}$$

The partition that minimizes J_d is often similar to the one that minimizes J_e, but the two need not be the same. 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 29). 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 $\lambda_1, ..., \lambda_d$ or $\mathbf{S}_W^{-1}\mathbf{S}_B$ are invariant under nonsingular linear transformations of the data (Problem 31). 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 pointed out in Sect. **??**, 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, since the trace of a matrix is the sum of its eigenvalues, one might elect to maximize the criterion function*

$$tr\mathbf{S}_W^{-1}\mathbf{S}_B = \sum_{i=1}^{d} \lambda_i. \tag{55}$$

By using the relation $\mathbf{S}_T = \mathbf{S}_W + \mathbf{S}_B$, one can derive the following invariant relatives of tr $\mathbf{S}_W$ and $|\mathbf{S}_W|$ (Problem 24):

$$tr\mathbf{S}_T^{-1}\mathbf{S}_W = \sum_{i=1}^{d} \frac{1}{1+\lambda_i} \tag{56}$$

and

$$\frac{|\mathbf{S}_W|}{|\mathbf{S}_T|} = \prod_{i=1}^{d} \frac{1}{1+\lambda_i}. \tag{57}$$

Since 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 (Fig. 11.16).

* Another invariant criterion is

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

though since its value is usually zero it is not very useful.

Figure 11.16: For more than two clusters, linear transformations can change the clustering.

Figure 11.17: a) for ???, b) for ??? c) for ??? for the data in the Table.

Example 3: Clustering criteria

We can gain insight into these criteria by applying them to the following data set.

sample	x_1	x_2	x_3
1	0.35	0.366	1.39
2	0.305	0.421	1.02
3	0.395	0.287	1.66
4	0.312	0.831	1.51
5	0.345	0.214	1.36
6	0.363	0.205	1.58
7	0.319	0.903	0.995
8	0.346	0.203	1.54
9	0.363	0.212	1.35
10	0.368	1.03	1.19
11	0.395	1.04	1.11
12	0.366	0.579	1.31
13	0.307	0.354	1.65
14	0.319	0.655	1.14
15	0.317	0.593	1.66
16	0.34	0.205	1.07
17	0.379	0.619	1.36
18	0.308	0.208	1.28
19	0.328	0.253	0.998
20	0.379	0.213	1.29

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 $tr\mathbf{S}_T^{-1}\mathbf{S}_W$ is equivalent to minimizing the sum-of-squared-error criterion tr $\mathbf{S}_W$ after performing this normalization. Clearly, this criterion suffers from the very defects that we warned about in Sect. **??**, 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 posses 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. For such cases one must devise other criterion functions that are better matched to the structure present or being sought.

11.9 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. Since 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 22). 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 significant.

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

$$J_e = \sum_{i=1}^{c} J_i, \tag{58}$$

where an effective error per cluster is

$$J_i = \sum_{\mathbf{x}\in\mathcal{H}_i} \|\mathbf{x} \Leftrightarrow \mathbf{m}_i\|^2 \tag{59}$$

and the mean of each cluster is, as before,

$$\mathbf{m}_i = \frac{1}{n_i}\sum_{\mathbf{x}\in\mathcal{H}_i} \mathbf{x}. \tag{39}$$

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

$$\mathbf{m}_j^* = \mathbf{m}_j + \frac{\hat{\mathbf{x}} \Leftrightarrow \mathbf{m}_j}{n_j + 1} \tag{60}$$

and J_j increases to

$$\begin{aligned} J_j^* &= \sum_{\mathbf{x}\in\mathcal{H}_i} \|\mathbf{x} \Leftrightarrow \mathbf{m}_j^*\|^2 + \|\hat{\mathbf{x}} \Leftrightarrow \mathbf{m}_j^*\|^2 \\ &= \left(\sum_{\mathbf{x}\in\mathcal{H}_i} \|\mathbf{x} \Leftrightarrow \mathbf{m}_j \Leftrightarrow \frac{\hat{\mathbf{x}} \Leftrightarrow \mathbf{m}_j}{n_j+1}\|^2\right) + \|\frac{n_j}{n_j+1}(\hat{\mathbf{x}} \Leftrightarrow \mathbf{m}_j)\|^2 \\ &= J_j + \frac{n_j}{n_j+1}\|\hat{\mathbf{x}} \Leftrightarrow \mathbf{m}_j\|^2. \end{aligned} \tag{61}$$

Under the assumption that $n_i \neq 1$ (singleton clusters should not be destroyed), a similar calculation (Problem 30) shows that $\mathbf{m}_i$ changes to

$$\mathbf{m}_i^* = \mathbf{m} \Leftrightarrow \frac{\hat{\mathbf{x}} \Leftrightarrow \mathbf{m}_i}{n_i \Leftrightarrow 1} \tag{62}$$

and J_i decreases to

$$J_i^* = J_i \Leftrightarrow \frac{n_i}{n_i \Leftrightarrow 1}\|\hat{\mathbf{x}} \Leftrightarrow \mathbf{m}_j\|^2. \tag{63}$$

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

$$\frac{n_i}{n_i \Leftrightarrow 1}\|\hat{\mathbf{x}} \Leftrightarrow \mathbf{m}_i\|^2 > \frac{n_j}{n_j+1}\|\hat{\mathbf{x}} \Leftrightarrow \mathbf{m}_j\|^2, \tag{64}$$

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 $n_j/(n_j+1)\|\hat{\mathbf{x}} \Leftrightarrow \mathbf{m}_j\|^2$ is minimum. This leads to the following clustering procedure:

```
                      Basic Minimum Squared Error Clustering

Initialize select partition of $n$ samples,
  compute $J_e$ and $c$ means $\mathbf{m}_1, \ldots, \mathbf{m}_c$               initialize
Do Select a candidate sample $\hat{\mathbf{x}}$;                    next sample
    suppose $\hat{\mathbf{x}} \in \mathcal{H}_i$
  If $n_i = 1$, Go to *, Else compute                 singleton cluster?
  $\rho_j = \begin{cases} \frac{n_j}{n_j+1}\|\hat{\mathbf{x}} - \mathbf{m}_j\|^2 & j \neq i \\ \frac{n_j}{n_j-1}\|\hat{\mathbf{x}} - \mathbf{m}_i\|^2 & j = i. \end{cases}$       criterion
If $\rho_k \leq \rho_j$ for all j, transfer $\hat{\mathbf{x}}$ to $\mathcal{H}_k$             improve criterion
Update $J_e, \mathbf{m}_i$ and $\mathbf{m}_k$                                recompute
* If $J_e$ has not changed in $n$ attempts, End  local optimum found
  Else Return                                         local optimum not yet found
End
```

If this procedure is compared to the `Basic K-means procedure` described in Sect. ??, it is clear that the former is essentially a sequential version of the latter. Where the `Basic K-means procedure` waits until all n samples have been reclassified before updating, the `Basic 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. Alternatively, repetition with different random selections can give some indication of the sensitivity of the solution to the starting point. Yet another aproach is to find the c-cluster starting point from the solutions to the $(c - a)$-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 $(c - a)$-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.

11.10 Hierarchical Clustering

Up to now, our methods have formed disjoint clusters — in computer science terminology, we would say that the data description is "flat." There are many times when clusters have subclusters, however, as in biological taxonomy, where individuals are grouped into species, speices into genera, genera into families, and so on. 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.

Figure 11.18: A dendrogram for hierarchical clustering. The vertical axis represents a generalized distance between groupings.

11.10.1 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 $n \Leftrightarrow 1$ clusters, the next a partition into $n \Leftrightarrow 2$, 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 $c = n \Leftrightarrow k + 1$. Thus, level one corresponds to n clusters and level n 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*, that shows how the samples are grouped. Figure 11.18 shows a dendrogram for a hypothetical problem involving six samples. Level 1 shows the six samples as singleton clusters. At level 2, samples x_3 and x_5 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. 11.18, for example, the similarity between the two groups of samples that are merged at level 6 has a value of 30. The similarity values are often used to help determine whether the groupings are natural or forced. For our hypothetical example, one would be inclined to say that the groupings at levels 4 or 5 are natural, but that the large reduction in similarity needed to go to level 6 makes that grouping forced. We shall see shortly how such similarity values can be obtained.

Agglomerative

divisive

Because of their conceptual simplicity, hierarchical clustering procedures are among the best-known methods. The procedures themselves can be divided into two distinct classes — agglomerative and divisive. *Agglomerative* (bottom-up, clumping) procedures start with n singleton clusters and form the sequence by successively merging clusters (as we saw). *Divisive* (top-down, splitting) procedures start with all of the samples in one cluster and form the sequence by successively splitting clusters.

* Another representation for hierarchical clustering is Venn diagrams, in which each level of cluster may contain sets that are subclusters. However, it is more difficult to represent quantitatively the distances in such diagrams, and thus we concentrate on tree structures.

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 limit our attention to the agglomerative procedures, referring the reader to the literature for divisive methods.

11.10.2 Agglomerative Hierarchical Clustering

The major steps in agglomerative clustering are contained in the following procedure:

```
                    Basic agglomerative clustering

Initialize $c, \hat{c} = n, \mathcal{H}_i = \{\mathbf{x}_i\}, i = 1, \ldots, n$      $c$ desired clusters
Do Find nearest clusters, e.g., $\mathcal{H}_i$ and $\mathcal{H}_j$  find candidate clusters
  Merge $\mathcal{H}_i$ and $\mathcal{H}_j$                            merge clusters
  Let $\hat{c} = \hat{c} \Leftrightarrow 1$                              decrement clusters
Until $\hat{c} = c$                                   desired number found
End
```

As described, this procedure terminates when the specified number of clusters has been obtained. However, if we continue until $c = 1$ we can produce a dendrogram like that shown in Fig. 11.18. At any level the "distance" between nearest clusters can provide 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 at any stage. The considerations here are much like those involved in selecting a criterion function. For simplicity, we shall generally restrict our attention to the following distance measures:

$$
\begin{aligned}
d_{min}(\mathcal{H}_i, \mathcal{H}_j) &= \min_{\substack{\mathbf{x} \in \mathcal{H}_i \\ \mathbf{x}' \in \mathcal{H}_j}} \|\mathbf{x} \Leftrightarrow \mathbf{x}'\| \\
d_{max}(\mathcal{H}_i, \mathcal{H}_j) &= \max_{\substack{\mathbf{x} \in \mathcal{H}_i \\ \mathbf{x}' \in \mathcal{H}_j}} \|\mathbf{x} \Leftrightarrow \mathbf{x}'\| \\
d_{avg}(\mathcal{H}_i, \mathcal{H}_j) &= \frac{1}{n_i n_j} \sum_{\mathbf{x} \in \mathcal{H}_i} \sum_{\mathbf{x}' \in \mathcal{H}_j} \|\mathbf{x} \Leftrightarrow \mathbf{x}'\| \\
d_{mean}(\mathcal{H}_i, \mathcal{H}_j) &= \|\mathbf{m}_i \Leftrightarrow \mathbf{m}_j\|. \qquad (65)
\end{aligned}
$$

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 use the two-dimensional point sets shown in Fig. 11.19 to 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 $d_{min}(\mathcal{H}_i, \mathcal{H}_j)$ defined in Eq. 65. We will, once and for all, need to calculate $n(n \Leftrightarrow 1)$ inter-point distances (each of which is an $O(d^2)$ calculation) and place the results in an inter-point distance table. Our 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 $O(n(n \Leftrightarrow 1)(d^2 + 1)) = O(n^2 d^2)$.

Figure 11.19: Three data sets, each of which responds differently to different clustering procedures (see text).

For an arbitrary agglomeration step (i.e., from $\hat{c}$ to $\hat{c} \Leftrightarrow 1$), we need merely step through the $n(n \Leftrightarrow 1) \Leftrightarrow \hat{c}$ "unused" distances in the list and find the smallest for which $\mathbf{x}$ and $\mathbf{x}'$ lie in different clusters. This is, again, $O(n(n \Leftrightarrow 1) \Leftrightarrow \hat{c})$. If we assume the typical conditions that $n \gg c$, the time complexity is thus $O(cn^2d^2)$.*

The Nearest-Neighbor Algorithm

Consider the algorithm's behavior when d_{min} is used.* 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 $\mathcal{H}_i$. When d_{min} is used to measure the distance between subsets, the nearest neighbors determine the nearest subsets. The merging of $\mathcal{H}_i$ and $\mathcal{H}_j$ corresponds to adding an edge between the nearest pair of nodes in $\mathcal{H}_i$ and $\mathcal{H}_j$. Since 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 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. Thus, with the use of d_{min} as the distance measure, the agglomerative clustering procedure becomes an algorithm for generating a *minimal spanning tree*.

SPANNING TREE

Figure 11.20 shows the results of applying this procedure to the data of Fig. 11.19. In all cases the procedure was stopped at $c = 2$; 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 compact and well separated, the obvious clusters are found. In the second case, the presence of a few points 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

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

* In the literature, the resulting procedure is often called the *nearest-neighbor* or the *minimum* algorithm. If it is terminated when the distance between nearest clusters exceed an arbitrary threshold, it is called the *single-linkage* algorithm.

Figure 11.20: Results of the nearest-neighbor clustering algorithm.

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. However, as the third case illustrates, this very tendency to form chains can be advantageous if the clusters are elongated or possess elongated limbs.

The Farthest-Neighbor Algorithm

When d_{max} is used to measure the distance between subsets, the growth of elongated clusters is discouraged.* 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. If we define the *diameter* of a cluster as the largest distance between points in the cluster, then the distance between two clusters is merely the diameter of their union. 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 possible. As Fig. 11.21 illustrates, this is advantageous when the true clusters are compact and roughly equal in size. However, 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.

COMPLETE SUBGRAPH

CLUSTER DIAMETER

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} and d_{mean} are natural compromises between d_{min} and d_{max}. Computationally, d_{mean} is the simplest of all of these measures, since the others require computing all $n_i n_j$ pairs of distances $\|\mathbf{x} \Leftrightarrow \mathbf{x}'\|$. However, a

* In the literature, the resulting procedure is often called the *farthest-neighbor* or the *maximum* algorithm. If it is terminated when the distance between nearest clusters exceeds an arbitrary threshold, it is called the *complete-linkage* algorithm.

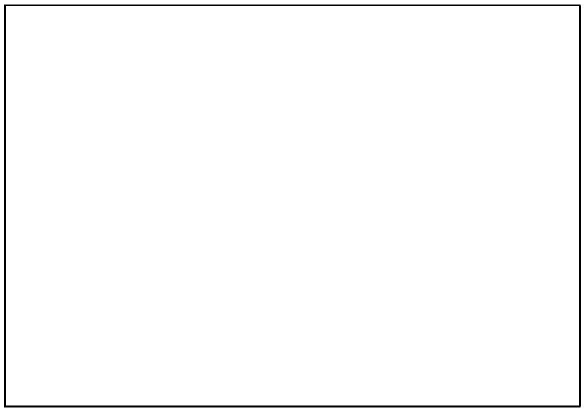

Figure 11.21: Results of the farthest-neighbor clustering algorithm.

measure such as d_{avg} can be used when the distances $\|\mathbf{x} \Leftrightarrow \mathbf{x}'\|$ are replaced by similarity measures, where the similarity between mean vectors may be difficult or impossible to define.

11.10.3 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, one 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. However, with a simple modification it is possible to obtain a stepwise-optimal procedure for extremizing a criterion function. This is done merely by replacing one step of the `Basic Agglomerative Clustering Procedure` (Sect. **??**) to get:

```
                Stepwise optimal hierarchical clustering

Initialize $c, \hat{c} = n, \mathcal{H}_i = \{\mathbf{x}_i\}, i = 1, \ldots, n$     $c$ desired clusters
Do Find clusters $\mathcal{H}_i$ and $\mathcal{H}_j$
     whose merger would increase
     or decrease the criterion
     as little as possible                        find candidate clusters
  Merge $\mathcal{H}_i$ and $\mathcal{H}_j$                             Merge clusters
  Let $\hat{c} = \hat{c} \Leftrightarrow 1$                              decrement clusters
Until $\hat{c} = c$                                     desired number found
End
```

We saw earlier that the use of d_{max} 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 Sect. **??**, we find that the pair of clusters whose merger increases J_e as little as possible is the pair for which the "distance"

$$d_e(\mathcal{H}_i, \mathcal{H}_j) = \sqrt{\frac{n_i n_j}{n_i + n_j}} \|\mathbf{m}_i \Leftrightarrow \mathbf{m}_j\| \qquad (66)$$

is minimum. 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 tends to favor growth by adding singletons or small clusters to 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.

11.10.4 Hierarchical Clustering and Induced Metrics

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 $\delta(\mathbf{x}, \mathbf{x}') \geq 0$, equality holding if an 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

DISSIMIL-ARITY

$$\delta_{min}(\mathcal{H}_i, \mathcal{H}_j) = \min_{\substack{\mathbf{x} \in \mathcal{H}_i \\ \mathbf{x}' \in \mathcal{H}_j}} \delta(\mathbf{x}, \mathbf{x}') \tag{67}$$

or

$$\delta_{max}(\mathcal{H}_i, \mathcal{H}_j) = \max_{\substack{\mathbf{x} \in \mathcal{H}_i \\ \mathbf{x}' \in \mathcal{H}_j}} \delta(\mathbf{x}, \mathbf{x}') \tag{68}$$

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 25).

We can now define the *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 three things:

METRIC

1. $d(\mathbf{x}, \mathbf{x}') = 0 \Leftrightarrow \mathbf{x} = \mathbf{x}'$	uniqueness
2. $d(\mathbf{x}, \mathbf{x}') = d(\mathbf{x}', \mathbf{x})$	symmetry
3. $d(\mathbf{x}, \mathbf{x}'') \leq d(\mathbf{x}, \mathbf{x}') + d(\mathbf{x}', \mathbf{x}'')$	triangle inequality

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

$$d(\mathbf{x}, \mathbf{x}'') \leq max[d(\mathbf{x}, \mathbf{x}'), d(\mathbf{x}', \mathbf{x}'')] \text{ for any } \mathbf{x}' \tag{69}$$

in which case we say that $d(\cdot, \cdot)$ is an *ultrametric* (Problem 16). Ultrametric criteria can be more immune to local minima problems since stricter ordering of distances among clusters is maintained.

ULTRA-METRIC

11.11 Graph Theoretic Methods

In two or three instances we have used linear graphs to add insight into the nature of certain clustering procedures. Where the mathematics of normal mixtures and minimum-variance partitions seems to keep returning us to the picture of clusters as isolated clumps of points, 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. 11.9. Here a threshold distance d_0 was selected, and two points were said to be in the same cluster if the distance between them was 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 $\mathbf{x}$ is similar to $\mathbf{x}'$ if $s(\mathbf{x}, \mathbf{x}') > s_0$. This defines an n-by-n *similarity matrix* $\mathbf{S} = [s_{ij}]$, where

SIMILARITY MATRIX

$$s_{ij} = \begin{cases} 1 & \text{if } s(\mathbf{x}_i, \mathbf{x}_j) > s_0 \\ 0 & \text{otherwise.} \end{cases} \tag{70}$$

SIMILARITY GRAPH

This matrix defines 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 $\mathbf{x}, \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k, \mathbf{x}'$ 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

In the preceding section we noted 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 an inverted way of obtaining a divisive hierarchical procedure, and 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 11.22 shows a minimal spanning tree for a two-dimensional point set and the clusters obtained by systematically removing all edges for which $l > 2\bar{l}$. Note how the sensitivity of this criterion to local conditions gives results that are quite different from merely removing the two longest edges.

INCON-SISTENT EDGE

When the data points are strung out in to 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 branching off the diameter path. In contrast, for a large, uniform cloud of data 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.

DIAMETER PATH

Figure 11.22: Clusters formed by removing inconsistent edges. a) Point set. b) Minimal spanning tree. c) Clusters.

Figure 11.23: A minimal spanning tree with bimodal edge length distribution.

One of the useful statistics that can be obtained from a minimal spanning tree is the edge length distribution. Figure 11.23 shows a situation in which a dense cluster is embedded in a sparse one. The lengths of the edges of the minimal spanning tree exhibit two distinct clusters which 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 can not be disposed of this easily, the flexibility of the graph-theoretic approach suggests that it is applicable to a wide variety of clustering problems.

11.12 The Problem of Validity

With almost all of the procedures we have 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 sample set, or if we are tracking slowly time-varying patterns. However, it may be an unjustified assumption if we are exploring an essentially unknown set of data. Thus, a constantly 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 $c = 1$, $c = 2$, $c = 3$, etc., 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, since the squared error can be reduced each time c is increased merely by transferring a single sample to the new 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 $c = \hat{c}$, 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 large disparities in the levels at which clusters merge indicate the presence of natural groupings.

A more formal apprach 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, such as a criterion function $J(c)$. Since 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 to advance the *null hypothesis* that there are exactly c clusters present, and 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 sub-of-squared-error criterion.

Suppose that we have a set $\mathcal{H}$ 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 μ and covariance matrix $\sigma^2\mathbf{I}$.* If this hypothesis were true, any 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, since it depends on the particular set of samples:

$$J_e(1) = \sum_{\mathbf{x} \in \mathcal{H}} \|\mathbf{x} \Leftrightarrow \mathbf{m}\|^2, \tag{71}$$

where $\mathbf{m}$ is the mean of the n samples. Under the null hypothesis, the distribution for $J_e(1)$ is approximately normal with mean $nd\sigma^2$ and variance $2nd\sigma^4$.

Suppose now that we partition the set of samples into two subsets $\mathcal{H}_1$ and $\mathcal{H}_2$ so as to minimize $J_e(2)$, where

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

$$J_e(2) = \sum_{i=1}^{2} \sum_{\mathbf{x}\in\mathcal{H}} \|\mathbf{x} \Leftrightarrow \mathbf{m}_i\|^2, \tag{72}$$

$\mathbf{m}_i$ being the mean of the samples in $\mathcal{H}_i$. 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 $n(d \Leftrightarrow 2/\pi)\sigma^2$ and variance $2n(d \Leftrightarrow 8/\pi^2)\sigma^4$ (Problem 23).

This result agrees with out statement that $J_e(2)$ is smaller than $J_e(1)$, since the mean of $J_e(2)$ for the suboptimal partition — $n(d \Leftrightarrow 2/\pi)\sigma^2$ — 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 by

$$\hat{\sigma}^2 = \frac{1}{nd} \sum_{\mathbf{x}\in\mathcal{H}} \|\mathbf{x} \Leftrightarrow \mathbf{m}\|^2 = \frac{1}{nd} J_e(1). \tag{73}$$

The final result can be stated as follows (Problem 26): Reject the null hypothesis at the p-percent significance level if

$$\frac{J_e(2)}{J_e(1)} < 1 \Leftrightarrow \frac{2}{\pi d} \Leftrightarrow \alpha \sqrt{\frac{2(1 \Leftrightarrow 8/\pi^2 d)}{nd}}, \tag{74}$$

where α is determined by

$$p = 100 \int_{\alpha}^{\infty} \frac{1}{2\pi} e^{-u^2/2}\, du = 100(1 \Leftrightarrow erf(\alpha)), \tag{75}$$

where $erf(\cdot)$ 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. ERROR FUNCTION

11.13 Leader-follower clustering

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 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 is present. Our graph theoretic methods can be perfomred on-line — one merely links the new pattrn to an existing cluster baseed on some similarity measure.

In order to make on-line versions of methods suc 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 these centers based soley on its current value and the incoming pattern. Here we shall assume that the number of clusters is known, and return in Sect. **??** to the case where it is not known.

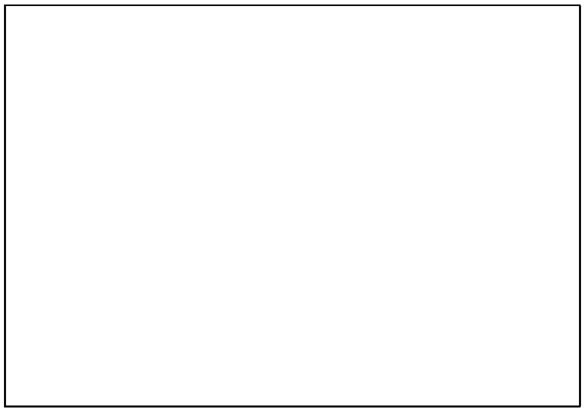

Figure 11.24: Leader-follower clustering adjusts cluster centers on-line, in response to unlabelled patterns.

Suppose we currently have c cluster centers; they may have been placed initially at random positions, or as the first c patterns presented, or 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. 11.24).

If we let $\mathbf{w}_i$ represent the current center for cluster i, our `Basic leader-follower clustering algorithm` is then:

```
                    Basic leader-follower clustering

Initialize $c$ cluster centers, rate $\eta$      initialize
For each input pattern $\mathbf{x}$                  new patterns
  Find nearest center, e.g., $\mathbf{w}_i$          find nearest cluster
  $\mathbf{w}_i = \mathbf{w}_i + \eta\mathbf{x}$                       update nearest center
  normalized weights                      normalized weights
Next pattern                              next pattern
End
```

Before we analyze some drawbacks of such a leader-follower clustering algorithm, let us consider one popular neural technique for achieving it.

11.13.1 Competitive Learning

Competitive learning uses a network structurally quite similar to a two-layer perceptron in order to perform leader-follower clustering. Each of the output neurons represents a different cluster center, and its input weights represent the cluster center, i.e., the pattern that maximally excites the unit.

When a new pattern is presented, each of the output units computes its net activation, $\mathbf{w}^t\mathbf{x}$. Only the most active neuron (i.e., the closest to the new pattern) is permitted to update its weights by the simple formula:*

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta\mathbf{x}, \tag{76}$$

* We note that a winner-take-all networks can be used to insure that only the most active unit will learn.

Figure 11.25: Competitive learning (on-line clustering). All patterns have been normalized $\sum_{i=1}^{d} x_i = 1$, and hence lie on a hypersphere. Likewise, the weights of the three cluster centers have been normalized.

followed by an overall weight normalization ($\sum_{i=0}^{d} w_i = 1$); this normalization is needed to keep the classification based on direction (i.e., position in feature space) rather than overall magnitude of $\mathbf{w}$. Figure 11.25 shows the trajectories of three cluster centers in response to a sequence of patterns chosen randomly from the set shown.

The above algorithm, however, can ocassionally present a problem, regardless of whether it is implemented via competitive learning. Consider a cluster $\mathbf{w}_1$ that originally codes a particular pattern $\mathbf{x}_0$, i.e., if $\mathbf{x}_0$ is presented, the output node having weights $\mathbf{w}_1$ is most activated. Suppose a "hostile" sequence of patterns is presented, i.e., one that sweeps the cluster centers in unusual ways (Fig. 11.26). It is possible that after the cluster centers have been swept, that $\mathbf{x}_0$ is coded by $\mathbf{w}_2$. Indeed, a particularly devious sequence can lead $\mathbf{x}_0$ to be coded by an *arbitrary* sequence of cluster centers, with any cluster center being active an arbitrary number of times.

In short, in a non-stationary environment, a we may want our clustering algorithm to be stable to prevent ceaseless recoding, and yet plastic, or changable, in response to a new pattern. (Freezing cluster centers would prevent recoding, but would not permit learning of new patterns.) This tradeoff has been called the *stability-plasticity* dilemma, and we shall soon see how it can be overcome. First, however, we turn to the problem of unknown number of clusters.

STABILITY-PLASTICITY

11.13.2 Unknown number of clusters

We have assumed that the number of cluster centers is known. If we do not know the number of cluster centers, we need some criterion for the *creation* of new centers. (One can also have cluster deletion, but this is rarely used.) The simplest and most natural method is to create a new cluster center if a pattern being presented is "sufficiently different" from any of the existing clusters.

Figure 11.26: Instability can arise when a pattern is assigned different cluster memberships at different times. Early in clustering the pattern marked $\mathbf{x}^*$ lies in the black cluster, while later in clustering it lies in the red cluster. Similar pattern presentations can make $\mathbf{x}^*$ alternate arbitrarily between clusters.

```
On-line clustering with cluster creation

Initialize cluster centers, $\delta$                initialize
Input pattern $\mathbf{x}(t)$                        step through patterns randomly
Find closest cluster center, e.g., $\mathbf{w}_i$   find closest cluster center
If $d(\mathbf{x}, \mathbf{w}_i) \leq \delta$         close enough?
  Update cluster center                             move cluster center
Else create new center $\mathbf{w}_j = \mathbf{x}(t)$   new center
Next pattern
End
```

11.13.3 Adaptive Resonance

The simplest adaptive resonance networks (or Adaptive Resonance Theory or ART networks) perform a modification of the `On-line clustering with cluster creation` procedure we have just seen. While the primary motivation for ART was to explain biological learning, we shall not be concerned here with their biological relevance nor with their use in *supervised* learning (but see Problem 33).

It is simplest to consider an ART as an elaboration upon competitive learning networks (Fig. 11.27). An ART system takes the competitive learning network and adds top-down modifiable weights. This comprises the *attentional system.* To this is added an *orienting subsystem*, whose function is to detect novelty (i.e., if a new pattern is sufficiently different from an existing cluster).

ATTENTIONAL SYSTEM

ORIENTING SYSTEM

The network works as follows. First a pattern is presented to the input units. This leads via bottom-up connections w_{ij} to activations in the output units. A winner-take-all computation leads to only the most activated ouput unit being active — all other output units are suppressed. Activation is then sent *back* to the input units via weights w_{ji}. This leads, in turn to a modification of the activation of the input units. Very quickly, a stable configuration of output and input units occurs, called a "resonance"(though this has nothing to do with the type of resonance in a driven oscillator).

Figure 11.27: Adaptive Resonance network (ART1 for binary patterns). Weights are bidirectional, gain, the orienting system controls the , and hence (indirectly) the number of clusters found.

ART networks detect novelty by means of the orienting subsystem. The details need not concern us here, but in broad overview, the orienting subsystem has two inputs: the total number of active input features and the total number of features that are active in the input layer. (Note that these two numbers need not be the same, since the top-down feedback affects the activation of the input units, but not the number of active inputs themselves.) If an input pattern is "too different" from any current cluster centers, then the orienting subsystem sends a *reset wave* signal that renders the active output unit quiet. This allows a *new* cluster center to be found, or if all have been explored, then a new cluster center is created.

The criterion for "too different" is a single number, set by the user, called the *vigilance*, $\rho(0 \leq \rho \leq 1$. Denoting the number of active input features as $|I|$ and the number active in the input layer during a resonance as $|R|$, then there will be a reset if

VIGILANCE

$$\frac{|R|}{|I|} < \rho, \tag{77}$$

where rho is a user-set number called the *vigilance parameter*. A low vigilance parmeter means that there can be a poor "match" between the input and the learned cluster and the network will accept it. (Thus vigilance and the ratio of the number of features used by ART, while motivated by *proportional* considerations, is just one of an infinite number of possible closeness criteria (related to δ). For the same data set, a low vigilance leads to a small number of large coarse clusters being formed, while a high vigilance leads to a large number of fine clusters (Fig. 11.28).

VIGILANCE PARAMETER

We have presented the basic approach and issues with ART1, but these return (though in a more subtle way) in analog versions of ART in the literature.

11.14 Low-Dimensional Representations and Multidimensional Scaling

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

Figure 11.28: The results of ART1 applied to a sequence of binary figures. a) $\rho = xx$. b) $\rho = 0.xx$.

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 the 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 $\mathbf{x}_1, ..., \mathbf{x}_n$. Let $\mathbf{y}_i$ be the lower-dimensional *image* of $\mathbf{x}_i$, δ_{ij} be the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$, and d_{ij} be the distance between $\mathbf{y}_i$ and $\mathbf{y}_j$ (Fig. 11.29). Then we are looking for a *configuration* of image points $\mathbf{y}_1, ..., \mathbf{y}_n$ for which the $n(n \Leftrightarrow 1)/2$ distances d_{ij} between image points are as close as possible to the corresponding original distances δ_{ij}. Since 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:

$$J_{ee} = \frac{\sum\limits_{i<j} (d_{ij} \Leftrightarrow \delta_{ij})^2}{\sum\limits_{i<j} \delta_{ij}^2} \tag{78}$$

$$J_{ff} = \sum_{i<j} \left(\frac{d_{ij} \Leftrightarrow \delta_{ij}}{\delta_{ij}} \right)^2 \tag{79}$$

$$J_{ef} = \frac{1}{\sum\limits_{i<j} \delta_{ij}} \sum_{i<j} \frac{(d_{ij} \Leftrightarrow \delta_{ij})^2}{\delta_{ij}}. \tag{80}$$

Since 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 the largest errors (regardless whether the distances δ_{ij}

Figure 11.29: The distance between points in the original space are δ_{ij} while in the projected space d_{ij}.

are large or small), J_{ff} emphasizes the largest fractional errors (regardless whether the errors $|d_{ij} \Leftrightarrow \delta_{ij}|$ are large or small). A useful compromise is J_{ef}, which emphasizes the largest product of error and fractional error.

Once a criterion function has been selected, an optimal configuration $\mathbf{y}_1, ..., \mathbf{y}_n$ is defined as one that minimizes that criterion function. An optimal 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. Since

$$d_{ij} = \|\mathbf{y}_i \Leftrightarrow \mathbf{y}_j\|,$$

the gradient of d_{ij} with respect to $\mathbf{y}_i$ is merely a unit vector in the direction of $\mathbf{y}_i \Leftrightarrow \mathbf{y}_j$. Thus, the gradients of the criterion functions are easy to compute:

$$\begin{aligned}
\nabla_{\mathbf{y}_k} J_{ee} &= \frac{2}{\sum\limits_{i<j} \delta_{ij}^2} \sum_{j \neq k} (d_{kj} \Leftrightarrow \delta_{kj}) \frac{\mathbf{y}_k \Leftrightarrow \mathbf{y}_j}{d_{kj}} \\
\nabla_{\mathbf{y}_k} J_{ff} &= 2 \sum_{j \neq k} \frac{d_{kj} \Leftrightarrow \delta_{kj}}{\delta_{kj}^2} \frac{\mathbf{y}_k \Leftrightarrow \mathbf{y}_j}{d_{kj}} \\
\nabla_{\mathbf{y}_k} J_{ef} &= \frac{2}{\sum\limits_{i<j} \delta_{ij}} \sum_{j \neq k} \frac{d_{kj} \Leftrightarrow \delta_{kj}}{\delta_{kj}} \frac{\mathbf{y}_k \Leftrightarrow \mathbf{y}_j}{d_{kj}}.
\end{aligned}$$

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 thirty points spaced at unit intervals along a three-dimensional spiral cone:

$$x_1(k) = k \cos(x_3)$$

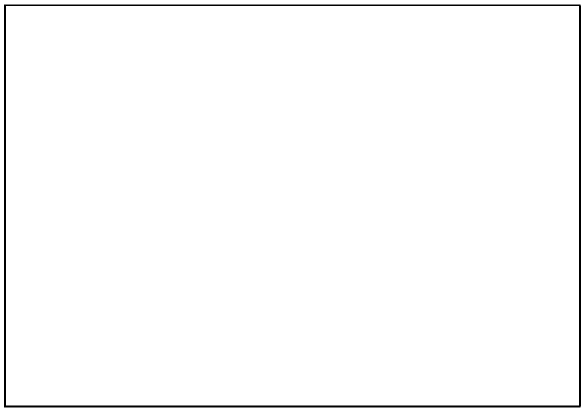

Figure 11.30: A two-dimensional representation of data points in three dimensions.

$$\begin{aligned} x_2(k) &= k\,\sin(x_3) \\ x_3(k) &= k, \qquad k = 0, 1, ..., 29. \end{aligned}$$

Figure 11.30a) shows a perspective representation of the three-dimensional data. When the J_{ef} criterion was used, twenty iterations of a gradient descent procedure produced the two-dimensional configuration shown in Fig. 11.30b). Of course, translations, rotations, and reflections of this configuration would be equally good solutions.

In non-metric 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 $m = n(n \Leftrightarrow 1)/2$ 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*

MONO-TONICITY CONSTRAIN-T

$$\hat{d}_{i_1 j_1} \leq \hat{d}_{i_2 j_2} \leq \cdots \leq \hat{d}_{i_m j_m}. \tag{81}$$

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

$$\hat{J}_{mon} = \min_{\hat{d}_{ij}} \sum_{i<j} (d_{ij} \Leftrightarrow \hat{d}_{ij})^2, \tag{82}$$

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 $\mathbf{y}_1, ..., \mathbf{y}_n$ represents the original data. Unfortunately, $\hat{J}_{mon}$ can not be used to define an optimal configuration because it can be made to vanish by collapsing the configuration to a single point. However, this defect is easily removed by a normalization such as the following:

$$J_{mon} = \frac{\hat{J}_{mon}}{\sum\limits_{i<j} d_{ij}^2}. \tag{83}$$

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

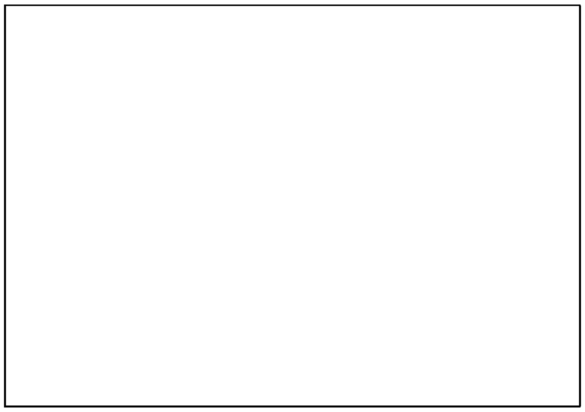

Figure 11.31: Multidimensional scaling of the data in the table. a) A representation in two dimensions. b) A representation in three dimensions.

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 revcovery 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.

A more typical use of multidimensional scaling occurs when one has complicated or noisy similarity information, and one seeks to represent it in a way that will illuminate the similarities and differences among elements. For instance, Table **??** shows perceptual confusions among spoken phonemes, and it is natural to represent two phonemes as categorically "close" if they are highly confused. Figure 11.31 shows the results of a simple multi-dimensional scaling algorithm on the data in the table.

kdjf	kjdf	kjdf	kdjf	kjdf	kjdf
kdjf	0.xxx	0.xxx	0.xxx	0.xxx	0.xxx
kdjf	0.xxx	0.xxx	0.xxx	0.xxx	0.xxx
kdjf	0.xxx	0.xxx	0.xxx	0.xxx	0.xxx
kdjf	0.xxx	0.xxx	0.xxx	0.xxx	0.xxx
kdjf	0.xxx	0.xxx	0.xxx	0.xxx	0.xxx

11.14.1 Self-organizing feature maps

A method closely related to multidimensional scaling is that of self-organizing feature maps (or topologically ordered maps, or Kohonen self-organizing feature maps). The basic goal is to represent feature vectors that are somehow close together in one space as begin close together in a constructed space (of typically lower intrinsic dimension).

Kohonen maps

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 target space, here chosen to be a square. The source space is sensed by a two-joint arm of fixed segment lengths, and thus each point $y_1 \Leftrightarrow y_2$ leads to a pair of angles $\phi_1 \Leftrightarrow \phi_2$

Figure 11.32: Self-organizing feature map a) application and b) network. Here, points will be sampled from the input space uniformly. However, because of the non-linearity of the joint system, the signals along the line $\mathbf{x}$ will not be uniform.

(vector $\boldsymbol{\phi}$), which are related to them by inverse trigonometric functions.

Our goal is this: given only a sequence of $\boldsymbol{\phi}$'s (corresponding to points sampled in the source space), create a mapping from $\boldsymbol{\phi}$ to $\mathbf{x}$ such that points neighboring in the souce space map to points that are neighboring in the target space. Note especially that we will *not* need to know the particular nonlinear mapping from $\mathbf{y}$ to $\boldsymbol{\phi}$. It is this goal of preserving neighborhoods that leads us to call the method "topologically correct mapping."

The network is fully connected, with modifiable weights (Fig. 11.32). When a pattern $\boldsymbol{\phi}$, each node in the target space computes its net activation, $net_k = \sum_i \phi_i w_{ki}$. One of the units is most activated; we call it $\mathbf{x}^*$. The weights to this unit and those in its immediate neighborhood are updated according to:

$$w_{ki}(t+1) \propto w_{ki}(t) + \eta(t)\Lambda(\mathbf{x} \Leftrightarrow \mathbf{x}^*)\boldsymbol{\phi}_i, \tag{84}$$

WINDOW FUNCTION

and then followed by an overall normalization such that $|\mathbf{w}| = 1$ for weights at each target unit. The function $\Lambda(\mathbf{x} \Leftrightarrow \mathbf{x}^*)$ is called the "window function," and is typically 1.0 for $\mathbf{x} = \mathbf{x}^*$ and smaller for large values of $|\mathbf{x} = \mathbf{x}^*|$ (Fig. 11.33). The learning rate $\eta(t)$ typically decreases slowly as patterns are presented to insure that learning will ultimately stop.

The effect of Eq. 84 is that after sufficiently many patterns have been presented, neighboring points in the source space lead to neighboring points in the target space, as can be seen in Fig. **??** — we have learned a topologically correct map. Note in particular that we have no direct access to the source space.

Equation 84 has a particularly straightforward interpretation. For each pattern presentation, the "winning" unit in the target space ($\mathbf{w}^*$) is adjusted so that it is more like the particular pattern. Others in the neighborhood of $\mathbf{w}^*$ are also adjusted so that their weights more nearly match that of the input pattern (though not quite as much as for $\mathbf{w}^*$, according to the window function). In this way, neighboring points in the input space lead to neighboring points being active. Figure 11.34 shows an example.

There are inherent ambiguities in the mapping, but these are generally irrelevant. For instance, even if all learning goes well, a mapping from a square to a square could

Figure 11.33: Typical window functions for self-organizing maps for target spaces in a) two dimensions and b) one dimension.

Figure 11.34: The source space and the square grid of the output space. Each point in the circle leads to a particular unit in the target space being active. a) Initial random weights. b) After 1000 pattern presentations. c) 10,000 presentations. d) 100,000 presentations.

Figure 11.35: The circular source space and the *line* of the output space. a) Initial random weights. b) After 1000 pattern presentations. c) 10,000 presentations. d) 100,000 presentations.

Figure 11.36: Some initial (random) weights and the particular sequence of patterns (randomly chosen) will lead to kinks in the mapping.

8 possible orientations (for each of the 90^o rotations and flips). But this ambiguity can lead to a more significant drawback — "kinks" in the map. A particular initial condition can lead to *part* of the map choosing one of the conficurations, while a different part chooses another one (Fig. 11.36). When this occurs, it is generally best to restart the learning from the beginning, with perhaps a wider window function or slower (which tends to increase the contribution of *global* constraints).

Another point is the number of dimensions in the target space. One typically chooses this dimension (and geometry) based on knowledge of the problem or on the use to which the system will be put. For instance, in forming a self-organized map from sounds for vowel recognition, a two-dimensional target space would be appropriate, since it is known that two dimensions suffice.

Such self-organizing feature maps can be used in a number of systems. For instance, one can take a fairly large number (e.g., 12) of temporal frequency filter outputs and use their output to map 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, e.g., /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.

11.15 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 analysis* and *factor analysis*, both of which reduce dimensionality by forming linear combinations of the features. The object of principal components analysis (known in communication theory as the Karhunen-Loéve expansion) 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.

PRINCIPAL COMPONENTS

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

$$\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}. \tag{85}$$

Since $0 \leq \rho_{ij}^2 \leq 1$, 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:

```
Hierarchical dimensionality reduction

Initialize $\hat{d} = d, \mathcal{F}_i = \{x_i\}, i = 1, \ldots, d$      initialize
Do If $\hat{d} = d'$ stop
  ElseCompute corr                                  Compute corr
  Find most correlated pair of                      ???
    distinct clusters, e.g., $\mathcal{F}_i$ and $\mathcal{F}_j$          ???
  Append $\mathcal{F}_j$ to $\mathcal{F}_i$ and delete $\mathcal{F}_j$
  Decrement                                         $\hat{d}$ by 1 decrement
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. 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, we are interested in something more like the method of multiple discriminant analysis described in Sect. **??**.

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 by 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 (or very complicated) problems. Various ways of circumventing this problem that may be useful under the proper circumstances can be found in the literature, and we have included a few entry points to this 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 or clustering consists of extracting information from unlabelled samples. If the underlying distribution comes from a mixture model of known number of components, and if we need only find parameters $\boldsymbol{\theta}$ for them, we can use Bayesian or maximum likelihood methods. Since there are only ocassionally analytic solutions to these problems, a number of greedy (*locally* step-wise optimal) iterative algorithms can be used, such as K-means clustering.

If the problem represents a natural hierarchy of subclusters, hierarchical methods are needed; the resulting cluster structure is revealed in a dendrogram. On-line clustering leads to a different set of compromises and leader-follower clusterers (which adjust cluster centers based on their current value and the pattern presented), and one particular version, Adaptive Resonance.

Unsupervised methods for feature selection and representation, such as multidimensional scaling and self-organizing feature maps, lead to representations that might illuminate data structure. If augmented by a small amount of supervised learning, the results can be useful in pattern classification.

Bibliographical and Historical Remarks

Books/Monographs

XX & Jain (19??)

Papers

Kohonen(19??), Grossberg (19??), Moore (19??), Stork (19??)

Problems

⊕ *Section 11.1*

⊕ *Section 11.2*

⊕ *Section 11.3*

⊕ *Section 11.4*

⊕ *Section 11.5*

⊕ *Section 11.6*

⊕ *Section 11.7*

⊕ *Section 11.8*

⊕ *Section 11.9*

⊕ *Section* **??**

⊕ *Section 11.10*

⊕ *Section 11.11*

⊕ *Section 11.12*

⊕ *Section 11.14*

⊕ *Section 11.14.1*

⊕ *Section 11.15*

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

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

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

(b) Is it *completely* unidentifiable?

(c) How does this answer change if the a priori probabilities are also unknown?

2. Let $\mathbf{x}$ be a binary vector and $P(\mathbf{x}|\boldsymbol{\theta})$ 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

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

(a) Show that

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

(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

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

(c) Interpret your answer in words.

3. 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 Sect. **??**, show that the maximum likelihood estimate for σ_i^2 must satisfy

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

where $\hat{\boldsymbol{\mu}}_i$ and $\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}_i)$ are given by Eqs. 17 & 19, respectively.

4. 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 a *number* of the component densities. Let l be the n-sample log-likelihood function, and show that

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

where

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

5. 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 $\boldsymbol{\Sigma}$, σ^{pq} be the pqth element of $\boldsymbol{\Sigma}^{-1}$, $x_p(k)$ be the pth element of $\mathbf{x}_k$, and $\mu_p(i)$ be the pth element of $\boldsymbol{\mu}_i$.

(a) Show that

$$\frac{\partial \log\, p(\mathbf{x}_k|\omega_i,\boldsymbol{\theta}_i)}{\partial \sigma^{pq}} = \Big(1 \Leftrightarrow \frac{\delta_{pq}}{2}\Big)\big[\sigma_{pq} \Leftrightarrow (x_p(k) \Leftrightarrow \mu_p(i))(x_q(k) \Leftrightarrow \mu_q(i))\big].$$

(b) Use this result and the results of Problem 4 to show that the maximum likelihood estimate for $\boldsymbol{\Sigma}$ must satisfy

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

where $\hat{P}(\omega_i)$ and $\hat{\boldsymbol{\mu}}_i$ are the maximum likelihood estimates given by Eqs. 16 & 17.

6. Show that the maximum likelihood estimate of an a priori 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

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

(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 $x_1^2 < \log 2$.

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

7. Consider the univariate normal mixture

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

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

$$\max_{\mu_1,...,\mu_c}\Big\{\frac{1}{n}\log\, p(x_1,...,x_n|\mu_1,...,\mu_c)\Big\}$$

ought to be approximately

$$\sum_{j=1}^{c} P(\omega_j)\log\, P(\omega_j) \Leftrightarrow \frac{1}{2}\log\, 2\pi\sigma e$$

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

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

$$p(\theta_1|x_1)p_2(\theta_2|x_1)$$

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

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

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

10. Let $s(\mathbf{x}, \mathbf{x}') = \mathbf{x}^t\mathbf{x}'/(\|\mathbf{x}\| \cdot \|\mathbf{x}'\|)$.

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

(b) Show that for this case

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

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

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

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

12. Consider a set of $n = 2k + 1$ samples, k of which coincide at $x = \Leftrightarrow 2$, 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 $a^2 > 2(k + 1)$?

13. Let $\mathbf{x}_1 = (4, 5)^t$, $\mathbf{x}_2 = (1, 4)^t$, $\mathbf{x}_3 = (0, 1)^t$, and $\mathbf{x}_4 = (5, 0)^t$, and consider the following three partitions:

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

Show that by the sum-of-square error (or $tr\mathbf{S}_W$) criterion, the third partition is favored, whereas by the invariant $|\mathbf{S}_W|$ criterion the first two partitions are favored.

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

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

(b) Show that the eigenvalues $\nu_1, ..., \nu_d$ 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?

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

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

where $\mathbf{m}_i$ is the mean of the n_i samples in $\mathcal{H}_i$ 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 $\mathcal{H}_i$ to $\mathcal{H}_j$ causes J_T to change to

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

(c) Using this results, write pseudocode for an iterative procedure for minimizing J_T (cf. Computer Exercise 5).

16. Let d be the dimensionality of the space, q a parameter and $\boldsymbol{\alpha}$ a vector of parameters. For each of the measures shown, state whether it represents a metric (or not), and whether it represents an ultrametric (or not).

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

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

(c) $s(\mathbf{x}, \mathbf{x}') = \left(\sum\limits_{k=1}^{d} |x_k \Leftrightarrow x'_k|^q \right)^{1/q}$ (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)

(f) $s(\mathbf{x}, \mathbf{x}') = max_{\boldsymbol{\alpha}}\|\mathbf{x} + \boldsymbol{\alpha}\mathbf{T}(\mathbf{x}) \Leftrightarrow \mathbf{x}'\|^2$ (one-sided tangent distance)

17. Use the facts that $\mathbf{S}_T = \mathbf{S}_W + \mathbf{S}_B, J_e = tr\mathbf{S}_W$, and $tr\mathbf{S}_B = \sum n_i\|\mathbf{m}_i \Leftrightarrow \mathbf{m}\|^2$ to derive Eqs. **??** & **??** for the change in J_e resulting from transferring a sample $\hat{\mathbf{x}}$ from cluster $\mathcal{H}_i$ to cluster $\mathcal{H}_j$.

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

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

Show that the following choices for the coefficients $\alpha_i, \alpha_j, \beta$, and γ lead to the distance functions indicated.

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

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

19. 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

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

is minimum.

20. Consider the representation of the points $\mathbf{x}_1 = (1, 0)^t, \mathbf{x}_2 = (0, 0)^t$ and $\mathbf{x}_3 = (0, 1)^t$ by a one-dimensional configuration. 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 $y_3 = 2y_2$.

21. Verify the derivation of Eqs. **??** – 10.
22. Consider the combinatorics of exhaustive inspection of clusters of n samples into c clusters.

(a) Show that there are exactly

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

such distinct clusterings.

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

(c) Find an approximation when $n \gg c$.

23. Show that the sum of squared error for the partition in Sect. **??** or Fig. **??** has mean $n(d \Leftrightarrow 2/\pi)\sigma^2$ and variance $2n(d \Leftrightarrow 8/\pi^2)\sigma^4$.
24. Derive the invariants of Eqs. 56 & 57.
25. Prove that the ranking of distances between samples discussed in Sect. **??** 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 $k \Leftrightarrow 1$. 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, i.e., that $0 = v_1 \leq v_2 \leq v_3 \leq \cdots \leq v_n$.

26. Derive Eqs. 74 & 75.
27. Consider ???

(a) Starting from Eq. 16, derive Eqs. 16 – 18.

(b) Repeat, but assume that the covariance matrices are diagonal.

(c) Repeat, but assume that the covariance matrices are equal.

28. Show how exact solutions in Bayes and Max Likelihood are exponential as follows. Consider [[more here]]
29. Show that the clustering criterion J_d in Eq. 54 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) Since 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.

Since the scale factor $|\mathbf{T}|^2$ is the same for all partitions, it follows 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.

30. Derive Eqs. 62 & 63.

31. Show that the eigenvalues $\lambda_1, \ldots, \lambda_d$ of $\mathbf{S}_W^1\mathbf{S}_B$ (for the within- and between-cluster scatter matrices) are invariant under nonsingular linear transformations of the data.

32. Problem where *some* parameters can b identified, but not all. [[more here]]

33. Show that a two-layer ART network cannot solve the XOR problem.

34. Assume that a mixture density $p(\mathbf{x}|\boldsymbol{\theta})$ is identifiable. Prove that under very general conditions that $p(\boldsymbol{\theta}|\mathcal{H}^n)$ convergest (in probability) to a Dirac delta function centered at the true value of $\boldsymbol{\theta}$.

Computer exercises

Data table for several problems (3 D) do vs 2 D projections...) create dendrograms

sample	x_1	x_2	x_3
1	0.523	0.59	1.26
2	0.517	0.519	0.921
3	0.680	0.276	1.26
4	0.558	0.597	1.19
5	0.514	0.251	1.36
6	0.604	0.783	1.12
7	0.697	0.755	1.28
8	0.629	0.285	1.36
9	0.59	0.213	1.26
10	0.515	0.212	1.31

sample	x_1	x_2	x_3
11	0.608	0.799	1.340
12	0.575	0.613	0.538
13	0.532	0.371	1.081
14	0.614	0.653	1.284
15	0.634	0.729	0.644
16	0.634	0.383	1.308
17	0.647	0.342	0.784
18	0.675	0.442	0.855
19	0.591	0.317	0.586
20	0.624	0.378	1.145

⊕ *Section 11.1*

⊕ *Section 11.2*

⊕ *Section 11.3*

⊕ *Section 11.4*

⊕ *Section 11.5*

⊕ *Section 11.6*

⊕ *Section 11.7*

⊕ *Section 11.8*

⊕ *Section 11.9*

⊕ *Section* **??**

⊕ *Section 11.10*

⊕ *Section 11.11*

⊕ *Section 11.12*

⊕ *Section 11.14*

⊕ *Section 11.14.1*

⊕ *Section 11.15*

1. Write a program to lkjsdflksdj lkjsdflsdj sdlkfjsd fsd dlfkjs df
2. Consider the univariate normal mixture

$$p(\mathbf{x}|\boldsymbol{\theta}) = \frac{P(\omega_1)}{\sqrt{2\pi}\sigma_1}\text{exp}\ \left[\Leftrightarrow\frac{1}{2}(\frac{x \Leftrightarrow \mu_1}{\sigma_1})^2\right] + \frac{P(\omega_2)}{\sqrt{2\pi}\sigma_2}\text{exp}\ \left[\Leftrightarrow\frac{1}{2}(\frac{x \Leftrightarrow \mu_2}{\sigma_2})^2\right].$$

3. Write a computer program that uses the general maximum likelihood equation of Sect. **??** iteratively to estimate the unknown means, variances, and a priori probabilities. Use this program to find maximum likelihood estimates of these parameters for the data in Table **??**.
4. hill climbing for clustering. Start at BAD and at GOOD starting places. Note that do not get same answer.
5. Write a program to perform the minimization of in Problem 15.
6. Cluster with one metric, then classify with a *different metric* [[more here]]