# Contents

# Chapter 10

# Unsupervised Learning and Clustering

## 10.1 Introduction

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

One might wonder why anyone is interested in such an unpromising problem, and whether or not it is possible even in principle to learn anything of value from unlabeled samples. There are at least five basic reasons for interest in unsupervised procedures. First, collecting and labeling a large set of sample patterns can be surprisingly costly. For instance, recorded speech is virtually free, but accurately *labeling* the speech — marking what word or phoneme is being uttered at each instant — can be very expensive and time consuming. If a classifier can be crudely designed on a small set of labeled samples, and then "tuned up" by allowing it to run without supervision on a large, unlabeled set, much time and trouble can be saved. Second, one might wish to proceed in the reverse direction: train with large amounts of (less expensive) unlabeled data, and only then use supervision to label the groupings found. This may be appropriate for large "data mining" applications where the contents of a large database are not known beforehand. Third, in many applications the characteristics of the patterns can change slowly with time, for example in automated food classification as the seasons change. If these changes can be tracked by a classifier running in an unsupervised mode, improved performance can be achieved. Fourth, we can use unsupervised methods to find *features*, that will then be useful for categorization. There are unsupervised methods that represent a form 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 unlabeled 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 densities are known, and that the only thing that must be learned is the value of an unknown parameter vector. Interestingly enough, the formal solution to this problem will turn out to be almost identical to the solution for the problem of supervised learning given in Chap. **??**. Unfortunately, in the unsupervised case the solution suffers from the usual problems associated with parametric assumptions without providing any of the benefits of computational simplicity. This will lead us to various attempts to reformulate the problem as one of partitioning the data into subgroups or clusters. While some of the resulting clustering procedures have no known significant theoretical properties, they are still among the more useful tools for pattern recognition problems.

## 10.2 Mixture Densities and Identifiability

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

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

Samples are assumed to be obtained by selecting a state of nature $\omega_j$ 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 $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_c)$. 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 prior probabilities $P(\omega_j)$ are called the *mixing parameters*. The mixing parameters can also be included among the unknown parameters, but for the moment we shall assume that only $\boldsymbol{\theta}$ is unknown.

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 and use a Bayesian classifier on the derived densities, if indeed classification is our final goal. Before seeking explicit solutions to this problem, however, let us ask whether or not it is possible in principle to recover $\boldsymbol{\theta}$ from the mixture. Suppose that we had an unlimited number of samples, and that we used one of the nonparametric methods of Chap. **??** to determine the value of $p(\mathbf{x}|\boldsymbol{\theta})$ for every $\mathbf{x}$. If

there is only one value of $\boldsymbol{\theta}$ that will produce the observed values for $p(\mathbf{x}|\boldsymbol{\theta})$, then a solution is at least possible in principle. However, if several different values of $\boldsymbol{\theta}$ can produce the same values for $p(\mathbf{x}|\boldsymbol{\theta})$, then there is no hope of obtaining a unique solution.

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

COMPLETE UNIDENTIFIABILITY

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

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

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 3).

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

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

cannot be uniquely identified if $P(\omega_1) = P(\omega_2)$, for then $\theta_1$ and $\theta_2$ 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 values, but such a guess would have to be biased in some way.

## 10.3 Maximum-Likelihood Estimates

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

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

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

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

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

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

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

and

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

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

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

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

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

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

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

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

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

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

and

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

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

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

and

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

where

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

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

## 10.4 Application to Normal Mixtures

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

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

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

### 10.4.1 Case 1: Unknown Mean Vectors

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

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

its derivative is

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

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

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

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

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

This equation is intuitively very satisfying. It shows that the maximum-likelihood estimate for $\boldsymbol{\mu}_i$ is merely a weighted average of the samples; the weight for the $k$th sample is an estimate of how likely it is that $\mathbf{x}_k$ belongs to the $i$th 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 $i$th class. More generally, suppose that $\hat{\boldsymbol{\mu}}_i$ is sufficiently close to the true value of $\boldsymbol{\mu}_i$ that $P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}})$ is essentially the true posterior probability for $\omega_i$. 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 $i$th class, then we see that Eq. 17 essentially gives $\hat{\boldsymbol{\mu}}_i$ as the average of the samples coming from the $i$th class.

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

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

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

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

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

This is basically a gradient ascent or hill-climbing procedure for maximizing the log-likelihood function. If the overlap between component densities is small, then the

coupling between classes will be small and convergence will be fast. However, when convergence does occur, all that we can be sure of is that the gradient is zero. Like all hill-climbing procedures, this one carries no guarantee of yielding the global maximum (Computer exercise 19). Note too that if the model is mis-specified (for instance we assume the "wrong" number of clusters) then the log-likelihood can actually decrease (Computer exercise 21).

## 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:

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

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

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

for various values of $\mu_1$ and $\mu_2$. The bottom figure shows how $l$ varies with $\mu_1$ and $\mu_2$. The maximum value of $l$ occurs at $\hat{\mu}_1 = -2.130$ and $\hat{\mu}_2 = 1.668$, which is in the rough vicinity of the true values $\mu_1 = -2$ and $\mu_2 = 2$. However, $l$ reaches another peak of comparable height at $\hat{\mu}_1 = 2.085$ and $\hat{\mu}_2 = -1.257$. Roughly speaking, this solution corresponds to interchanging $\mu_1$ and $\mu_2$. Note that had the prior probabilities been equal, interchanging $\mu_1$ and $\mu_2$ 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.

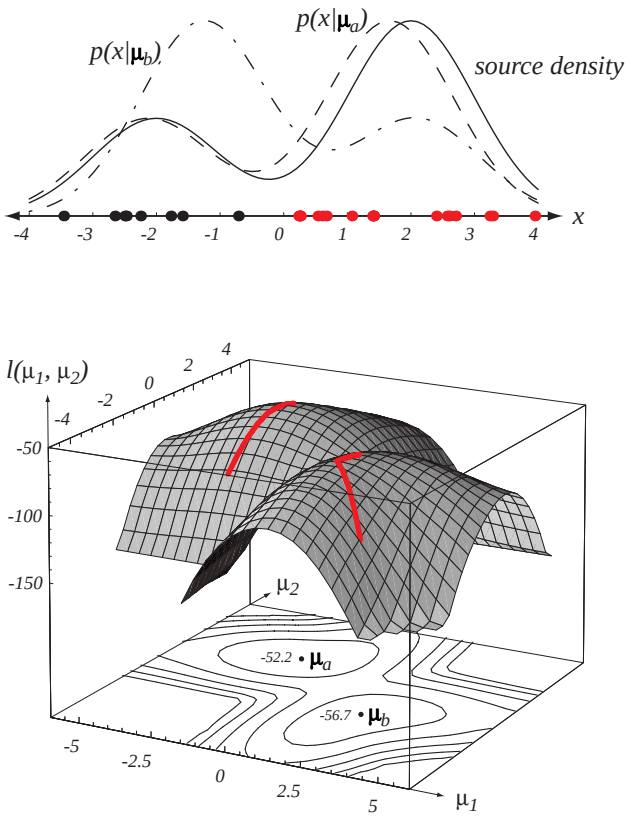
| $k$ | $x_k$ | $\omega_1$ | $\omega_2$ |
|---|---|---|---|
| 1 | 0.608 | | × |
| 2 | -1.590 | × | |
| 3 | 0.235 | | × |
| 4 | 3.949 | | × |
| 5 | -2.249 | × | |
| 6 | 2.704 | | × |
| 7 | -2.473 | × | |
| 8 | 0.672 | | × |

| $k$ | $x_k$ | $\omega_1$ | $\omega_2$ |
|---|---|---|---|
| 9 | 0.262 | | × |
| 10 | 1.072 | | × |
| 11 | -1.773 | × | |
| 12 | 0.537 | | × |
| 13 | 3.240 | | × |
| 14 | 2.400 | | × |
| 15 | -2.499 | × | |
| 16 | 2.608 | | × |

| $k$ | $x_k$ | $\omega_1$ | $\omega_2$ |
|---|---|---|---|
| 17 | -3.458 | × | |
| 18 | 0.257 | | × |
| 19 | 2.569 | | × |
| 20 | 1.415 | | × |
| 21 | 1.410 | | × |
| 22 | -2.653 | × | |
| 23 | 1.396 | | × |
| 24 | 3.286 | | × |
| 25 | -0.712 | × | |

Additional insight into the nature of these multiple solutions can be obtained by examining the resulting estimates for the mixture density. The figure at the top shows the true (source) mixture density and the estimates obtained by using the two maximum-likelihood estimates as if they were the true parameter values. The 25 sample values are shown as a scatter of points along the abscissa — $\omega_1$ points in

black, $\omega_2$ points in red. Note that the peaks of both the true mixture density and the maximum-likelihood solutions are located so as to encompass two major groups of data points. The estimate corresponding to the smaller local maximum of the log-likelihood function has a mirror-image shape, but its peaks also encompass reasonable groups of data points. To the eye, neither of these solutions is clearly superior, and both are interesting.




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

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

answer.

### 10.4.2 Case 2: All Parameters Unknown

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

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

The likelihood function for $n$ samples drawn from this probability density is merely the product of the $n$ densities $p(x_k|\mu, \sigma^2)$. Suppose that we let $\mu = x_1$, the value of the first sample. In this situation the density is

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

Clearly, for the rest of the samples

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

so that

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

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

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

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

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

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

and

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

where $\delta_{pq}$ is the Kronecker delta. We substitute these results in Eq. 12 and perform a small amount of algebraic manipulation (Problem 16) and thereby obtain the following equations for the local-maximum-likelihood estimate $\hat{\boldsymbol{\mu}}_i$, $\hat{\boldsymbol{\Sigma}}_i$, and $\hat{P}(\omega_i)$:

$$\hat{P}(\omega_i) = \frac{1}{n}\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) \tag{19}$$

$$\hat{\boldsymbol{\mu}}_i = \frac{\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})\mathbf{x}_k}{\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})} \tag{20}$$

$$\hat{\boldsymbol{\Sigma}}_i = \frac{\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})(\mathbf{x}_k - \boldsymbol{\mu}_i)(\mathbf{x}_k - \boldsymbol{\mu}_i)^t}{\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})} \tag{21}$$

where

$$\begin{aligned}\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) &= \frac{p(\mathbf{x}_k|\omega_i, \hat{\boldsymbol{\theta}}_i)\hat{P}(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}_k|\omega_j, \hat{\boldsymbol{\theta}}_j)\hat{P}(\omega_j)} \\ &= \frac{|\hat{\boldsymbol{\Sigma}}_i|^{-1/2}\, \exp\big[-\frac{1}{2}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)^t \hat{\boldsymbol{\Sigma}}_i^{-1}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)\big]\hat{P}(\omega_i)}{\sum_{j=1}^{c} |\hat{\boldsymbol{\Sigma}}_j|^{-1/2}\, \exp\big[-\frac{1}{2}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_j)^t \hat{\boldsymbol{\Sigma}}_j^{-1}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_j)\big]\hat{P}(\omega_j)}.\end{aligned} \tag{22}$$

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 $\omega_i$ and 0.0 otherwise, $\hat{P}(\omega_i)$ is the fraction of samples from $\omega_i$, $\hat{\boldsymbol{\mu}}_i$ 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. 22 for $\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})$ and then

to use Eqs. 19 – 21 to update these estimates. If the initial estimates are very good, having perhaps been obtained from a fairly large set of labeled samples, convergence can be quite rapid. However, the results do depend upon the starting point, and the problem of multiple solutions is always present. Furthermore, the repeated computation and inversion of the sample covariance matrices can be quite time consuming.

Considerable simplification can be obtained if it is possible to assume that the covariance matrices are diagonal. This has the added virtue of reducing the number of unknown parameters, which is very important when the number of samples is not large. If this assumption is too strong, it still may be possible to obtain some simplification by assuming that the $c$ covariance matrices are equal, which also may eliminate the problem of singular solutions (Problem 16).

### 10.4.3 K-means clustering

Of the various techniques that can be used to simplify the computation and accelerate convergence, we shall briefly consider one elementary, approximate method. From Eq. 22, it is clear that the probability $\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})$ is large when the squared Mahalanobis distance $(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)^t \hat{\boldsymbol{\Sigma}}_i^{-1}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)$ is small. Suppose that we merely compute the squared Euclidean distance $\|\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i\|^2$, find the mean $\hat{\boldsymbol{\mu}}_m$ nearest to $\mathbf{x}_k$, and approximate $\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})$ as

$$\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) \approx \begin{cases} 1 & \text{if } i = m \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

Then the iterative application of Eq. 20 leads to the following procedure for finding $\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_c$. (Although the algorithm is historically referred to as $k$-means clustering, we retain the notation $c$, our symbol for the number of clusters.)

**Algorithm 1 (K-means clustering)**

```
1 begin initialize n, c, μ_1, μ_2, ..., μ_c
2         do classify n samples according to nearest μ_i
3            recompute μ_i
4         until no change in μ_i
5   return μ_1, μ_2, ..., μ_c
6 end
```

The computational complexity of the algorithm is $O(ndcT)$ where $d$ the number of features and $T$ the number of iterations (Problem 15). In practice, the number of iterations is generally much less than the number of samples.

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

It is interesting to see how this procedure behaves on the example data we saw in Example 1. Figure 10.1 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 $\hat{\mu}_1 = -2.176, \hat{\mu}_2 = 1.684$ or to its symmetric

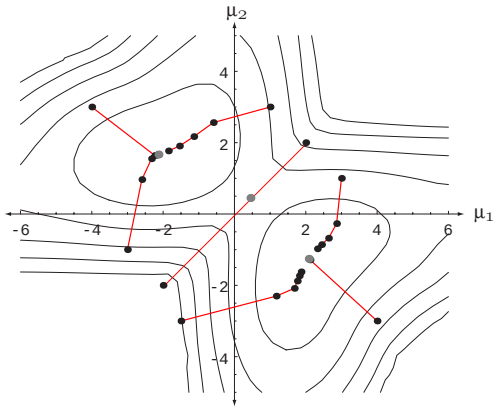



Figure 10.1: The k-means clustering procedure is a form of stochastic hill climbing in the log-likelihood function. The contours represent equal log-likelihood values for the one-dimensional data in Example 1. The dots indicate parameter values after different iterations of the k-means algorithm. Six of the starting points shown lead to local maxima, whereas two (i.e., $\mu_1(0) = \mu_2(0)$) lead to a saddle point near $\boldsymbol{\mu} = \mathbf{0}$.

image. This is close to the solution found by the maximum-likelihood method (viz., $\hat{\mu}_1 = -2.130$ and $\hat{\mu}_2 = 1.688$), and the trajectories show a general resemblance to those shown in Example 1. In general, when the overlap between the component densities is small the maximum-likelihood approach and the k-means procedure can be expected to give similar results.

Figure 10.2 shows a two-dimensional example, with the assumption of $c = 3$ clusters. The three initial cluster centers, chosen randomly from the training points, and their associated Voronoi tesselation, are shown in pink. According to the algorithm, the points in each of the three Voronoi cells are used to calculate new cluster centers (dark pink), and so on. Here, after the third iteration the algorithm has converged (red). Because the k-means algorithm is very simple and works well in practice, it is a staple of clustering methods.

### 10.4.4 *Fuzzy k-means clustering

In every iteration of the classical k-means procedure, each data point is assumed to be in exactly one cluster, as implied by Eq. 23 and by lines 2 & 3 of Algorithm 1. We can relax this condition and assume that each sample $\mathbf{x}_j$ has some graded or "fuzzy" cluster membership $\mu_i(\mathbf{x}_j)$ in cluster $\omega_i$, where $0 \leq \mu_i(\mathbf{x}_j) \leq 1$. At root, these "memberships" are equivalent to the probabilities $\hat{P}(\omega_i|\mathbf{x}_j, \hat{\theta})$ given in Eq. 22, and thus we use this symbol. In the resulting fuzzy k-means clustering algorithm we seek a minimum of a global cost function

$$L = \sum_{i=1}^{c} \sum_{j=1}^{n} [\hat{P}(\omega_i|\mathbf{x}_j, \hat{\theta})]^b ||\mathbf{x}_j - \boldsymbol{\mu}_i||^2, \tag{24}$$

where $b > 1$ is a free parameter chosen to adjust the "blending" of different clusters.

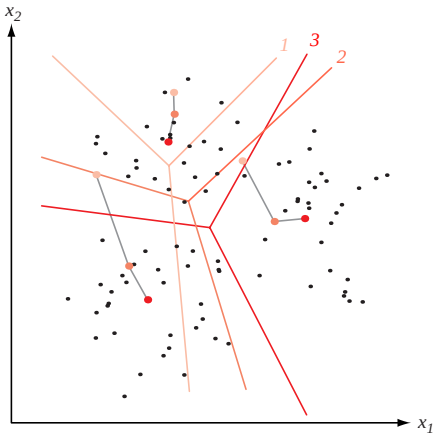



Figure 10.2: Trajectories for the means of the k-means clustering procedure applied to two-dimensional data. The final Voronoi tesselation (for classification) is also shown — the means correspond to the "centers" of the Voronoi cells.

If $b$ is set to 0, this criterion function is merely a sum-of-squared errors criterion we shall see again in Eq. 49. The probabilities of cluster membership for each point are normalized as

$$\sum_{i=1}^{c} \hat{P}(\omega_i|\mathbf{x}_j) = 1, \qquad j = 1, \ldots, n. \tag{25}$$

At the solution, i.e., the minimum of $L$, we have

$$\partial L/\partial \boldsymbol{\mu}_i = 0 \quad \text{and} \quad \partial L/\partial \hat{P}_j = 0, \tag{26}$$

and these lead (Problem 14) to the conditions

$$\boldsymbol{\mu}_j = \frac{\sum_{j=1}^{n} [P(\omega_i|\mathbf{x}_j)]^b \mathbf{x}_j}{\sum_{j=1}^{n} [P(\omega_i|\mathbf{x}_j)]^b} \tag{27}$$

and

$$P(\omega_i|\mathbf{x}_j) = \frac{(1/d_{ij})^{1/(b-1)}}{\sum_{r=1}^{c} (1/d_{rj})^{1/(b-1)}}, \qquad d_{ij} = ||\mathbf{x}_j - \boldsymbol{\mu}_i||^2. \tag{28}$$

In general, the criterion is minimized when the cluster centers $\boldsymbol{\mu}_j$ are near those points that have high estimated probability of being in cluster $j$. Since Eqs 27 & 28 rarely have analytic solutions, the cluster means and point probabilities are estimated iteratively according to the following algorithm:

**Algorithm 2 (Fuzzy k-means clustering)**

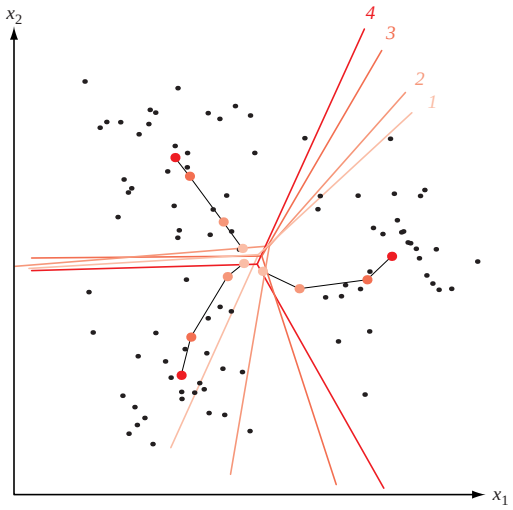



Figure 10.3: At each iteration of the fuzzy k-means clustering algorithm, the probability of category memberships for each point are adjusted according to Eqs. 27 & 28 (here $b = 2$). While most points have non-negligible memberships in two or three clusters, we nevertheless draw the boundary of a Voronoi tesselation to illustrate the progress of the algorithm. After four iterations, the algorithm has converged and the red cluster centers and associated Voronoi tesselation would be used for assigning new points to clusters.

1 **begin** **initialize** $n, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_c, P(\omega_i \mid \mathbf{x}_j), i = 1 \ldots, c;\ j = 1, \ldots, n$
2 normalize proabilities of cluster memberships by Eq. 25
3 **do** classify $n$ samples according to nearest $\boldsymbol{\mu}_i$
4 recompute $\boldsymbol{\mu}_i$ by Eq. 27
5 recompute $P(\omega_i \mid \mathbf{x}_j)$ by Eq. 28
6 **until** no change in $\boldsymbol{\mu}_i$ and $P(\omega_i \mid \mathbf{x}_j)$
7 **return** $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_c$
8 **end**

Figure 10.3 illustrates the algorithm. At early iterations the means lie near the center of the full data set because each point has a non-negligible "membership" (i.e., probability) in each cluster. At later iterations the means separate and each membership tends toward the value 1.0 or 0.0. Clearly, the classical k-means algorithm is just of special case where the memberships for all points obey

$$P(\omega_i|\mathbf{x}_j) = \begin{cases} 1 & \text{if } \|\mathbf{x}_j - \boldsymbol{\mu}_i\| < \|\mathbf{x}_j - \boldsymbol{\mu}_{i'}\| \text{ for all } i' \neq i \\ 0 & \text{otherwise,} \end{cases} \tag{29}$$

as given by Eq. 17.

While it may seem that such graded membership might improve convergence of k-means over its classical counterpart, in practice there are several drawbacks to the fuzzy method. One is that according to Eq. 25 the probability of "membership" of a point $\mathbf{x}_j$ in a cluster $i$ depends implicitly on the number of clusters, and when the number of clusters is specified incorrectly, serious problems may arise (Computer exercise 4).

## 10.5 Unsupervised Bayesian Learning

### 10.5.1 The Bayes Classifier

As we saw in Chap. **??**, maximum-likelihood methods do not assume the parameter vector $\boldsymbol{\theta}$ to be random — it is just unknown. In such methods, prior knowledge about the likely values for $\boldsymbol{\theta}$ is not directly relevant, although in practice such knowledge may be used in choosing good starting points for hill-climbing procedures. In this section, however, we shall take a Bayesian approach to unsupervised learning. That is, we shall assume that $\boldsymbol{\theta}$ is a random variable with a known prior distribution $p(\boldsymbol{\theta})$, and we shall use the samples to compute the posterior density $p(\boldsymbol{\theta}|\mathcal{D})$. 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 prior probabilities $P(\omega_j)$ for each class are known, $j = 1, \ldots, c$.
3. The forms for the class-conditional probability densities $p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)$ are known, $j = 1, \ldots, c$, but the full parameter vector $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_c)$ is not known.
4. Part of our knowledge about $\boldsymbol{\theta}$ is contained in a known prior density $p(\boldsymbol{\theta})$.
5. The rest of our knowledge about $\boldsymbol{\theta}$ is contained in a set $\mathcal{D}$ of $n$ samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ drawn independently from the familiar mixture density

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{c} p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j) P(\omega_j). \tag{30}$$

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

$$P(\omega_i|\mathbf{x}, \mathcal{D}) = \frac{p(\mathbf{x}|\omega_i, \mathcal{D}) P(\omega_i|\mathcal{D})}{\sum_{j=1}^{c} p(\mathbf{x}|\omega_j, \mathcal{D}) P(\omega_j|\mathcal{D})}. \tag{31}$$

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

$$P(\omega_i|\mathbf{x}, \mathcal{D}) = \frac{p(\mathbf{x}|\omega_i, \mathcal{D}) P(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}|\omega_j, \mathcal{D}) P(\omega_j)}. \tag{32}$$

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

$$\begin{aligned} p(\mathbf{x}|\omega_i, \mathcal{D}) &= \int p(\mathbf{x}, \boldsymbol{\theta}|\omega_i, \mathcal{D})\ d\boldsymbol{\theta} \\ &= \int p(\mathbf{x}|\boldsymbol{\theta}, \omega_i, \mathcal{D}) p(\boldsymbol{\theta}|\omega_i, \mathcal{D})\ d\boldsymbol{\theta}. \end{aligned} \tag{33}$$

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

$$P(\mathbf{x}|\omega_i, \mathcal{D}) = \int p(\mathbf{x}|\omega_i, \boldsymbol{\theta}_i) p(\boldsymbol{\theta}|\mathcal{D})\ d\boldsymbol{\theta}. \tag{34}$$

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

### 10.5.2 Learning the Parameter Vector

We can use Bayes' formula to write

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta})}{\int p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta})\ d\boldsymbol{\theta}}, \tag{35}$$

where the independence of the samples yields the likelihood

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{k=1}^{n} p(\mathbf{x}_k|\boldsymbol{\theta}). \tag{36}$$

Alternatively, letting $\mathcal{D}^n$ denote the set of $n$ samples, we can write Eq. 35 in the recursive form

$$p(\boldsymbol{\theta}|\mathcal{D}^n) = \frac{p(\mathbf{x}_n|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}^{n-1})}{\int p(\mathbf{x}_n|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}^{n-1})\ d\boldsymbol{\theta}}. \tag{37}$$

These are the basic equations for unsupervised Bayesian learning. Equation 35 emphasizes the relation between the Bayesian and the maximum-likelihood solutions. If $p(\boldsymbol{\theta})$ is essentially uniform over the region where $p(\mathcal{D}|\boldsymbol{\theta})$ peaks, then $p(\boldsymbol{\theta}|\mathcal{D})$ peaks at the same place. If the only significant peak occurs at $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$, and if the peak is very sharp, then Eqs. 32 & 34 yield

$$p(\mathbf{x}|\omega_i, \mathcal{D}) \approx p(\mathbf{x}|\omega_i, \hat{\boldsymbol{\theta}}) \tag{38}$$

and

$$P(\omega_i|\mathbf{x}, \mathcal{D}) \approx \frac{p(\mathbf{x}|\omega_i, \hat{\boldsymbol{\theta}}_i) P(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}|\omega_j, \hat{\boldsymbol{\theta}}_j) P(\omega_j)}. \tag{39}$$

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). While many *small* sample size

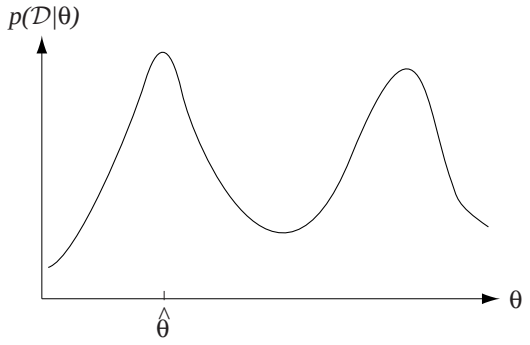



Figure 10.4: In a highly skewed or multiple peak posterior distribution such as illustrated here, the maximum-likelihood solution $\hat{\theta}$ will yield a density very different from a Bayesian solution, which requires the integration over the full range of parameter space $\theta$.

problems they will agree, there exist small problems where the approximations are poor (Fig. 10.4). As we saw in the analogous case in supervised learning whether one chooses to use the maximum-likelihood or the Bayes method depends not only on how confident one is of the prior distributions, but also on computational considerations; maximum-likelihood techniques are often easier to implement than Bayesian ones.

Of course, if $p(\boldsymbol{\theta})$ has been obtained by supervised learning using a large set of labeled samples, it will be far from uniform, and it will have a dominant influence on $p(\boldsymbol{\theta}|\mathcal{D}^n)$ when $n$ is small. Equation 37 shows how the observation of an additional unlabeled 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 $p(\boldsymbol{\theta}|\mathcal{D}^n)$, and under fairly general conditions $p(\boldsymbol{\theta}|\mathcal{D}^n)$ can be shown to converge (in probability) to a Dirac delta function centered at the true value of $\boldsymbol{\theta}$ (Problem 8). Thus, even though we do not know the categories of the samples, identifiability assures us that we can learn the unknown parameter vector $\boldsymbol{\theta}$, and thereby learn the component densities $p(\mathbf{x}|\omega_i, \boldsymbol{\theta})$.

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

There are, however, some significant differences between supervised and unsupervised learning. One of the major differences concerns the issue of identifiability. With supervised learning, the lack of identifiability merely means that instead of obtaining a unique parameter vector we obtain an equivalence class of parameter vectors. Because all of these yield the same component density, lack of identifiability presents no theoretical difficulty. A lack of identifiability is much more serious in unsupervised learning. When $\boldsymbol{\theta}$ cannot be determined uniquely, the mixture cannot be decomposed into its true components. Thus, while $p(\mathbf{x}|\mathcal{D}^n)$ may still converge to $p(\mathbf{x})$, $p(\mathbf{x}|\omega_i, \mathcal{D}^n)$ given by Eq. 34 will not in general converge to $p(\mathbf{x}|\omega_i)$, and a theoretical barrier to learning exists. It is here that a few *labeled* training samples would be valuable: for "decomposing" the mixture into its components.

Another serious problem for unsupervised learning is computational complexity. With supervised learning, the possibility of finding sufficient statistics allows solutions

that are analytically pleasing and computationally feasible. With unsupervised learning, there is no way to avoid the fact that the samples are obtained from a mixture density,

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

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

$$p(\mathcal{D}|\boldsymbol{\theta}) = g(\mathbf{s}, \boldsymbol{\theta}) h(\mathcal{D}). \tag{40}$$

But from Eqs. 36 & 1, we see that the likelihood can be written as

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{k=1}^{n} \Big[ \sum_{j=1}^{c} p(\mathbf{x}_k|\omega_j, \boldsymbol{\theta}_j) P(\omega_j) \Big]. \tag{41}$$

Thus, $p(\mathcal{D}|\boldsymbol{\theta})$ is the sum of $c^n$ products of component densities. Each term in this sum can be interpreted as the joint probability of obtaining the samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ bearing a particular labeling, with the sum extending over all of the ways that the samples could be labeled. Clearly, this results in a thorough mixture of $\boldsymbol{\theta}$ and the $\mathbf{x}$'s, and no simple factoring should be expected. An exception to this statement arises if the component densities do not overlap, so that as $\boldsymbol{\theta}$ varies only one term in the mixture density is non-zero. In that case, $p(\mathcal{D}|\boldsymbol{\theta})$ 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. 37 and obtain

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

If we consider the special case where $P(\omega_1) = 1$ and all the other prior probabilities are zero, corresponding to the supervised case in which all samples come from Class $\omega_1$, then Eq. 42 simplifies to

$$p(\boldsymbol{\theta}|\mathcal{D}^n) = \frac{p(\mathbf{x}_n|\omega_1, \boldsymbol{\theta}_1)}{\int p(\mathbf{x}_n|\omega_1, \boldsymbol{\theta}_1) p(\boldsymbol{\theta}|\mathcal{D}^{n-1}) \, d\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D}^{n-1}). \tag{43}$$

Let us compare Eqs. 42 & 43 to see how observing an additional sample changes our estimate of $\boldsymbol{\theta}$. In each case we can ignore the normalizing denominator, which is independent of $\boldsymbol{\theta}$. Thus, the only significant difference is that in the supervised case we multiply the "prior" density for $\boldsymbol{\theta}$ by the component density $p(\mathbf{x}_n|\omega_1, \boldsymbol{\theta}_1)$, while in the unsupervised case we multiply it by the mixture density $\sum_{j=1}^{c} p(\mathbf{x}_n|\omega_j, \boldsymbol{\theta}_j) P(\omega_j)$. Assuming that the sample really did come from Class $\omega_1$, 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 we have

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

and we seek the mean of the second component.

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

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

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

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

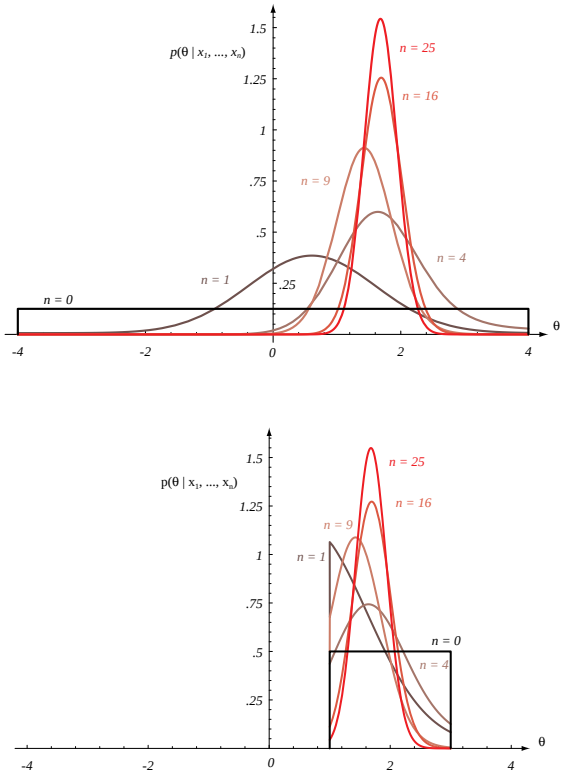
$$\begin{aligned} p(\theta|x_1, x_2) &= \beta p(x_2|\theta)p(\theta|x_1) \\ &= \left\{ \begin{array}{ll} \beta'\{P(\omega_1)P(\omega_1)\exp[-\frac{1}{2}(x_1-\mu)^2 - \frac{1}{2}(x_2-\mu)^2] & \\ \quad +[P(\omega_1)P(\omega_2)\exp[-\frac{1}{2}(x_1-\mu)^2 - \frac{1}{2}(x_2-\theta)^2] & \\ \quad +[P(\omega_2)P(\omega_1)\exp[-\frac{1}{2}(x_1-\theta)^2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

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

It is possible to use the relation

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

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




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

One of the main differences between the Bayesian and the maximum-likelihood approaches to unsupervised learning appears in the presence of the prior density $p(\theta)$. The figure shows how $p(\theta|\mathcal{D}^n)$ changes when $p(\theta)$ is assumed to be uniform from 1 to 3, corresponding to more certain initial knowledge about $\theta$. The results of this change are most pronounced when $n$ is small. It is here (just as in the classification analog of Chap. **??**) that the differences between the Bayesian and the maximum-likelihood

solutions are most significant. As $n$ increases, the importance of prior knowledge diminishes, and in the particular case the curves for $n = 25$ are virtually identical. In general, one would expect the difference to be small when the number of unlabeled samples is several times the effective number of labeled samples used to determine $p(\theta)$.

### 10.5.3 Decision-Directed Approximation

Although the problem of unsupervised learning can be stated as merely the problem of estimating parameters of a mixture density, neither the maximum-likelihood nor the Bayesian approach yields analytically simple results. Exact solutions for even the simplest nontrivial examples lead to computational requirements that grow exponentially with the number of samples (Problem **??**). 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 important difference between supervised and unsupervised learning is the presence or absence of labels for the samples, an obvious approach to unsupervised learning is to use the prior information to design a classifier and to use the decisions of this classifier to label the samples. This is called the *decision-directed* approach to unsupervised learning, and it is subject to many variations. It can be applied sequentially on-line by updating the classifier each time an unlabeled sample is classified. Alternatively, it can be applied in parallel (batch mode) by waiting until all $n$ samples are classified before updating the classifier. If desired, this process can be repeated until no changes occur in the way the samples are labeled. Various heuristics can be introduced to make the extent of any corrections depend upon the confidence of the classification decision.

There are some obvious dangers associated with the decision-directed approach. If the initial classifier is not reasonably good, or if an unfortunate sequence of samples is encountered, the errors in classifying the unlabeled samples can drive the classifier the wrong way, resulting in a solution corresponding roughly to one of the lesser peaks of the likelihood function. Even if the initial classifier is optimal, in general the resulting labeling will not be the same as the true class membership; the act of classification will exclude samples from the tails of the desired distribution, and will include samples from the tails of the other distributions. Thus, if there is significant overlap between the component densities, one can expect biased estimates and less than optimal results.

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

## 10.6 *Data Description and Clustering

Let us reconsider our original problem of learning something of use from a set of unlabeled 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 $\mathbf{m}$ that best represents all of the data in the sense of minimizing the sum of squared distances from $\mathbf{m}$ to the samples. The sample covariance matrix describes the amount the data scatters along various directions. If the data points are actually normally distributed, then the cloud has a simple hyperellipsoidal shape, and the sample mean tends to fall in the region where the samples are most densely concentrated.

Of course, if the samples are not normally distributed, these statistics can give a very misleading description of the data. Figure 10.5 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.

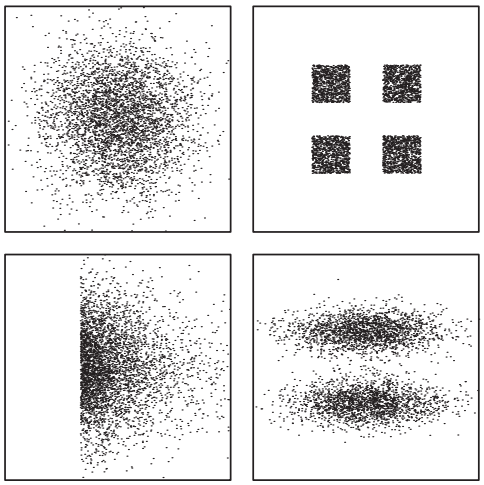


Figure 10.5: These four data sets have identical statistics up to second-order, i.e., the same mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. In such cases it is important to include in the model more parameters to represent the structure more completely.

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

it.

One alternative is to use one of the nonparametric methods described in Chap. **??** 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.

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

CLUSTERING PROCEDURE

### 10.6.1 Similarity Measures

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

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

In this section we address the first of these issues.

The most obvious measure of the similarity (or dissimilarity) between two samples is the distance between them. One way to begin a clustering investigation is to define a suitable 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 significantly less than the distance between samples in *different* clusters.

Suppose for the moment that we say that two samples belong to the same cluster if the Euclidean distance between them is less than some threshold distance $d_0$. It is immediately obvious that the choice of $d_0$ is very important. If $d_0$ is very large, all of the samples will be assigned to one cluster. If $d_0$ is very small, each sample will form an isolated, singleton cluster. To obtain "natural" clusters, $d_0$ will have to be greater than the typical within-cluster distances and less than typical between-cluster distances (Fig. 10.6).

Less obvious perhaps is the fact that the results of clustering depend on the choice of Euclidean distance as a measure of dissimilarity. That particular choice is generally justified if the feature space is isotropic and the data is spread roughly evenly along all directions. Clusters defined by Euclidean distance will be invariant to translations or rotations in feature space — rigid-body motions of the data points. However, they will not be invariant to linear transformations in general, or to other transformations that distort the distance relationships. Thus, as Fig. 10.7 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

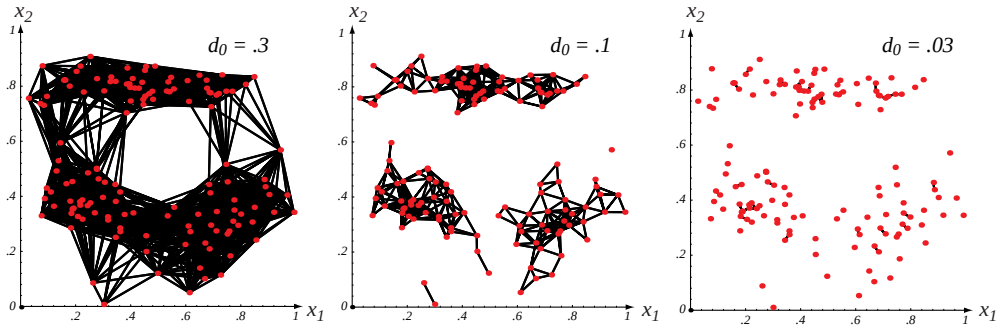



Figure 10.6: The distance threshold affects the number and size of clusters. Lines are drawn between points closer than a distance $d_0$ apart for three different values of $d_0$ — the smaller the value of $d_0$, the smaller and more numerous the clusters.

or meaningless transformation. However, if clusters are to mean anything, they should be invariant to transformations natural to the problem.

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

However, we should not conclude that this kind of normalization is necessarily desirable. Consider, for example, the matter of translating and whitening — 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. **??**). Thus, this routine normalization may be less than helpful in the cases of greatest interest.* Section **??** describes other 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

$$d(\mathbf{x}, \mathbf{x}') = \left( \sum_{k=1}^{d} |x_k - x'_k|^q \right)^{1/q}, \tag{44}$$

MINKOWSKI METRIC

where $q \geq 1$ is a selectable parameter — the general *Minkowski metric* we considered in Chap. **??**. 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. Note that only $q = 2$ is invariant to an arbitrary rotation or

CITY BLOCK METRIC

* 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.

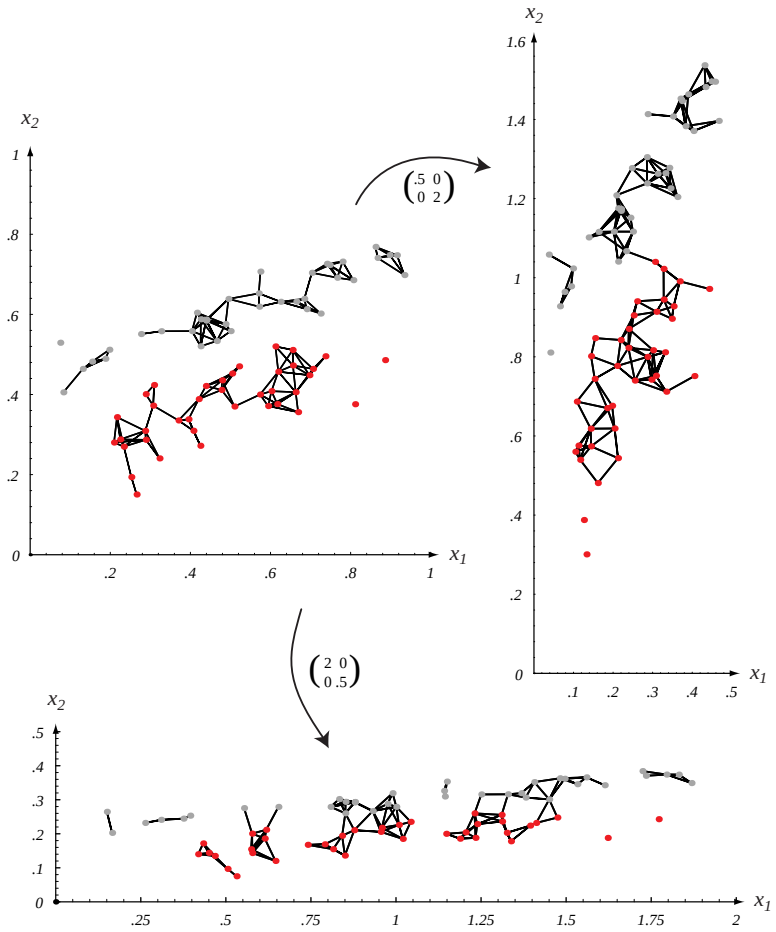



Figure 10.7: Scaling axes affects the clusters in a minimum distance cluster method. The original data and minimum-distance clusters are shown in the upper left — points in one cluster are shown in red, the other gray. When the vertical axis is expanded by a factor of 2.0 and the horizontal axis shrunk by a factor of 0.5, the clustering is altered (as shown at the right). Alternatively, if the vertical axis is shrunk by a factor of 0.5 and the horizontal axis expanded by a factor of 2.0, smaller more numerous clusters result (shown at the bottom). In both these scaled cases, the clusters differ from the original.

translation in feature space. Another alternative is to use some kind of metric based on the data itself, such as the Mahalanobis distance.

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

$$s(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^t \mathbf{x}'}{\|\mathbf{x}\| \, \|\mathbf{x}'\|} \tag{45}$$

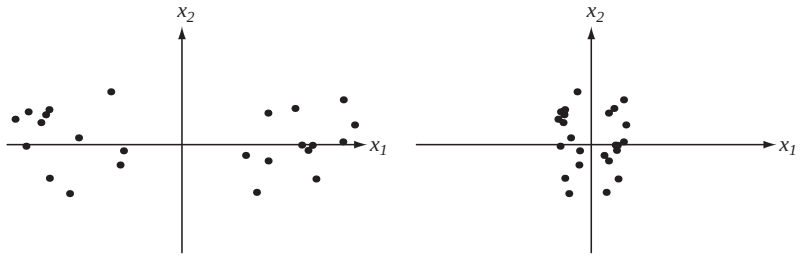



Figure 10.8: If the data fall into well-separated clusters (left), normalization by a whitening transform for the full data may reduce the separation, and hence be undesirable (right). Such a whitening normalization may be appropriate if the full data set arises from a single fundamental process (with noise), but inappropriate if there are several different processes, as shown here.

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 shared features or shared attributes. Let us say that a sample $\mathbf{x}$ *possesses* the $i$th attribute if $x_i = 1$. Then $\mathbf{x}^t\mathbf{x}'$ is merely the number of attributes possessed by both $\mathbf{x}$ and $\mathbf{x}'$, and $\|\mathbf{x}\|\,\|\mathbf{x}'\| = (\mathbf{x}^t\mathbf{x}\mathbf{x}'^t\mathbf{x}')^{1/2}$ is the geometric mean of the number of attributes possessed by $\mathbf{x}$ and the number possessed by $\mathbf{x}'$. Thus, $s(\mathbf{x}, \mathbf{x}')$ is a measure of the relative possession of common attributes. Some simple variations are

$$s(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^t\mathbf{x}'}{d}, \tag{46}$$

the fraction of attributes shared, and

$$s(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^t\mathbf{x}'}{\mathbf{x}^t\mathbf{x} + \mathbf{x}'^t\mathbf{x}' - \mathbf{x}^t\mathbf{x}'}, \tag{47}$$

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

TANIMOTO DISTANCE

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. (Competitive

learning, discussed in Sect. 10.11, is a popular decision directed clustering algorithm.) Beyond that we can do little more than alert the unwary to these pitfalls of clustering.

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

## 10.7 Criterion Functions for Clustering

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

### 10.7.1 The Sum-of-Squared-Error Criterion

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

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x}. \tag{48}$$

Then the sum-of-squared errors is defined by

$$J_e = \sum_{i=1}^{c} \sum_{\mathbf{x} \in \mathcal{D}_i} \|\mathbf{x} - \mathbf{m}_i\|^2. \tag{49}$$

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

MINIMUM VARIANCE

What kind of clustering problems are well suited to a sum-of-squared-error criterion? Basically, $J_e$ is an appropriate criterion when the clusters form compact clouds that are rather well separated from one another. A less obvious problem arises when there are great differences in the number of samples in different clusters. In that case it can happen that a partition that splits a large cluster is favored over one that maintains the integrity of the natural clusters, as illustrated in Fig. 10.9. 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.

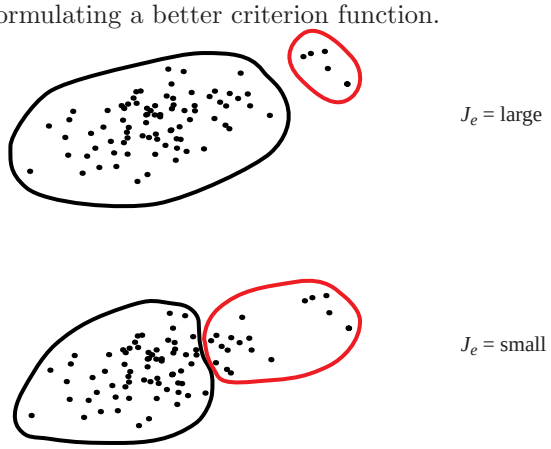



Figure 10.9: When two natural groupings have very different numbers of points, the clusters minimizing a sum-squared-error criterion (Eq. 49) may not reveal the true underlying structure. Here the criterion is smaller for the two clusters at the bottom than at the more natural clustering at the top.

### 10.7.2 Related Minimum Variance Criteria

By some simple algebraic manipulation (Problem 19) we can eliminate the mean vectors from the expression for $J_e$ and obtain the equivalent expression

$$J_e = \frac{1}{2}\sum_{i=1}^{c} n_i \bar{s}_i, \tag{50}$$

where

$$\bar{s}_i = \frac{1}{n^2}\sum_{\mathbf{x}\in\mathcal{D}_i}\sum_{\mathbf{x}'\in\mathcal{D}_i} \|\mathbf{x}-\mathbf{x}'\|^2. \tag{51}$$

Equation 51 leads us to interpret $\bar{s}_i$ as the average squared distance between points in the $i$th 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

$$\bar{s}_i = \frac{1}{n_i^2}\sum_{\mathbf{x}\in\mathcal{D}_i}\sum_{\mathbf{x}'\in\mathcal{D}_i} s(\mathbf{x},\mathbf{x}') \tag{52}$$

or

$$\bar{s}_i = \min_{\mathbf{x},\mathbf{x}'\in\mathcal{D}_i} s(\mathbf{x},\mathbf{x}'). \tag{53}$$

Table 10.1: Mean vectors and scatter matrices used in clustering criteria.

| | Depend on cluster center? | | |
|---|---|---|---|
| | Yes | No | |
| Mean vector for the $i$th cluster | | × | $\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x}$ (54) |
| Total mean vector | | × | $\mathbf{m} = \frac{1}{n} \sum_{\mathcal{D}} \mathbf{x} = \frac{1}{n} \sum_{i=1}^{c} n_i \mathbf{m}_i$ (55) |
| Scatter matrix for the $i$th cluster | × | | $\mathbf{S}_i = \sum_{\mathbf{x} \in \mathcal{D}_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^t$ (56) |
| Within-cluster scatter matrix | × | | $\mathbf{S}_W = \sum_{i=1}^{c} \mathbf{S}_i$ (57) |
| Between-cluster scatter matrix | × | | $\mathbf{S}_B = \sum_{i=1}^{c} n_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^t$ (58) |
| Total scatter matrix | | × | $\mathbf{S}_T = \sum_{\mathbf{x} \in \mathcal{D}} (\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^t$ (59) |

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

### 10.7.3 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 those given in Chapt. **??**.

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:

$$\mathbf{S}_T = \mathbf{S}_W + \mathbf{S}_B. \tag{60}$$

Note that the total scatter matrix does not depend on how the set of samples is partitioned into clusters; it depends only on the total set of samples. The within-cluster and between-cluster scatter matrices taken separately do depend on the partitioning, of course. Roughly speaking, there is an exchange between these two matrices, the between-cluster scatter 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. 56 & 57) yield

$$tr\ \mathbf{S}_W = \sum_{i=1}^{c} tr\ \mathbf{S}_i = \sum_{i=1}^{c} \sum_{\mathbf{x}\in\mathcal{D}_i} \|\mathbf{x} - \mathbf{m}_i\|^2 = J_e. \tag{61}$$

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

$$trS_B = \sum_{i=1}^{c} n_i \|\mathbf{m}_i - \mathbf{m}\|^2. \tag{62}$$

### The Determinant Criterion

In Sect. **??** we used the determinant of the scatter matrix to obtain a scalar measure of scatter. Roughly speaking, the determinant measures the square of the scattering volume, since it is proportional to the product of the variances in the directions of the principal axes. 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 $n - c$ is less than the dimensionality $d$ (Problem 27). However, if we assume that $\mathbf{S}_W$ is nonsingular, we are led to consider the determinant criterion function

$$J_d = |\mathbf{S}_W| = \left| \sum_{i=1}^{c} \mathbf{S}_i \right|. \tag{63}$$

The partition that minimizes $J_d$ is often similar to the one that minimizes $J_e$, but the two need not be the same, as shown in Example 3. 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 26). Thus $J_d$ is to be favored under conditions where there may be unknown or irrelevant linear transformations of the data.

### Invariant Criteria

It is not particularly hard to show that the eigenvalues $\lambda_1, \ldots, \lambda_d$ of $\mathbf{S}_W^{-1}\mathbf{S}_B$ are invariant under nonsingular linear transformations of the data (Problem **??**). 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

$$tr\mathbf{S}_W^{-1}\mathbf{S}_B = \sum_{i=1}^{d} \lambda_i. \tag{64}$$

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 25):

$$J_f = tr\mathbf{S}_T^{-1}\mathbf{S}_W = \sum_{i=1}^{d} \frac{1}{1+\lambda_i} \tag{65}$$

and

$$\frac{|\mathbf{S}_W|}{|\mathbf{S}_T|} = \prod_{i=1}^{d} \frac{1}{1+\lambda_i}. \tag{66}$$

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, as shown in Example 3.

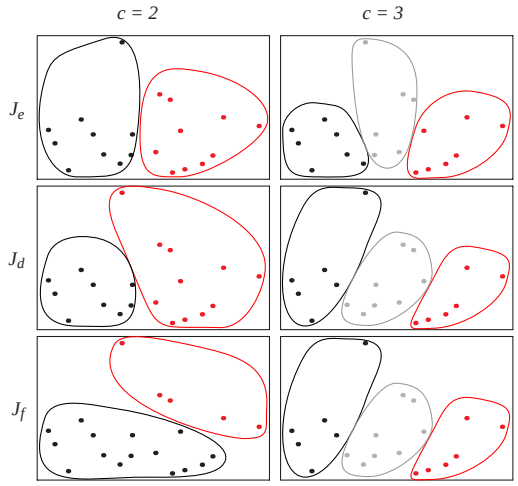
Example 3: Clustering criteria

We can gain some intuition by considering these criteria applied to the following data set.

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

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

All of the clusterings seem reasonable, and there is no strong argument to favor one over the others. For the case $c = 2$, the clusters minimizing the $J_e$ indeed tend to favor clusters of roughly equal numbers of points, as illustrated in Fig. 10.9; in contrast, $J_d$ favors one large and one fairly small cluster. Since the full data set happens to be spread horizontally more than vertically, the eigenvalue in the horizontal direction is greater than that in the vertical direction. As such, the clusters are "stretched"




The clusters found by minimizing a criterion depends upon the criterion function as well as the assumed number of clusters. The sum-of-squared-error criterion $J_e$ (Eq. 49), the determinant criterion $J_d$ (Eq. 63) and the more subtle trace criterion $J_f$ (Eq. 65) were applied to the 20 points in the table with the assumption of $c = 2$ and $c = 3$ clusters. (Each point in the table is shown, with bounding boxes defined by $-1.8 < x < 2.5$ and $-0.6 < y < 1.9$.)

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

With regard to the criterion function involving $\mathbf{S}_T$, note that $\mathbf{S}_T$ does not depend on how the samples are partitioned into clusters. Thus, the clusterings that minimize $|\mathbf{S}_W|/|\mathbf{S}_T|$ are exactly the same as the ones that minimize $|\mathbf{S}_W|$. If we rotate and scale the axes so that $\mathbf{S}_T$ becomes the identity matrix, we see that minimizing $tr[\mathbf{S}_T^{-1}\mathbf{S}_W]$ 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 possess multiple local extrema, and are correspondingly more difficult to optimize.

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

criterion functions that are better matched to the structure present or being sought.

## 10.8 *Iterative Optimization

Once a criterion function has been selected, clustering becomes a well-defined problem in discrete optimization: find those partitions of the set of samples that extremize the criterion function. 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 17). For example an exhaustive search for the best set of 5 clusters in 100 samples would require considering more than $10^{67}$ partitionings. Simply put, in most applications an exhaustive search is completely infeasible.

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

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

$$J_e = \sum_{i=1}^{c} J_i, \tag{67}$$

where an effective error per cluster is defined to be

$$J_i = \sum_{\mathbf{x}\in\mathcal{D}_i} \|\mathbf{x} - \mathbf{m}_i\|^2 \tag{68}$$

and the mean of each cluster is, as before,

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x}\in\mathcal{D}_i} \mathbf{x}. \tag{48}$$

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

$$\mathbf{m}_j^* = \mathbf{m}_j + \frac{\hat{\mathbf{x}} - \mathbf{m}_j}{n_j + 1} \tag{69}$$

and $J_j$ increases to

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

$$= \quad J_j + \frac{n_j}{n_j + 1}\|\hat{\mathbf{x}} - \mathbf{m}_j\|^2. \tag{70}$$

Under the assumption that $n_i \neq 1$ (singleton clusters should not be destroyed), a similar calculation (Problem 29) shows that $\mathbf{m}_i$ changes to

$$\mathbf{m}_i^* = \mathbf{m} - \frac{\hat{\mathbf{x}} - \mathbf{m}_i}{n_i - 1} \tag{71}$$

and $J_i$ decreases to

$$J_i^* = J_i - \frac{n_i}{n_i - 1}\|\hat{\mathbf{x}} - \mathbf{m}_i\|^2. \tag{72}$$

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

$$\frac{n_i}{n_i - 1}\|\hat{\mathbf{x}} - \mathbf{m}_i\|^2 > \frac{n_j}{n_j + 1}\|\hat{\mathbf{x}} - \mathbf{m}_j\|^2, \tag{73}$$

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

**Algorithm 3 (Basic iterative minimum-squared-error clustering)**

1 **begin initialize** $n, c, \mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$
2 **do** randomly select a sample $\hat{\mathbf{x}}$;
3 $i \leftarrow \arg\min_{i'} \|\mathbf{m}_{i'} - \hat{\mathbf{x}}\|$ (classify $\hat{\mathbf{x}}$)
4 **if** $n_i \neq 1$ **then** compute
5 $\rho_j = \begin{cases} \frac{n_j}{n_j+1}\|\hat{\mathbf{x}} - \mathbf{m}_j\|^2 & j \neq i \\ \frac{n_j}{n_j-1}\|\hat{\mathbf{x}} - \mathbf{m}_i\|^2 & j = i \end{cases}$
6 **if** $\rho_k \leq \rho_j$ for all $j$ **then** transfer $\hat{\mathbf{x}}$ to $\mathcal{D}_k$
7 recompute $J_e, \mathbf{m}_i, \mathbf{m}_k$
8 **until** no change in $J_e$ in $n$ attempts
9 **return** $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$
10 **end**

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

One question that plagues all hill-climbing procedures is the choice of the starting point. Unfortunately, there is no simple, universally good solution to this problem. One approach is to select $c$ samples randomly for the initial cluster centers, using them to partition the data on a minimum-distance basis. Repetition with different random selections can give some indication of the sensitivity of the solution to the

starting point. Yet another approach is to find the $c$-cluster starting point from the solutions to the $(c-a)$-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.

## 10.9 Hierarchical Clustering

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

### 10.9.1 Definitions

Let us consider a sequence of partitions of the $n$ samples into $c$ clusters. The first of these is a partition into $n$ clusters, each cluster containing exactly one sample. The next is a partition into $n-1$ clusters, the next a partition into $n-2$, and so on until the $n$th, in which all the samples form one cluster. We shall say that we are at level $k$ in the sequence when $c = n-k+1$. Thus, level one corresponds to $n$ clusters and level $n$ 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*.

The most natural representation of hierarchical clustering is a corresponding tree, called a *dendrogram*, which shows how the samples are grouped. Figure 10.10 shows a dendrogram for a simple problem involving eight samples. Level 1 shows the eight samples as singleton clusters. At level 2, samples $\mathbf{x}_6$ and $\mathbf{x}_7$ have been grouped to form a cluster, and they stay together at all subsequent levels. If it is possible to measure the similarity between clusters, then the dendrogram is usually drawn to scale to show the similarity between the clusters that are grouped. In Fig. 10.10, for example, the similarity between the two groups of samples that are merged at level 5 has a value of roughly 60.

DENDROGRAM

We shall see shortly how such similarity values can be obtained, but first note that the similarity values can be used to help determine whether groupings are natural or forced. If the similarity values for the levels are roughly evenly distributed throughout

the range of possible values, then there is no principled argument that any particular number of clusters is better or "more natural" than another. Conversely, suppose that there is a unusually large gap between the similarity values for the levels corresponding to $c = 3$ and to $c = 4$ clusters. In such a case, one can argue that $c = 3$ is the most natural number of clusters (Problem 35).

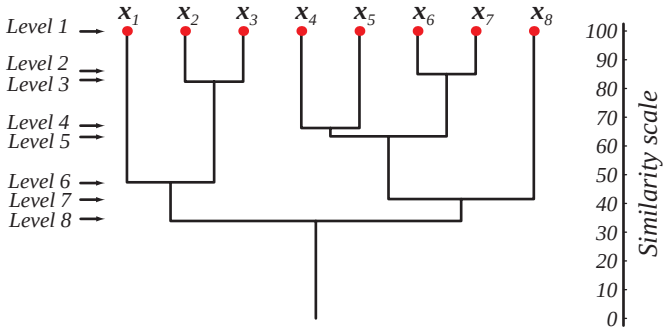



Figure 10.10: A dendrogram can represent the results of hierarchical clustering algorithms. The vertical axis shows a generalized measure of similarity among clusters. Here, at level 1 all eight points lie in singleton clusters; each point in a cluster is highly similar to itself, of course. Points $\mathbf{x}_6$ and $\mathbf{x}_7$ happen to be the most similar, and are merged at level 2, and so forth.

Another representation for hierarchical clustering is based on sets, in which each level of cluster may contain sets that are subclusters, as shown in Fig. 10.11. Yet another, textual, representation uses brackets, such as: $\{\{\mathbf{x}_1, \{\mathbf{x}_2, \mathbf{x}_3\}\}, \{\{\{\mathbf{x}_4, \mathbf{x}_5\}, \{\mathbf{x}_6, \mathbf{x}_7\}\}, \mathbf{x}_8\}\}$. While such representations may reveal the hierarchical structure of the data, they do not naturally represent the similarities *quantitatively*. For this reason dendrograms are generally preferred.

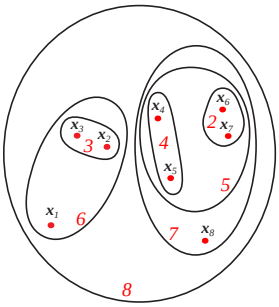



Figure 10.11: A set or Venn diagram representation of two-dimensional data (which was used in the dendrogram of Fig. 10.10) reveals the hierarchical structure but not the quantitative distances between clusters. The levels are numbered in red.

AGGLOMERATIVE

DIVISIVE

Because of their conceptual simplicity, hierarchical clustering procedures are among the best-known of unsupervised methods. The procedures themselves can be divided according to two distinct approaches — agglomerative and divisive. *Agglomerative* (bottom-up, clumping) procedures start with $n$ singleton clusters and form the sequence by successively merging clusters. *Divisive* (top-down, splitting) procedures start with all of the samples in one cluster and form the sequence by successively splitting clusters. The computation needed to go from one level to another is usually

simpler for the agglomerative procedures. However, when there are many samples and one is interested in only a small number of clusters, this computation will have to be repeated many times. For simplicity, we shall concentrate on agglomerative procedures, and merely touch on some divisive methods in Sect. 10.12.

### 10.9.2 Agglomerative Hierarchical Clustering

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

**Algorithm 4 (Agglomerative hierarchical clustering)**

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

As described, this procedure terminates when the specified number of clusters has been obtained and returns the clusters, described as set of points (rather than as mean or representative vectors). If we continue until $c = 1$ we can produce a dendrogram like that in Fig. 10.10. 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, required by line 3 of the Algorithm. The considerations here are much like those involved in selecting a general clustering criterion function. For simplicity, we shall generally restrict our attention to the following distance measures:

$$d_{min}(\mathcal{D}_i, \mathcal{D}_j) = \min_{\substack{\mathbf{x}\in\mathcal{D}_i \\ \mathbf{x}'\in\mathcal{D}_j}} \|\mathbf{x}-\mathbf{x}'\| \tag{74}$$

$$d_{max}(\mathcal{D}_i, \mathcal{D}_j) = \max_{\substack{\mathbf{x}\in\mathcal{D}_i \\ \mathbf{x}'\in\mathcal{D}_j}} \|\mathbf{x}-\mathbf{x}'\| \tag{75}$$

$$d_{avg}(\mathcal{D}_i, \mathcal{D}_j) = \frac{1}{n_i n_j} \sum_{\mathbf{x}\in\mathcal{D}_i} \sum_{\mathbf{x}'\in\mathcal{D}_j} \|\mathbf{x}-\mathbf{x}'\| \tag{76}$$

$$d_{mean}(\mathcal{D}_i, \mathcal{D}_j) = \|\mathbf{m}_i - \mathbf{m}_j\|. \tag{77}$$

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

But first let us consider the computational complexity of a particularly simple agglomerative clustering algorithm. Suppose we have $n$ patterns in $d$-dimensional space, and we seek to form $c$ clusters using $d_{min}(\mathcal{D}_i, \mathcal{D}_j)$ defined in Eq. 74. We will, once and for all, need to calculate $n(n-1)$ inter-point distances — each of which is an $O(d^2)$ calculation — and place the results in an inter-point distance table. The space complexity is, then, $O(n^2)$. Finding the minimum distance pair (for the first merging) requires that we step through the complete list, keeping the

index of the smallest distance. Thus for the first agglomerative step, the complexity is $O(n(n-1)(d^2+1)) = O(n^2d^2)$. For an arbitrary agglomeration step (i.e., from $\hat{c}$ to $\hat{c}-1$), we need merely step through the $n(n-1)-\hat{c}$ "unused" distances in the list and find the smallest for which $\mathbf{x}$ and $\mathbf{x}'$ lie in different clusters. This is, again, $O(n(n-1)-\hat{c})$. The full time complexity is thus $O(cn^2d^2)$, and in typical conditions $n \gg c$.*

### The Nearest-Neighbor Algorithm

MINIMUM ALGORITHM

SINGLE-LINKAGE ALGORITHM

When $d_{min}$ is used to measure the distance between clusters (Eq. 74) the algorithm is sometimes called the nearest-neighbor cluster algorithm, or *minimum algorithm* Moreover, if it is terminated when the distance between nearest clusters exceeds an arbitrary threshold, it is called the *single-linkage algorithm*. Suppose that we think of the data points as being nodes of a graph, with edges forming a path between the nodes in the same subset $\mathcal{D}_i$. When $d_{min}$ is used to measure the distance between subsets, the nearest neighbor nodes determine the nearest subsets. The merging of $\mathcal{D}_i$ and $\mathcal{D}_j$ corresponds to adding an edge between the nearest pair of nodes in $\mathcal{D}_i$ and $\mathcal{D}_j$. 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 (Problem 37). 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 10.12 shows the results of applying this procedure to Gaussian data. In both cases the procedure was stopped giving two large clusters (plus three singleton outliers); a minimal spanning tree can be obtained by adding the shortest possible edge between the two clusters. In the first case where the clusters are fairly well separated, the obvious clusters are found. In the second case, the presence of a point located so as to produce a bridge between the clusters results in a rather unexpected grouping into one large, elongated cluster, and one small, compact cluster. This behavior is often called the "chaining effect," and is sometimes considered to be a defect of this distance measure. To the extent that the results are very sensitive to noise or to slight changes in position of the data points, this is certainly a valid criticism.

### The Farthest-Neighbor Algorithm

MAXIMUM ALGORITHM

COMPLETE-LINKAGE ALGORITHM

When $d_{max}$ (Eq. 75) is used to measure the distance between subsets, the algorithm is sometimes called the farthest-neighbor clustering algorithm, or *maximum algorithm*. If it is terminated when the distance between nearest clusters exceeds an arbitrary threshold, it is called the *complete-linkage algorithm*. The farthest-neighbor algorithm discourages the growth of elongated clusters. Application of the procedure can be thought of as producing a graph in which edges connect all of the nodes in a cluster. In the terminology of graph theory, every cluster constitutes a *complete* subgraph. The distance between two clusters is determined by the most distant nodes in the two

COMPLETE SUBGRAPH

* 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.

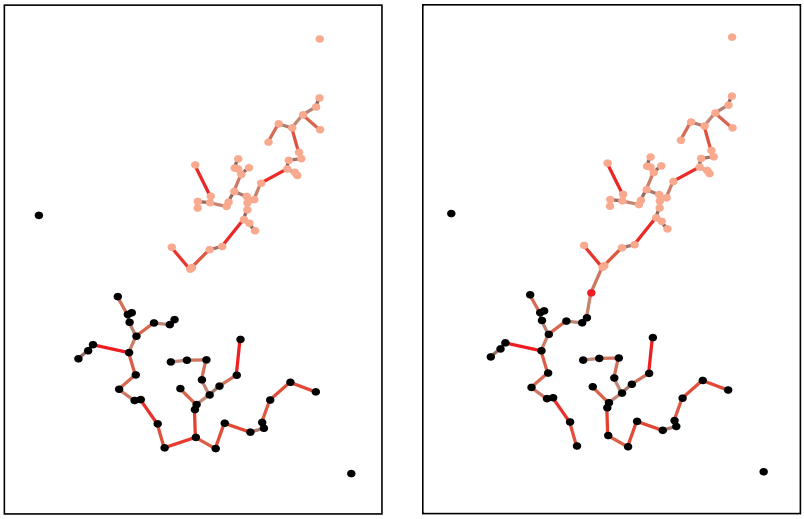


Figure 10.12: Two Gaussians were used to generate two-dimensional samples, shown in pink and black. The nearest-neighbor clustering algorithm gives two clusters that well approximate the generating Gaussians (left). If, however, another particular sample is generated (red point at the right) and the procedure re-started, the clusters do not well approximate the Gaussians. This illustrates how the algorithm is sensitive to the details of the samples.

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

#### Compromises

The minimum and maximum measures represent two extremes in measuring the distance between clusters. Like all procedures that involve minima or maxima, they tend to be overly sensitive to "outliers" or "wildshots." The use of averaging is an obvious way to ameliorate these problems, and $d_{avg}$ and $d_{mean}$ (Eqs. 76 & 77) 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 $\|\mathbf{x} - \mathbf{x}'\|$. However, a measure such as $d_{avg}$ can be used when the distances $\|\mathbf{x} - \mathbf{x}'\|$ are replaced by similarity measures, where the similarity between mean vectors may be difficult or impossible to define.

### 10.9.3 Stepwise-Optimal Hierarchical Clustering

We observed earlier that if clusters are grown by merging the nearest pair of clusters, then the results have a minimum variance flavor. However, when the measure

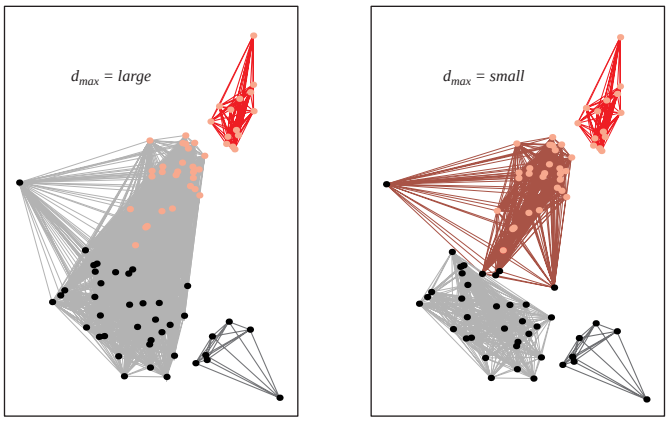



Figure 10.13: The farthest-neighbor clustering algorithm uses the separation between the most distant points as a criterion for cluster membership. If this distance is set very large, then all points lie in the same cluster. In the case shown at the left, a fairly large $d_{max}$ leads to three clusters; a smaller $d_{max}$ gives four clusters, as shown at the right.

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. Nevertheless, with a simple modification it is possible to obtain a stepwise-optimal procedure for extremizing a criterion function. This is done merely by replacing line 3 of the Basic Iterative Agglomerative Clustering Procedure (Algorithm 4) by a more general form to get:

**Algorithm 5 (Stepwise optimal hierarchical clustering)**

1 **begin** **initialize** $c, \hat{c} \leftarrow n, \mathcal{D}_i \leftarrow \{\mathbf{x}_i\}, i = 1, \ldots, n$
2 **do** $\hat{c} \leftarrow \hat{c} - 1$
3 Find clusters whose merger changes the criterion the least, say, $\mathcal{D}_i$ and $\mathcal{D}_j$
4 Merge $\mathcal{D}_i$ and $\mathcal{D}_j$
5 **until** $c = \hat{c}$
6 **return** $c$ clusters
7 **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"

$$d_e(\mathcal{D}_i, \mathcal{D}_j) = \sqrt{\frac{n_i n_j}{n_i + n_j}} \|\mathbf{m}_i - \mathbf{m}_j\| \tag{78}$$

is minimum (Problem 34). 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 merging singletons or small clusters with large clusters over merging medium-sized clusters. While the final partition may not minimize $J_e$, it usually provides a very good starting point for further iterative optimization.

### 10.9.4 Hierarchical Clustering and Induced Metrics

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

DISSIMILARITY

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

or

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

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 18).

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

METRIC

**non-negativity:** $d(\mathbf{x}, \mathbf{x}') \geq 0$

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

**symmetry:** $d(\mathbf{x}, \mathbf{x}') = d(\mathbf{x}', \mathbf{x})$

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

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

$$d(\mathbf{x}, \mathbf{x}'') \leq max[d(\mathbf{x}, \mathbf{x}'), d(\mathbf{x}', \mathbf{x}'')] \quad \text{for any } \mathbf{x}' \tag{81}$$

in which case we say that $d(\cdot, \cdot)$ is an *ultrametric* (Problem 31). Ultrametric criteria can be more immune to local minima problems since stricter ordering of distances among clusters is maintained.

ULTRAMETRIC

## 10.10 *The Problem of Validity

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

When clustering is done by extremizing a criterion function, a common approach is to repeat the clustering procedure for $c = 1$, $c = 2$, $c = 3$, 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 a new singleton cluster. If the $n$ samples are really grouped into $\hat{c}$ compact, well separated clusters, one would expect to see $J_e$ decrease rapidly until $\hat{c} = c$, decreasing much more slowly thereafter until it reaches zero at $c = n$. Similar arguments have been advanced for hierarchical clustering procedures and can be apparent in a dendrogram, the usual assumption being that a large disparity in the levels at which clusters merge indicates the presence of natural groupings.

A more formal approach to this problem is to devise some measure of goodness of fit that expresses how well a given $c$-cluster description matches the data. The chi-squared and Kolmogorov-Smirnov statistics are the traditional measures of goodness of fit, but the curse of dimensionality usually demands the use of simpler measures, some criterion function, which we denote $J(c)$. 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 sum-of-squared-error criterion which closely parallels our discussion in Chap. **??**.

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

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

$$J_e(1) = \sum_{\mathbf{x}\in\mathcal{D}} \|\mathbf{x} - \mathbf{m}\|^2, \tag{82}$$

where $\mathbf{m}$ is the sample mean of the full data set. Under the null hypothesis, the distribution for $J_e(1)$ is approximately normal with mean $nd\sigma^2$ and variance $2nd\sigma^4$ (Problem 38). Suppose now that we partition the set of samples into two subsets $\mathcal{D}_1$ and $\mathcal{D}_2$ so as to minimize $J_e(2)$, where

$$J_e(2) = \sum_{i=1}^{2} \sum_{\mathbf{x}\in\mathcal{D}_i} \|\mathbf{x} - \mathbf{m}_i\|^2, \tag{83}$$

$\mathbf{m}_i$ being the mean of the samples in $\mathcal{D}_i$. Under the null hypothesis, this partitioning is spurious, but it nevertheless results in a value for $J_e(2)$ that is smaller than $J_e(1)$.

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

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

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

$$\hat{\sigma}^2 = \frac{1}{nd}\sum_{\mathbf{x}\in\mathcal{D}} \|\mathbf{x}-\mathbf{m}\|^2 = \frac{1}{nd}J_e(1). \tag{84}$$

The final result can be stated as follows (Problem 39): Reject the null hypothesis at the $p$-percent significance level if

$$\frac{J_e(2)}{J_e(1)} < 1 - \frac{2}{\pi d} - \alpha\sqrt{\frac{2(1-8/\pi^2 d)}{nd}}, \tag{85}$$

where $\alpha$ is determined by

$$p = 100\int_{\alpha}^{\infty} \frac{1}{2\pi}e^{-u^2/2}\,du = 100(1-erf(\alpha)), \tag{86}$$

and $erf(\cdot)$ 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

## 10.11 Competitive Learning

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

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

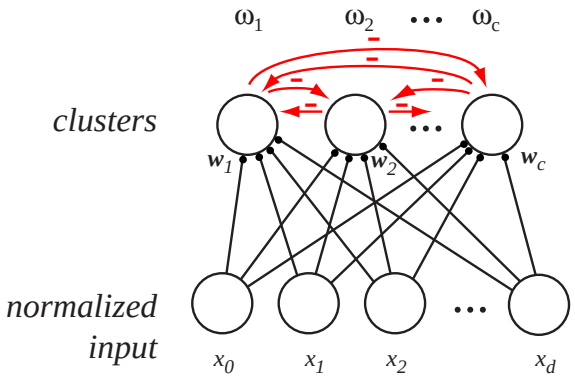



Figure 10.14: The two-layer network which implements the competitive learning algorithm consists of $d+1$ input units and $c$ output or cluster units. Each augmented input pattern is normalized to unit length, i.e., $\|\mathbf{x}\| = 1$, as is the set of weights at each cluster unit. When a pattern is presented, each of the cluster units computes its net activation $net_j = \mathbf{w}_j^t\mathbf{x}$; only the weights at the most active cluster unit are modified. (The suppression of activity in all but the most active cluster units can be implemented by competition among these units, as indicated by the red arrows.) The weights of the most active unit are then modified to be more similar to the pattern presented.

The competitive learning algorithm can be understood by its neural network implementation (Fig. 10.14), which resembles a Perceptron network (Chapt. **??**, Fig. **??**), with input units fully connected to $c$ output or cluster units.

Each of the $c$ cluster centers is initialized with a randomly chosen weight vector, also normalized $\|\mathbf{w}_j\| = 1$, $j = 1, \ldots c$. It is traditional but not required to initialize cluster centers to be $c$ points randomly selected from the data. When a new pattern is presented, each of the cluster units computes its net activation, $net_j = \mathbf{w}_j^t\mathbf{x}$. Only the most active neuron (i.e., the closest to the new pattern) is permitted to update its weights. While this selection of the most active unit is algorithmically trivial, it can be implemented in a winner-take-all network, where each cluster unit $j$ inhibits others by an amount proportional to $net_j$, as shown by the red arrows in Fig. 10.14. It is this competition between cluster units, and the rsulting suppression of activity in all but the one with the largest *net* that gives the algorithm its name.

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

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta\mathbf{x}, \tag{87}$$

where $\eta$ is a learning rate. The weights are then normalized to insure $\sum_{i=0}^{d} w_i^2 = 1$. This normalization is needed to keep the classification and clustering based on the position in feature space rather than overall magnitude of $\mathbf{w}$. Without such weight normalization, a single weight, say $\mathbf{w}_{j'}$, could grow in magnitude and forever give the greatest value $net_{j'}$, and through competition thereby prevent other clusters from learning. Figure 10.15 shows the trajectories of three cluster centers in response to a sequence of patterns chosen randomly from the set shown.

**Algorithm 6 (Competitive learning)**

1 **begin initialize** $\eta, n, c, \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_c$
2 $\quad \mathbf{x}_i \leftarrow \{1, \mathbf{x}_i\}\ i = 1, \ldots n$ augment all patterns
3 $\quad \mathbf{x}_i \leftarrow \mathbf{x}_i / \|\mathbf{x}_i\|\ i = 1, \ldots n$ normalize all patterns
4 $\quad$ **do** randomly select a pattern $\mathbf{x}$
5 $\quad\quad j \leftarrow \arg\max_{j'} \mathbf{w}_{j'}^t \mathbf{x}$ classify $\mathbf{x}$
6 $\quad\quad \mathbf{w}_j \leftarrow \mathbf{w}_j + \eta \mathbf{x}$ weight update
7 $\quad\quad \mathbf{w}_j \leftarrow \mathbf{w}_j / \|\mathbf{w}_j\|$ weight normalization
8 $\quad$ **until** no significant change in $\mathbf{w}$ in $n$ attempts
9 **return** $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_c$
10 **end**

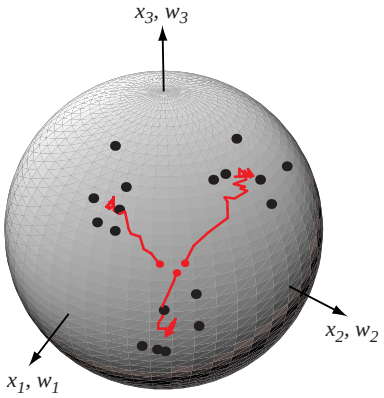



Figure 10.15: All of the three-dimensional patterns have been normalized ($\sum_{i=1}^{3} x_i^2 = 1$), and hence lie on a two-dimensional sphere. Likewise, the weights of the three cluster centers have been normalized. The red curves show the trajectory of the weight vectors; at the end of learning, each lies near the center of a cluster.

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

In a non-stationary environment, a we may want a clustering algorithm to be stable to prevent ceaseless recoding, and yet plastic, or changeable, 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 see in Sect. 10.11.2 how it can be addressed. First, however, we turn to the problem of unknown number of clusters.

STABILITY-PLASTICITY

### 10.11.1 Unknown number of clusters

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

Whereas clustering algorithms such as k-means and hierarchical clustering typically have all data present before clustering begins (i.e., are off-line), there are occasionally situations in which clustering must be performed on-line as the data streams 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 performed on-line — one merely links the new pattern to an existing cluster based on some similarity measure.

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

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. 10.16).

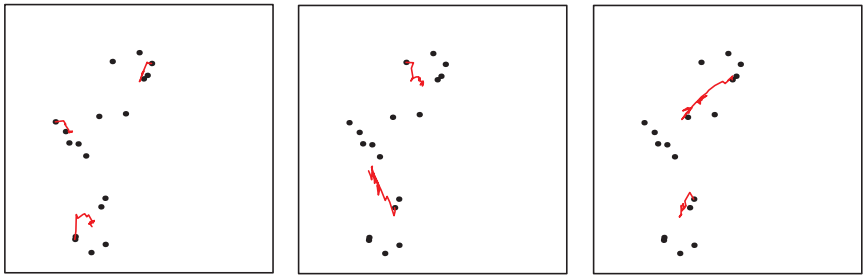


Figure 10.16: In leader-follower clustering, the number of clusters and their centers depend upon the random sequence of data presentations. The three simulations shown employed the same learning rate $\eta$, threshold $\theta$, and number of presentations of each point (50), but differ in the random sequence of presentations. Notice that in the simulation on the left, three clusters are generated whereas in the other simulations, only two.

If we let $\mathbf{w}_i$ represent the current center for cluster $i$, $\eta$ a learning rate and introduce a threshold $\theta$, a relative of the Basic leader-follower clustering algorithm is then:

**Algorithm 7 (Basic leader-follower clustering)**

1 **begin** **initialize** $\eta, \theta \leftarrow$ threshold

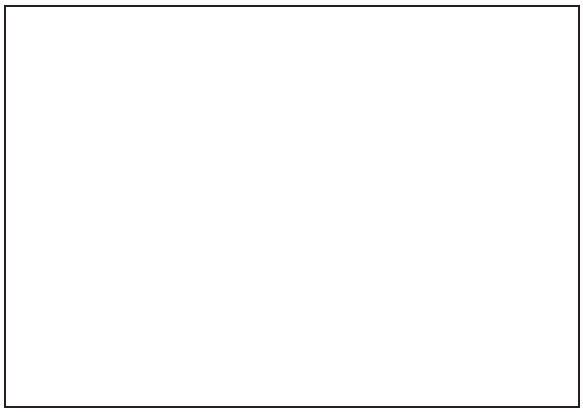


Figure 10.17: Instability can arise when a pattern is assigned different cluster memberships at different times. Early in clustering the pattern marked $\mathbf{x}^*$ lies in the black cluster, while later in clustering it lies in the red cluster. Similar pattern presentations can make $\mathbf{x}^*$ alternate arbitrarily between clusters.

```
2        μ_1 ← x
3        do accept new x
4           j ← arg min_j' ||x − μ_j'||     (find nearest cluster)
5           if ||x − μ_j|| < θ
6              then μ_j ← μ_j + ηx
7              else add new μ ← x
8              μ ← μ/||μ||     (normalize weight)
9           until no more patterns
10       return μ_1, μ_2, ...
11   end
```

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

### 10.11.2 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 41).

The above algorithm, however, can occasionally 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. 10.17). 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.

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-

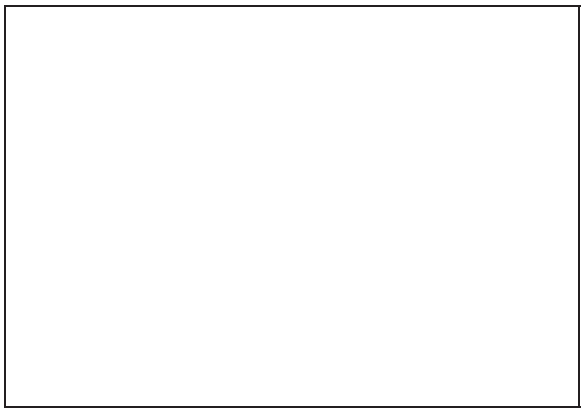


Figure 10.18: Adaptive Resonance network (ART1 for binary patterns). Weights are bidirectional, gain, the orienting system controls the , and hence (indirectly) the number of clusters found.

take-all computation leads to only the most activated output 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).

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.

VIGILANCE

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

$$\frac{|R|}{|I|} < \rho, \tag{88}$$

VIGILANCE PARAMETER

where *rho* is a user-set number called the *vigilance parameter*. A low vigilance parameter 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 $\delta$). 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. 10.19).

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.

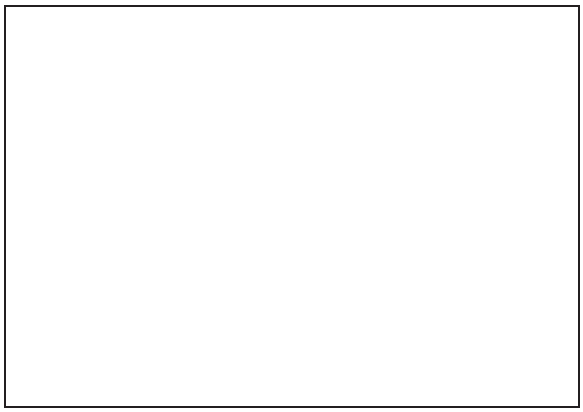


Figure 10.19: The results of ART1 applied to a sequence of binary figures. a) $\rho = xx$. b) $\rho = 0.xx$.

## 10.12 *Graph Theoretic Methods

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

We begin our brief look into graph-theoretic methods by reconsidering the simple procedures that produce the graphs shown in Fig. 10.6. Here a threshold distance $d_0$ was selected, and two points are placed in the same cluster if the distance between them is less than $d_0$. This procedure can easily be generalized to apply to arbitrary similarity measures. Suppose that we pick a threshold value $s_0$ and say that $\mathbf{x}_i$ is similar to $\mathbf{x}_j$ if $s(\mathbf{x}_i, \mathbf{x}_j) > s_0$. This defines an $n$-by-$n$ *similarity matrix* $\mathbf{S} = [s_{ij}]$, with binary component

SIMILARITY MATRIX

$$s_{ij} = \begin{cases} 1 & \text{if } s(\mathbf{x}_i, \mathbf{x}_j) > s_0 \\ 0 & \text{otherwise.} \end{cases} \tag{89}$$

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

SIMILARITY GRAPH

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

CONNECTED COMPONENT

MAXIMAL COMPLETE SUBGRAPH

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 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 10.20 shows a minimal spanning tree for a two-dimensional point set and the clusters obtained by systematically removing all edges for which $l > 2\bar{l}$ in this way. This criterion is sensitive to local conditions gives results that are quite different from merely removing the two longest edges.

INCONSISTENT EDGE

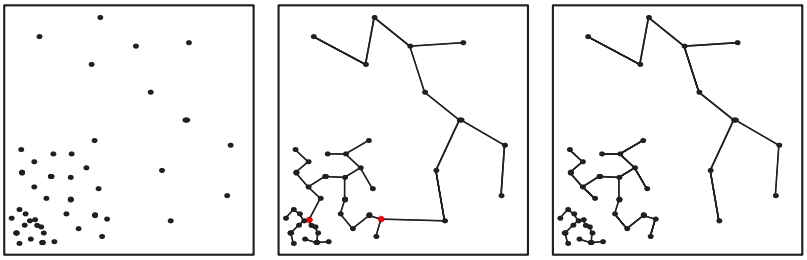


Figure 10.20: The removal of inconsistent edges — ones with length significantly larger than the average incident upon a node — may yield natural clusters. The original data is shown at the left and its minimal spanning tree is shown in the middle. At virtually every node, incident edges are of nearly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

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

One of the useful statistics that can be obtained from a minimal spanning tree is the edge length distribution. Figure 10.21 shows a situation in which two dense clusters are embedded in a sparse set of points; the lengths of the edges of the minimal spanning tree exhibit two distinct clusters 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.

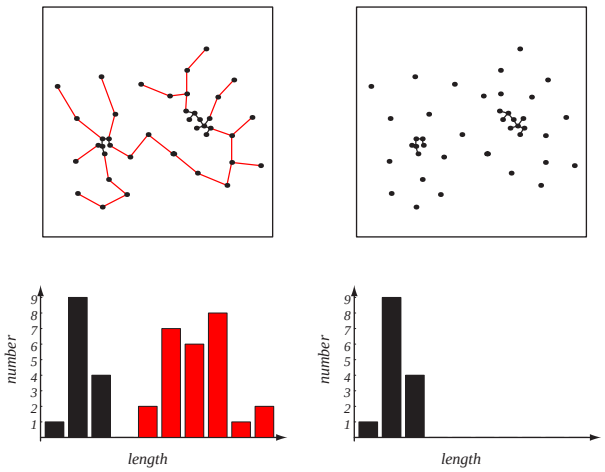



Figure 10.21: A minimal spanning tree is shown at the left; its bimodal edge length distribution is evident in the histogram below. If all links of intermediate or high length are removed (red), the two natural clusters are revealed (right).

## 10.13 Component analysis

Component analysis is an unsupervised approach to finding the "right" features from the data. We shall discuss two leading methods, each having a somewhat different goal. In principal component analysis (PCA), we seek to represent the $d$-dimensional data in a lower-dimensional space. This will reduce the degrees of freedom, reduce the space and time complexities. The goal is to represent data in a space that best describes the variation in a sum-squared error sense, as we shall see. In independent component analysis (ICA) we seek those directions that show the independence of signals. This method is particularly helpful for segmenting signals from multiple sources. As with standard clustering methods, it helps greatly if we know how many independent components exist ahead of time.

### 10.13.1 Principal component analysis (PCA)

KARHUNEN-LOÉVE TRANSFORM

The basic approach in principal componements or *Karhunen-Loéve transform* is conceptually quite simple. First, the $d$-dimensional mean vector $\boldsymbol{\mu}$ and $d \times d$ covariance matrix $\boldsymbol{\Sigma}$ are computed for the full data set. Next, the eigenvectors and eigenvalues are computed (cf. Appendix **??**), and sorted according to decreasing eigenvalue. Call these eigenvectors $\mathbf{e}_1$ with eigenvalue $\lambda_1$, $\mathbf{e}_2$ with eigenvalue $\lambda_2$, and so on. Next, the largest $k$ such eigenvectors are chosen. In practice, this is done by looking at a spectrum of eigenvectors. Often there will be xxx implying an inherent dimensionality of the subspace governing the "signal." The other dimensions are noise. Form a $k \times k$ matrix $\mathbf{A}$ whose columns consist of the $k$ eigenvectors.

Preprocess data according to:

$$\mathbf{x}' = \mathbf{A}^t(\mathbf{x} - \boldsymbol{\mu}). \tag{90}$$

It can be shown that this representation minimizes a squared error criterion (Problem 42).

### 10.13.2 Non-linear component analysis

We have just seen how to find a $k$-dimensional linear subspace of feature space that best represents the full data according to a minimum-square-error sense. If the data set is not well described by a sample mean and covariance matrix, but instead involves complicated interactions of features, then the linear subspace may be a poor representation. In such a case a non-linear component may be needed.

A neural network approach to such non-linear component analysis employs a network with five layers of units, as shown in Fig. 10.22. The middle layer consists of $k < d$ linear units, and it is here that the non-linear components will be revealed. It is important that the two other internal layers have nonlinear units (Problem 44).

AUTO-ENCODER

The entire network is trained using the techniques of Chapt. **??** as an *auto-encoder* or auto-associator. That is, each $d$-dimensional pattern is presented as input *and* as the target or desired output. When trained on a sum-squared error criterion, such a network readily learns the auto-encoder problem.

The top two layers of the trained network are discarded, and the rest used for non-linear components. For each input pattern $\mathbf{x}$, the output of the $k$ units of the three-layer network correspond to the non-linear components.

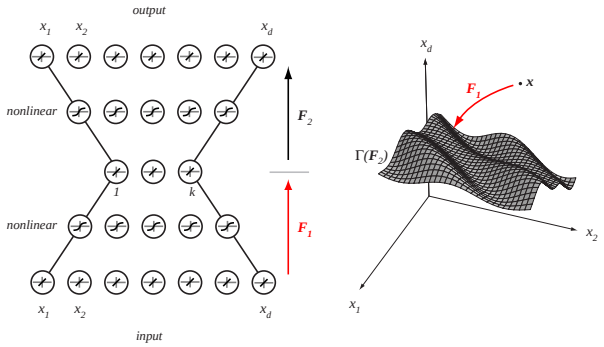



Figure 10.22: A five-layer neural network with two layers of non-linear units (e.g., sigmoidal), trained to be an auto-encoder, develops an internal representation that corresponds to the non-linear principal components of the full data set. (Bias units are not shown.) The process can be viewed in feature space (at the right). The transformation $\mathbf{F}_1$ is a non-linear projection onto a $k$-dimensional non-linear subspace denoted $\Gamma(\mathbf{F}_2)$. Points in $\Gamma(\mathbf{F}_2)$ are mapped via $\mathbf{F}_2$ back to the the $d$-dimensional space of the original data.

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

Learning in the original network is highly nonlinear, and during training care must be taken so as to avoid a poor local minimum (Chap. **??**). Naturally, one must take care to set an appropriate number $k$ of units. Recall that in (linear) principal component analysis, the number of components $k$ could be chosen based on the spectrum of eigenvectors. If the eigenvalues are ordered by magnitude, any significant drop between successive values indicates a "natural" number dimension to the subspace. Likewise, suppose five-layer networks are trained, with different numbers $k$ of units in the middle layer. Assuming poor local minima have been

avoided, the training error will surely decrease for successively larger values of $k$. If the improvement $k+1$ over $k$ is small, this may indicate that $k$ is the "natural" dimension of the nonlinear subspace.

We should not conclude that principal component analysis is always beneficial for classification. If the noise is large compared to the difference between categories, then component analysis will find the directions of the noise, rather than the signal, as illustrated in Fig. 10.23. In such cases, we seek to ignore the noise, and instead extract the directions that are indicative of the categories — a technique we consider next.

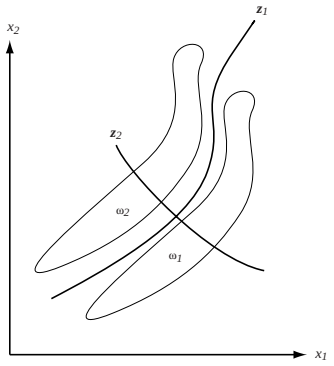



Figure 10.23: Features from two classes are as shown, along with nonlinear components of the full data set. Apparently, these classes are well separated along the $\mathbf{y}_2$ direction, but the large noise gives the largest nonlinear component to be $\mathbf{y}_1$. Preprocessing by keeping merely the largest nonlinear component would retain the "noise" and discard the "signal," giving poor recognition. The same defect arises in linear principal components, where the compoenents are linear and everywhere perpendicular.

### 10.13.3 *Independent component analysis (ICA)

Suppose there are $c$ independent scalar source signals $x_i(t)$ for $i = 1, \ldots c$ where we can consider $t$ to be a time index $1 \leq t \leq T$. For notational convenience we group the $c$ values at an instant into a vector $\mathbf{x}(t)$ and assume, further, that the vector has zero mean. Because of our independence assumption, and an assumption of no noise, we the multivariate density function can be written as

$$p(\mathbf{x}(t)) = \prod_{i=1}^{c} p(x_i(t)). \tag{91}$$

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

$$\mathbf{y}(t) = \mathbf{A}\mathbf{x}(t), \tag{92}$$

where $\mathbf{A}$ is a $c \times d$ scalar matrix, and below we shall require $d \geq c$.

The method is perhaps best illustrated in its most typical use. Suppose there are $c$ sound sources being sensed by $d$ microphones, all in a room. Each microphone gets a mixture of the sources, with amplitudes depending upon the distances (Fig. 10.24). (We shall ignore any effects of delays.)

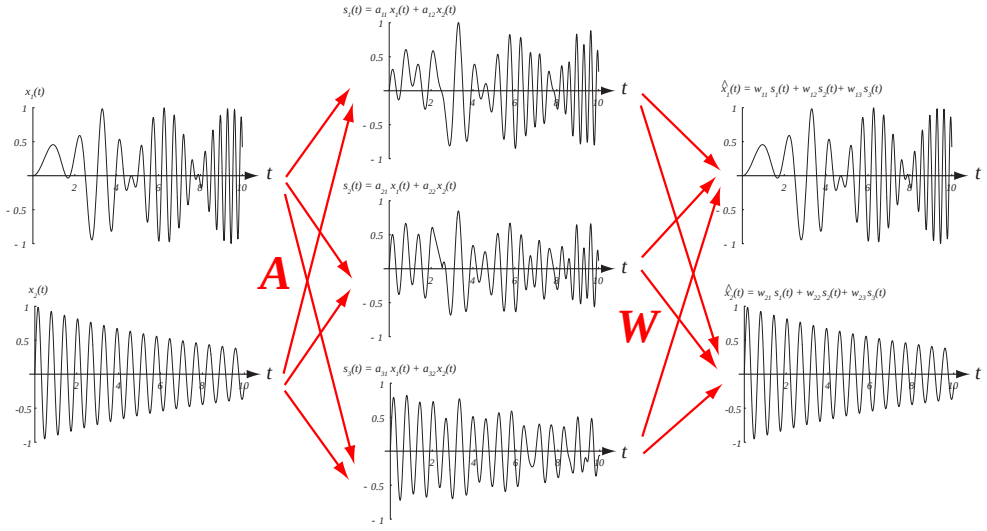



Figure 10.24: Independent component analysis (ICA) is an unsupervised method that can be applied to the problem of blind source separation. In such problems, two or more source signals (assumed independent) $x_1(t)$, $x_2(t)$, $\ldots$, $x_d(t)$ are combined to yield a sum signal, $s_1(t)+s_2(t)+\ldots+s_c(t)$ where $c \geq d$. (This figure illustrates a case with only two components.) Given merely the linear signals, and the assumption of the number of components, $d$, the task of ICA is to recover the source signals. This is equivalent to finding a matrix $\mathbf{W}$ that is the inverse of $\mathbf{A}$. In general appalications of ICA, one seeks to extraction independent components from the sensed signals, whether or not they arose from a linear mixture of initial sources.

The task of independent component analysis is to recover the source signals from the sensed signals. More specifically, we seek a real matrix $\mathbf{W}$ such that

$$\mathbf{z}(t) = \mathbf{W}\mathbf{y}(t) = \mathbf{W}\mathbf{A}\mathbf{x}(t), \tag{93}$$

where $\mathbf{z}$ is an estimate of the sources $\mathbf{x}(t)$. Of couse we seek $\mathbf{W} = \mathbf{A}^{-1}$, but neither $\mathbf{A}$ nor its inverse are known.

We approach the determination of $\mathbf{A}$ by maximum-likelihood techniques. We use an estimate of the density, parameterized by $\mathbf{a}$ $\hat{p}(\mathbf{y};\ \mathbf{a})$ and seek the parameter vector $\mathbf{a}$ that minimizes the diffrerence between the source distribution and the estimate. That is, $\mathbf{a}$ is the basis vectors of $\mathbf{A}$ and thus $\hat{p}(\mathbf{y};\ \mathbf{a})$ is an estimate of the $p(\mathbf{y})$.

This difference can be quantified by the Kullback-Liebler divergence:

$$\begin{aligned} D(p(\mathbf{y}), \hat{p}(\mathbf{y};\ \mathbf{a})) &= D(p(\mathbf{y})||\hat{p}(\mathbf{y};\ \mathbf{a})) \\ &= \int p(\mathbf{y}) log \frac{p(\mathbf{y})}{\hat{p}(\mathbf{y};\ \mathbf{a})} d\mathbf{y} \\ &= H(\mathbf{y}) - \int p(\mathbf{y}) log \hat{p}(\mathbf{y};\ \mathbf{a}) d\mathbf{y} \end{aligned} \tag{94}$$

The log-likelihood is

$$l(\mathbf{a}) = \frac{1}{n} \sum_{i=1}^{n} log \hat{p}(\mathbf{x}_i;\ \mathbf{a}). \tag{95}$$

and using the law of large numbers, the Kullback-Liebler divergence can be written as

$$\begin{aligned} l(\mathbf{a}) &= -\int p(\mathbf{y}) log p(\mathbf{y}) d\mathbf{y} - \int p(\mathbf{y}) log \frac{p(\mathbf{y})}{\hat{p}(\mathbf{y};\ \mathbf{a})} d\mathbf{y} \\ &= \underbrace{H(\mathbf{y})}_{indep.\ of\ \mathbf{W}} - D(p(\mathbf{y})||\hat{p}(\mathbf{y};\ \mathbf{a})), \end{aligned} \tag{96}$$

where the entropy $H(\mathbf{y})$ is independent of $\mathbf{W}$. Thus we maximize the log-likelihood by minimizing the Kullback-Liebler divergence with respect to the estimated density $\hat{p}(\mathbf{y};\ \mathbf{a})$:

$$\frac{\partial l(\mathbf{a})}{\partial \mathbf{W}} = -\frac{\partial}{\partial \mathbf{W}} D(p(\mathbf{y})||\hat{p}(\mathbf{y};\ \mathbf{a})). \tag{97}$$

Because $\mathbf{A}$ is an invertible matrix, and because the Kullback-Liebler divergence is invariant under invertible transformation (Problem 47), we have

$$\frac{\partial l(\mathbf{a})}{\partial \mathbf{W}} = -\frac{\partial}{\partial \mathbf{W}} D(p(\mathbf{x})||\hat{p}(\mathbf{z})). \tag{98}$$

$$\begin{aligned} \frac{\partial H(\mathbf{yyy})}{\partial \mathbf{WWW}} &= \frac{\partial}{\partial \mathbf{WWW}} log[|\mathbf{WWW}|] + \frac{\partial}{\partial \mathbf{WWW}} log\left[\prod_{i=1}^{n} \left|\frac{\partial xx_i}{\partial yy_i}\right|\right] \\ &= [\mathbf{WWW}^{-1}]^t - \phi(\mathbf{xxx})\mathbf{zzz}^t, \end{aligned} \tag{99}$$

where $\phi(\mathbf{xxx})$ is the *score function*, the gradient fector of the log likelihood: SCORE FUNCTION

$$\phi(\mathbf{z}) = -\frac{\partial p(\mathbf{z})/\partial \mathbf{z}}{p(\mathbf{z})} = -\begin{pmatrix} \frac{\partial p(z_1)/\partial z_1}{p(z_1)} \\ \vdots \\ \frac{\partial p(z_q)/\partial z_q}{p(z_q)} \end{pmatrix} \tag{100}$$

Thus the learning rule is

$$\frac{\partial H(\mathbf{xxx})}{\partial \mathbf{xxx}} = [\mathbf{xxx}^t]^{-1} - \phi(\mathbf{xx})\mathbf{yy}^t. \tag{101}$$

A simpler form comes if we merely scale, following the natural gradient

$$\Delta \mathbf{xxx} \propto \frac{\partial H(\mathbf{xxx})}{\partial \mathbf{xxx}} \mathbf{WW}^t\mathbf{WW} = [\mathbf{I} - \phi(\mathbf{xx})\mathbf{xx}^t]\mathbf{WWW}. \tag{102}$$

This, then is the learning algorithm.

An assumption is that at most one of the sources is Gaussian distributed (Problem 46). Indeed this method is most successful if the distributions are highly skewed or otherwise deviate markedly from Gaussian.

We can understand the difference between PCA and ICA in the following way. Imagine that there were two sources that are correlated and large correlated signals in a particular direction. PCA would find that direction, and indeed would reduce the sum-squared error. Such components are not independent, and would not be useful for separating the sources. As such, they would not be found by ICA. Instead, ICA

would find those directions that are best for separating the sources — even if those directions have small eigenvectors.

Generally speaking, when used as preprocessing for *classification*, independent component analysis has several characteristics that make it more desirable than linear or non-linear principal component analysis. As we saw in Fig. 10.23, such principal components need not be effective in separating classes. Recall that the sensed input consists of a signal (due to the true categories) plus noise. If the noise is large much larger than the signal, principal components will depend more upon the noise than on the signal. Since the different categories are, we assume, independent, independent component analysis is likely to extract those features that are useful in distinguishing the classes.

## 10.14 Low-Dimensional Representations and Multidimensional Scaling (MDS)

Part of the problem of deciding whether or not a given clustering means anything stems from our inability to visualize the structure of multidimensional data. This problem is further aggravated when similarity or dissimilarity measures are used that lack the familiar properties of distance. One way to attack this problem is to try to represent the data points as points in some lower-dimensional space in such a way that the distances between points in 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 $\mathbf{x}_1, \ldots, \mathbf{x}_n$. Let $\mathbf{y}_i$ be the lower-dimensional *image* of $\mathbf{x}_i$, $\delta_{ij}$ 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. 10.25). Then we are looking for a *configuration* of image points $\mathbf{y}_1, \ldots, \mathbf{y}_n$ for which the $n(n-1)/2$ distances $d_{ij}$ between image points are as close as possible to the corresponding original distances $\delta_{ij}$. 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:

$$J_{ee} = \frac{\sum_{i<j} (d_{ij} - \delta_{ij})^2}{\sum_{i<j} \delta_{ij}^2} \tag{103}$$

$$J_{ff} = \sum_{i<j} \left( \frac{d_{ij} - \delta_{ij}}{\delta_{ij}} \right)^2 \tag{104}$$

$$J_{ef} = \frac{1}{\sum_{i<j} \delta_{ij}} \sum_{i<j} \frac{(d_{ij} - \delta_{ij})^2}{\delta_{ij}}. \tag{105}$$

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

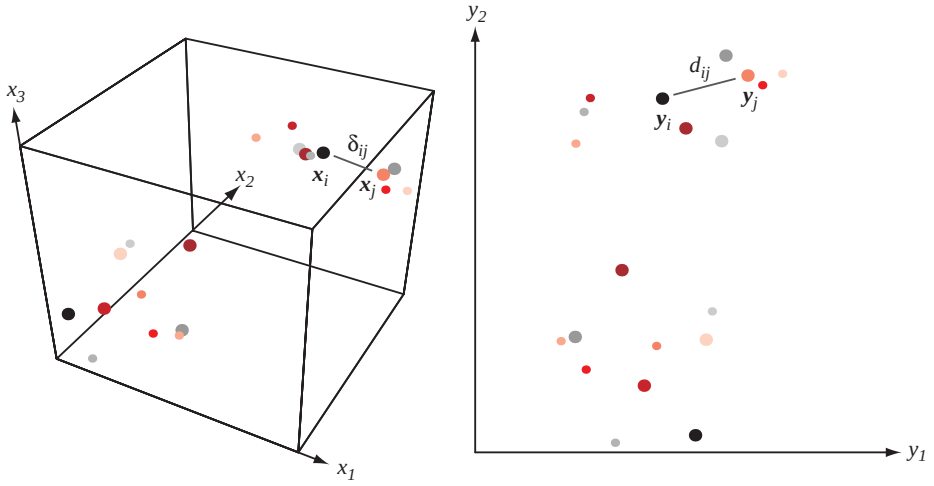



Figure 10.25: The distance between points in the original space are $\delta_{ij}$ while in the projected space $d_{ij}$. In practice, the source space is typically of very high dimension, and the mapped space of just two or three dimensions, to aid visualization. (In order to illustrate the correspondence between points in the two spaces, the size and color of each point $\mathbf{x}_i$ matches that of its image $\mathbf{y}_i$.

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 $\delta_{ij}$ are large or small), $J_{ff}$ emphasizes the largest fractional errors (regardless whether the errors $|d_{ij} - \delta_{ij}|$ 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 $\mathbf{y}_1, \ldots, \mathbf{y}_n$ 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

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

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

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

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 spiral in three-dimensions:

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

Figure 10.26 shows a the three-dimensional data. When the $J_{ef}$ criterion was used, twenty iterations of a gradient descent procedure produced the two-dimensional configuration shown at the right. Of course, translations, rotations, and reflections of this configuration would be equally good solutions.

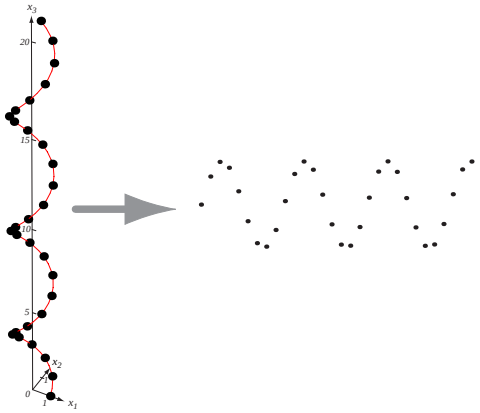



Figure 10.26: Thirty points of the form $(\cos(k/\sqrt{2}), \sin(k/\sqrt{2}), k/\sqrt{2})^t$ for $k = 0, 1, \ldots, 29$ are shown at the left. Multidimensional scaling using the $J_{ef}$ criterion (Eq. 105) and a two-dimensional target space leads to the image points shown at the right. This lower-dimensional representation shows clearly the fundamental sequential nature of the points in the original, source space.

In non-metric multidimensional scaling problems, the quantities $\delta_{ij}$ 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 $\delta_{ij}$. Let us order the $m = n(n-1)/2$ dissimilarities so that $\delta_{i_1 j_1} \leq \cdots \leq \delta_{i_m j_m}$, and let $\hat{d}_{ij}$ be *any* $m$ numbers satisfying the *monotonicity constraint*

MONOTONICITY CONSTRAINT

$$\hat{d}_{i_1 j_1} \leq \hat{d}_{i_2 j_2} \leq \cdots \leq \hat{d}_{i_m j_m}. \tag{106}$$

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

$$\hat{J}_{mon} = \min_{\hat{d}_{ij}} \sum_{i<j} (d_{ij} - \hat{d}_{ij})^2, \tag{107}$$

where it is always to be understood that the $\hat{d}_{ij}$ must satisfy the monotonicity constraint. Thus, $\hat{J}_{mon}$ measures the degree to which the configuration of points $\mathbf{y}_1, \ldots, \mathbf{y}_n$ represents the original 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:

$$J_{mon} = \frac{\hat{J}_{mon}}{\sum_{i<j} d_{ij}^2}. \tag{108}$$

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

### 10.14.1 Self-organizing feature maps

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

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, as shown in Fig. 10.27. The source space is sensed by a movable two-joint arm of fixed segment lengths; thus each point $(x_1, x_2)$ in the disk area leads to a pair of angles $(\phi_1, \phi_2)$, which we denote as a vector $\boldsymbol{\phi}$. The algorithm uses a sequence of $\boldsymbol{\phi}$ values but not the $(x_1, x_2)$ values themselves, since they and their nonlinear transformation are not directly accessible. In our illustration the nonlinearity involves inverse trigonometric functions, but in most applications it is more complicated and not even known.

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

The mapping is learned by a simple two-layer neural network, here with two inputs ($\phi_1$ and $\phi_2$), fully connected to a large number of outputs, corresponding to points

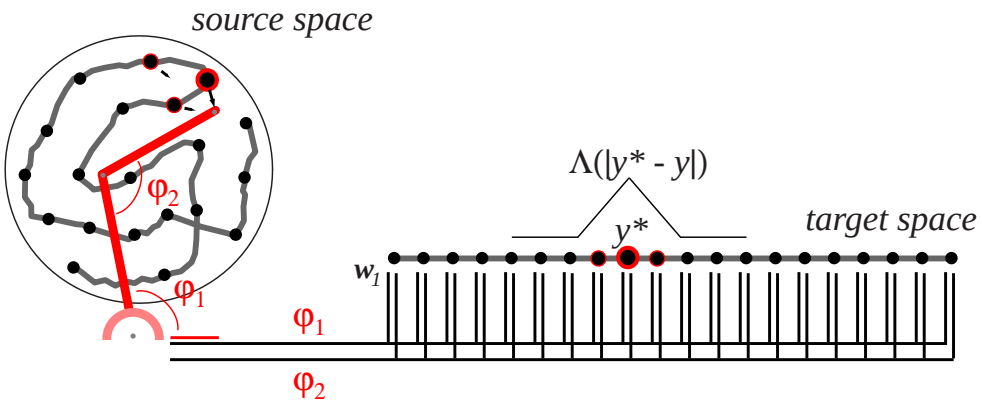



Figure 10.27: A self-organizing map from the (two-dimensional) disk source space to the (one-dimensional) line of the target space can be learned as follows. For each point $\mathbf{x}$ in the target line, there exists a corresponding point in the source space that, if sensed, would lead to $\mathbf{x}$ begin most active. For clarity, then, we can link theses points in the source; it is as if the image line is placed in the source space. At the state shown, the particular sensed point leads to $\mathbf{x}^*$ begin most active. The learning rule (Eq. 109) makes its source point move toward the sensed point, as shown by the small arrow. Because of the window function $\Lambda(|\mathbf{y}^* - \mathbf{y}|)$, points adjacent to $\mathbf{x}^*$ are also moved toward the sensed point, thought not as much. If such learning is repeated many times as the arm randomly senses the whole source space, a topologically correct map is learned.

along the target line. 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; call it $\mathbf{y}^*$. The weights to this unit and those in its immediate neighborhood are updated according to:

$$w_{ki}(t+1) = w_{ki}(t) + \eta(t)\Lambda(|\mathbf{y} - \mathbf{y}^*|)\boldsymbol{\phi}_i, \tag{109}$$

where $\eta(t)$ is a learning rate which depends upon the iteration number $t$. Next, every weight vector is normalized such that $|\mathbf{w}| = 1$. (Naturally, only those weight vectors that have been altered during the learning trial need be re-normalized.) The function $\Lambda(|\mathbf{y}-\mathbf{y}^*|)$ is called the "window function," and has value 1.0 for $\mathbf{y} = \mathbf{y}^*$ and smaller for large values of $|\mathbf{y}-\mathbf{y}^*|$. The window function is vital to the success of the algorithm: it insures that neighboring points in the target space have weights that are similar, and thus correspond to neighboring points in the source space, thereby insuring topological neighborhoods (Fig. 10.28). The learning rate $\eta(t)$ decreases slowly as a function of iteration number (i.e., as patterns are presented) to insure that learning will ultimately stop.

WINDOW FUNCTION

Equation 109 has a particularly straightforward interpretation. For each pattern presentation, the "winning" unit in the target space ($\mathbf{y}^*$) is adjusted so that it is more like the particular pattern. Others in the neighborhood of $\mathbf{y}^*$ are also adjusted so that their weights more nearly match that of the input pattern (though not quite as much as for $\mathbf{y}^*$, according to the window function). In this way, neighboring points in the input space lead to neighboring points being active.

After are large number of pattern presentations, learning according to Eq. 109

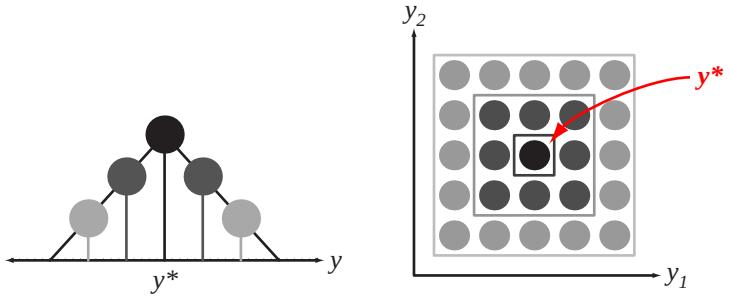



Figure 10.28: Typical window functions for self-organizing maps for target spaces in one dimension (left) and two dimensions (right). In each case, the weights at the maximally active unit, $\mathbf{y}*$, in the target space get the largest weight update while units more distant get smaller update.

insures that neighboring points in the source space lead to neighboring points in the target space. Informally speaking, it is as if the target space line has been placed on the source space, and learning pulls and stretches the line to fill the source space, as illustrated in Fig. 10.29 shows the development of the map. After 150000 training presentations, a topological map has been learned.

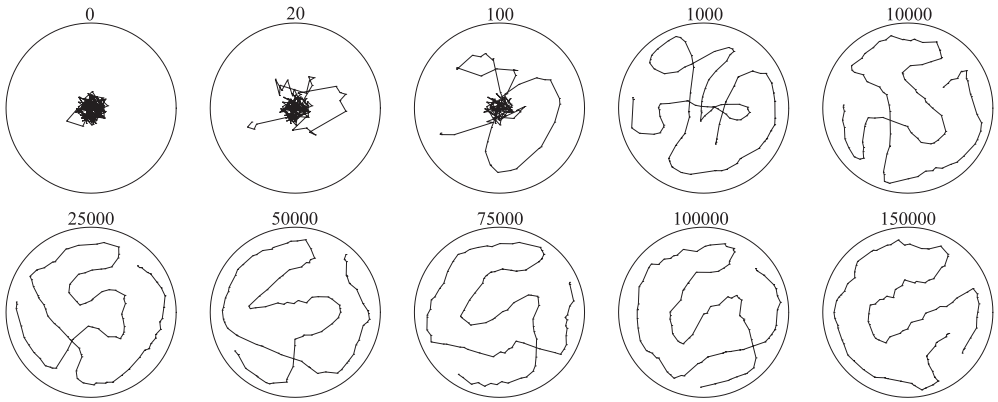



Figure 10.29: If a large number of pattern presentations are made using the setup of Fig. 10.27, a topologically ordered map develops. The number of pattern presentations is listed.

The learning of such self-organizing maps is very general, and can be applied to virtually any source space, target space and continuous nonlinear mapping. Figure 10.30 shows the development of a self-organizing map from a square source space to a square (grid) target space.

There are generally inherent ambiguities in the maps learned by this algorithm. For instance, a mapping from a square to a square could eight possible orientations, corresponding to the four rotation and two flip symmetries. Such ambiguity is generally irrelevant for subsequent clustering or classification in the target space. Nev-

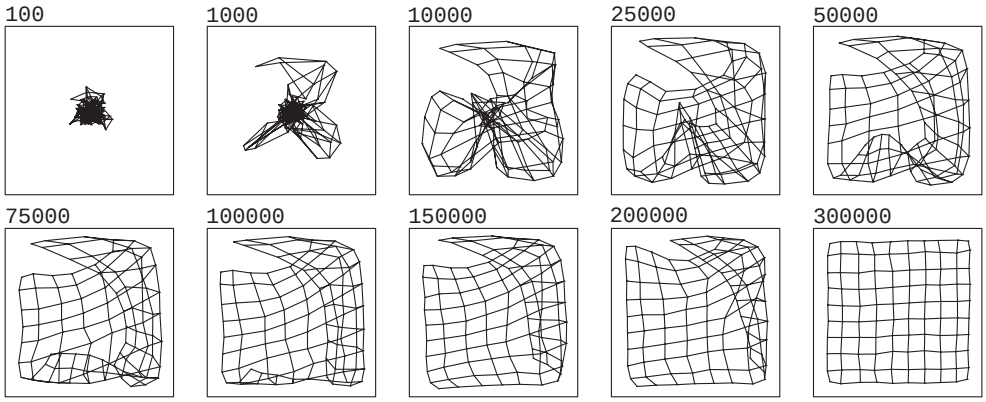



Figure 10.30: A self-organizing feature map from a square source space to a square (grid) target space. As in Fig. 10.27, each grid point of the target space is shown atop the the point in the source space that leads maximally excites that target point. This example also used the non-linear

ertheless the mapping ambiguities are related to a more significant drawback — the possibility of "kinks" in the map. A particular initial condition can lead to *part* of the map learning one of the orientations, while a different part learns another one (Fig. 10.31). When this occurs, it is generally best to re-initialize the weights randomly and restart the learning with perhaps a wider window function or slower decay in the learning rate.

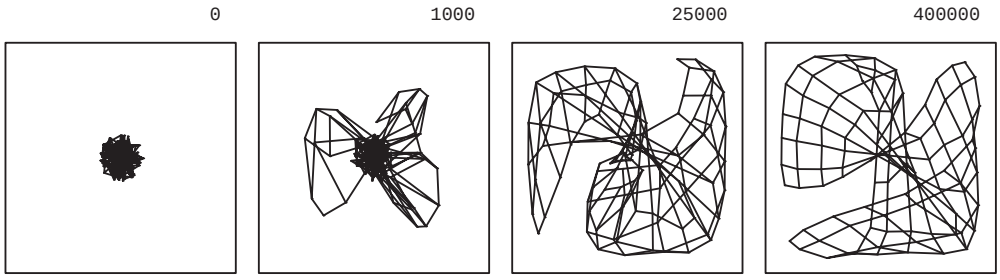



Figure 10.31: Some initial (random) weights and the particular sequence of patterns (randomly chosen) lead to kinks in the map; even extensive further training does not eliminate the kink. In such cases, learning should be re-started with randomized weights and possibly a wider window function and slower decay in learning.

One of the benefits of this learning algorithm is that it naturally takes account of the probability of sampling in the source space, i.e., $p(\mathbf{x})$. Regions of high such probability attract more of the points in the target space, and this yields xxx, as shown in Fig. 10.32. Thus in the target space, xxx points are spread apart — just as we would want for preprocessing for subsequent classification.

Another issue is the number of dimensions in the target space. One typically chooses this dimension (and

run in unsupervised mode — track slow changes.

Such self-organizing feature maps can be used in a number of systems. For in-

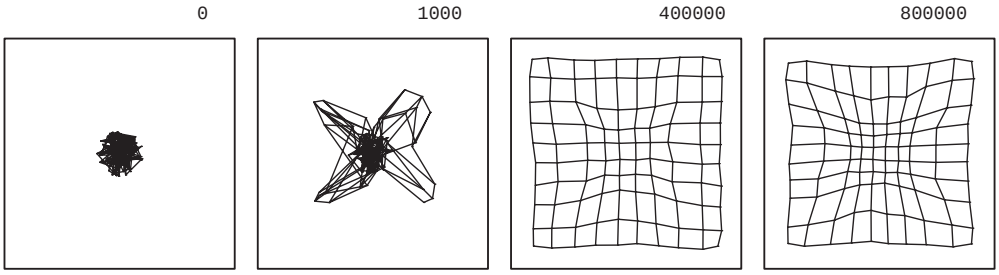



Figure 10.32: Uneven density: 20 times more likely to choose a point in the center (density is 20 times greater).

stance, 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.

### 10.14.2 Clustering and Dimensionality Reduction

Because the curse of dimensionality plagues so many pattern recognition procedures, a variety of methods for dimensionality reduction have been proposed. Unlike the procedures that we have just examined, most of these methods provide a functional mapping, so that one can determine the image of an arbitrary feature vector. The classical procedures of statistics are *principal components 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 COMPONENT

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 $\rho_{ij}$ is related to the covariances (or sample covariances) by

CORRELATION MATRIX

$$\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}. \tag{110}$$

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, $\rho_{ij}^2$ plays the role of a similarity function for features. Two features

for which $\rho_{ij}^2$ is large are clearly good candidates to be merged into one feature, thereby reducing the dimensionality by one. Repetition of this process leads to the following hierarchical procedure:

**Algorithm 8 (Hierarchical dimensionality reduction)**

```
1  begin initialize d', D_i ← {x_i}, i = 1, ..., d
2        d̂ ← d + 1
3        do d̂ ← d̂ − 1
4           compute R by Eq. 110
5           Find most correlated distinct clusters, say D_i and D_j
6           D_i ← D_i ∪ D_j merge
7           delete D_j
8        until d̂ = d'
9    return d' clusters
10 end
```

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

For the purposes of pattern *classification*, the most serious criticism of all of the approaches to dimensionality reduction that we have mentioned is that they are overly concerned with faithful *representation* of the data. Greatest emphasis is usually placed on those features or groups of features that have the greatest variability. But for classification, we are interested in *discrimination* — not representation. While it is a truism that the ideal representation is the one that makes classification easy, it is not always so clear that clustering without explicitly incorporating classification criteria will find such a representation. 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. When it is possible to exploit knowledge of the problem domain to obtain more informative features, that is usually the most profitable course of action.

## Summary

Unsupervised learning and clustering seek to extract information from unlabeled samples. If the underlying distribution comes from a mixture of component densities de-

scribed by a set of unknown parameters $\boldsymbol{\theta}$, then $\boldsymbol{\theta}$ can be estimated by Bayesian or maximum-likelihood methods. A more general approach is to define some measure of similarity between two clusters, as well as a global criterion such as a sum-squared-error or trace of a scatter matrix. Since there are only occasionally analytic methods for computing the clustering which optimizes the criterion, a number of greedy (*locally* step-wise optimal) iterative algorithms can be used, such as k-means and fuzzy k-means clustering.

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

Competitive learning is an on-line neural network clustering algorithm in which the cluster center most similar to an input pattern is modified to become more like that pattern. In order to guarantee that learning stops for an arbitrary data set, the learning rate must decay. Competitive learning can be modified to allow for the creation of new cluster centers, if no center is sufficiently similar to a particular input pattern, as in leader-follower clustering and Adaptive Resonance. While these methods have many advantages, such as computational ease and tracking gradual variations in the data, they rarely optimize an easily specified global criterion such as sum-of-squared error.

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

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

Two general methods for dimensionality reduction is self-organizing feature maps and multidimensional scaling. Self-organizaing feature maps can be highly nonlinear, and represents points close in the source space by points close in the lower-dimensional target space. In preserving neighborhoods in this way, such maps also called "topologically correct." The source and target spaces can be of very general shapes, and the

mapping will depend upon the the distribution of samples within the source space. Multidimensional scaling similarly learns a nonlinear mapping that, too, seeks to preserve neighborhoods, and is often used for data visualization. Because the basic method requires all the inter-point distances for minimizing a global criterion function, its space complexity limits the usefulness of multidimensional scaling to problems of moderate size.

## Bibliographical and Historical Remarks

Historically, the literature on unsupervised learning and clustering dates to Karl Pearson, who in 1894 used sample moments to determine the parameters in a misture of two univariate Gaussians. While most books on pattern classification address unsupervised learning, there are several modern books[21, 1] and review articles on unsupervised learning that go into great detail. Much of the work on unsupervised methods comes from the signal compression community, where vector quantization (VQ) seeks to represent an arbitrary vector by one of $c$ vectors prototype vectors corresponding to our clusters [17].

A clear book on mixture models is [29]. The issue of identifiability in unsupervised learning is [37]. Hasselblad showed how the parameters of one-dimensional normals could be learned in an unsupervised environment [19]. The k-means algorithm was introduced in a paper by Lloyd [28], which inspired many variations (including fuzzy" ones [4, 5]) and computational improvements.

Efficient agglomerative methods for hierarchical clustering are summarized in [10].

The key mathematical concepts underlying principal component analysis appear in [22] as well as [7, 26, 11], which stress neural implementation. Independent component analysis was introduced by Jutten and Herault[23], and the maximum-likelihood approach introduced by Gaeta and Lacoume[15] Generalizations and a maximum-likelihood approach are given in [32]. Bell and Sejnowski [3] showed a neural network. A good compendium is [38]. Another Perlmutter paper [31]. Several studies have shown the benefits of ICA for classification [13].

Multidimensional scaling discussed in [34, 6] and its relationship to clustering is explored in [27].

The classificatory foundations of biology, cladistics (from the Greek *klados*, branch) provide useful background for the use of classification in all scientific fields [14].

Kohonen's long series of papers on self-organizing feature maps began in the early 1980s [24] and a good compendium can be found in [25]; convergence properties of algorithms for self-organizing feature maps are proved in [39]. There have been numerous applications of the method, from speech to finding patterns of poverty in the world.

Also goes under the name Learning Vector Quantization (LVQ).

The main emphasis of research on Adaptive Resonance has been to explore [8, Chapter 10] A wonderfully clear exposition of the central algorithmic ideas is [30]; an attempt to translate the ideas and terminology of adaptive resonance, including a glossary, is given in [36].

## Problems

⊕ *Section 10.2*

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

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

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

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

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

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

**2.** Consider a mixture distribution of two triangle distributions, where component density $\omega_i$ is centered on $\mu_i$ and has "halfwidth" $w_i$, according to:

$$p(x|\omega_i) \sim T(\mu_i, w_i) = \begin{cases} (1 - |x - \mu_1|)/(2w_i) & \text{for } |x - \mu_i| < w_i \\ 0 & \text{otherwise.} \end{cases}$$

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

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

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

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

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

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

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

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

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

(c) Suppose $\sigma_1$ and $\sigma_2$ are known, but $P(\omega_1)$ is unknown. Is this resulting model identifiable? That is, can $P(\omega_1)$ be identified using data?

⊕ *Section 10.3*

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

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

where

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

(a) Derive the formula for the partial derivative:

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

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

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

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

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

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

where $\hat{\boldsymbol{\mu}}_i$ and $\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}_i)$ are given by Eqs. 20 & 22, respectively.

**6.** 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 $\alpha$ is a parameter that appears in *several* (and possibly all) of the component densities. Let $l$ be the $n$-sample log-likelihood function, and show that

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

where

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

**7.** Let $\theta_1$ and $\theta_2$ be unknown parameters for the component densities $p(x|\omega_1, \theta_1)$ and $p(x|\omega_2, \theta_2)$, respectively. Assume that $\theta_1$ and $\theta_2$ 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?

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

**9.** Assume the likelihood function of Eq. 3 is differentiable and derive the maximum likelihood conditions of Eqs. 11 – 13.

⨁ *Section 10.4*

**10.** 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 $\sigma_{pq}$ be the $pq$th element of $\boldsymbol{\Sigma}$, $\sigma^{pq}$ be the $pq$th element of $\boldsymbol{\Sigma}^{-1}$, $x_p(k)$ be the $p$th element of $\mathbf{x}_k$, and $\mu_p(i)$ be the $p$th element of $\boldsymbol{\mu}_i$.

(a) Show that

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

where

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

(b) Use this result and the results of Problem 6 to show that the maximum-likelihood estimate for $\boldsymbol{\Sigma}$ must satisfy

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

where $\hat{P}(\omega_i)$ and $\hat{\boldsymbol{\mu}}_i$ are the maximum-likelihood estimates given by Eqs. 19 & 20.

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

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

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

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

(c) Summarize and interpret your answer in words.

**12.** Consider the univariate normal mixture

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

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

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

ought to be approximately

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

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

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

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

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

**14.** Perform the differentiation in Eq. 26 to derive Eqs. 27 & 28.

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

**16.** Fill in the steps of the derivation of Eqs. 19 – 21.

⊕ *Section 10.5*

**17.** Consider the combinatorics of exhaustive inspection of clusters of $n$ samples into $c$ clusters.

(a) Show that there are exactly

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

such distinct clusterings.

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

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

⊕ *Section 10.6*

**18.** 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 $k-1$. Explain why with both $\delta_{min}$ and $\delta_{max}$ 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 $0 = v_1 \leq v_2 \leq v_3 \leq \cdots \leq v_n$.

⊕ *Section 10.7*

**19.** Derive Eq. 50 from Eq. 49 using the definition given in Eq. 51.

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

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

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

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

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

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

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

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

Show that by the sum-of-square error $J_e$ criterion (Eq. **??**), the third partition is favored, whereas by the invariant $J_d$ (Eq. 63) criterion the first two partitions are favored.

**23.** Let $\mathbf{x}_1 = \binom{xx}{xx}$, $\mathbf{x}_2 = \binom{xx}{xx}$, $\mathbf{x}_3 = \binom{xx}{xx}$, and $\mathbf{x}_4 = \binom{xx}{xx}$, and consider the following three partitions:

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

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

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

(b) Find the clustering that minimizes the trace criterion, $J_d$ (Eq. 63).

**24.** Consider the problem of invariance to transformation of the feature space.

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

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

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

**25.** Recall the definitions of the within-cluster and the between-cluster scatter matrices (Eqs. 57 & 58). Define the total scatter matrix to be $\mathbf{S}_T = \mathbf{S}_W + \mathbf{S}_B$. Show that the following measures (Eqs. 65 & 66) are invariant to linear transformations of the data.

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

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

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

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

**26.** Show that the clustering criterion $J_d$ in Eq. 63 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.

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

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

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

⊕ *Section 10.8*

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

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

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

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

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

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

(c) Using this result, write pseudocode for an iterative procedure for minimizing $J_T$ (cf. Computer Exercise 20).

**29.** Consider how the transfer of a single point from one cluster to another affects the mean and sum-squared error, and thereby derive Eqs. 71 & 72.

⊕ *Section 10.9*

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

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

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

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

**31.** Let $d$ be the dimensionality of the space, $q$ a scalar parameter ($q > 1$). For each of the measures shown, state whether it represents a metric (or not), and whether it represents an ultrametric (or not).

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

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

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

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

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

(f) $s(\mathbf{x}, \mathbf{x}') = \min_{\boldsymbol{\alpha}} \|\mathbf{x} + \boldsymbol{\alpha}\mathbf{T}(\mathbf{x}) - \mathbf{x}'\|^2$ (one-sided tangent distance) where $\mathbf{T}$ is a linear transform and $\boldsymbol{\alpha}$ a vector of coefficients (cf. Chap. **??**, Sect. **??**).

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

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

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

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

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

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

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

**33.** 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 $i$th cluster contains $n_i$ samples with sample mean $\mathbf{m}_i$, show that the smallest increase results from merging the pair of clusters for which

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

is minimum.

**34.** Assume we are clustering using the sum-of-squared error criterion $J_e$ (Eq. **??**). Show that a "distance" measure between clusters can be derived, Eq. 78, such that merging the "closest" such clusters increases $J_e$ as little as possible.

**35.** Create by hand a dendrogram for the following eight points in one dimension: $\{-5.5, -4.1, -3.0, -2.6, 10.1, 11.9, 12.3, 13.6\}$. Define the similarity between to clusters to be $20 - d_{min}(\mathcal{D}_i, \mathcal{D}_j)$, where $d_{min}(\mathcal{D}_i, \mathcal{D}_j)$ is given in Eq. 74. Based on your dendrogram, argue that two is the natural number of clusters.

**36.** Create by hand a dendrogram for the following 10 points in one dimension: $\{-2.2, -2.0, -0.3, 0.1, 0.2, 0.4, 1.6, 1.7, 1.9, 2.0\}$. Define the similarity between to clusters to be $20 - d_{min}(\mathcal{D}_i, \mathcal{D}_j)$, where $d_{min}(\mathcal{D}_i, \mathcal{D}_j)$ is given in Eq. 74. Based on your dendrogram, argue that three is the natural number of clusters.

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

⊕ *Section 10.10*

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

(a) Prove that the distribution of the criterion function $J_e(1)$ given in Eq. 82 is normal with mean $nd\sigma^2$. Express $\sigma$ in terms of $\mathbf{\Sigma}$.

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

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

**39.** Derive Eqs. 85 & 86.

⊕ *Section 10.12*

**40.** Consider a simple greedy algorithm for creating a spanning tree.

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

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

(c) What is the average time complexity?

⊕ *Section 10.11*

**41.** Consider the adaptive resonance clustering algorithm.

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

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

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

⊕ *Section 10.13*

**42.** Show that minimizing a mean-squared error criterion for $d$-dimensional data leads to the $k$-dimensional representation ($k < d$) of the Karhunen-Loéve transform (Eq. 90) as follows. For simplicity, assume that the data set has zero mean. (If the mean is not zero, we can always subtract off the mean from each vector to define a new vectors.)

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

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

(c) Consider small variations $\delta\mathbf{e}$ that do not change the length of the vector, i.e., ones in which $\delta\mathbf{e}$ is perpendicular to $\mathbf{e}$. Use this condition and your above results to show that $(\delta\mathbf{e})^t\mathbf{\Sigma}\lambda(\delta\mathbf{e})^t\mathbf{e} = 0$, where $\lambda$ is a scalar. Show that the necessary and sufficient solution is $\mathbf{\Sigma}\mathbf{e} = \lambda\mathbf{e}$ — that is, the eigenvector equation of Eq. 99.

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

**43.** Show that a neural net auto-association network consisting of $d - k - d$ input, hidden and output layer (with $k < d$)

(a) Show that a neural net auto-association network consisting of $d - k - d$ input, hidden and output layer (with $k < d$) and linear hidden units performs principal component analysis by considering the minimization it solves. Trained on sum squared error.

(b) Show that a neural net auto-association network consisting of $d - k - d$ input, hidden and output layer (with $k < d$)

(c) Show that the five layer neural net auto-association network of Fig. **??** consisting of $d - k - r - k - d$ where both layers having $k$ units are nonlinear will perform nonlinear dimensionality reduction.

**44.** Consider the use of neural networks for nonlinear principal componet analysis.

(a) Prove that if all units in the five-layer network of Fig. 10.22 are linear, and the network trained to serve as an auto-encoder, then the representation learned at the middle layer corresponds to the linear principal component of the data.

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

**45.** The derivation of the Independent component analysis algorithm, summarized in Eq. 99, assumed that the sources and sum signals were all scalars, that there was no noise, and that the number of observations, $T$, is equal to the number of points generated by each source.

(a) Relax all of these conditions to generalize the method to vectors, $\mathbf{x}_1(t) + \ldots + \mathbf{x}_c(t)$. Assume, moreover, that the sum signal is corrupted by additive Gaussian noise of zero mean, but unknown covariance: $p(\mathbf{y}) \sim N(0, \mathbf{\Sigma})$.

(b) Suppose the noise is sufficiently small ($|\mathbf{\Sigma}| \ll 1$), and that the dimensionality of the vectors is set to $d = 1$. Show that your learning rule reduces to that of Eq. 99.

**46.** Use the fact that the sum samples from two Gaussians is again a Gaussian to show why independent component analysis can not isolate sources perfectly if more than one has a Gaussian distribution.

**47.** It is a fact that the Kullback-Liebler divergence is invariant under general invertible transforms. Prove this for the special case of linear transforms, as used in Sect. 10.13.

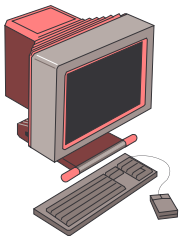
⊕ *Section 10.14*

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

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

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

## Computer exercises



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

| sample | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|
| 1 | -7.82 | -4.58 | -3.97 |
| 2 | -6.68 | 3.16 | 2.71 |
| 3 | 4.36 | -2.19 | 2.09 |
| 4 | 6.72 | 0.88 | 2.80 |
| 5 | -8.64 | 3.06 | 3.50 |
| 6 | -6.87 | 0.57 | -5.45 |
| 7 | 4.47 | -2.62 | 5.76 |
| 8 | 6.73 | -2.01 | 4.18 |
| 9 | -7.71 | 2.34 | -6.33 |
| 10 | -6.91 | -0.49 | -5.68 |

| sample | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|
| 11 | 6.18 | 2.81 | 5.82 |
| 12 | 6.72 | -0.93 | -4.04 |
| 13 | -6.25 | -0.26 | 0.56 |
| 14 | -6.94 | -1.22 | 1.13 |
| 15 | 8.09 | 0.20 | 2.25 |
| 16 | 6.81 | 0.17 | -4.15 |
| 17 | -5.19 | 4.24 | 4.04 |
| 18 | -6.38 | -1.74 | 1.43 |
| 19 | 4.08 | 1.30 | 5.33 |
| 20 | 6.27 | 0.93 | -2.78 |

*Section 10.4*

**1.** Consider the univariate normal mixture

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

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

(a) Known: $P(\omega_1) = 0.5$, $\sigma_1 = \sigma_2 = 1$; Unknown: $\mu_1$ and $\mu_2$.

(b) Known: $P(\omega_1) = 0.5$; Unknown: $\sigma_1 = \sigma_2 = \sigma$, $\mu_1$ and $\mu_2$.

(c) Known: $P(\omega_1) = 0.5$; Unknown: $\sigma_1$, $\sigma_2$, $\mu_1$ and $\mu_2$.

(d) Unknown: $P(\omega_1)$, $\sigma_1$, $\sigma_2$, $\mu_1$ and $\mu_2$.

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

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

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

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

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

**3.** Repeat Computer exercise 2, but use instead a fuzzy k-means algorithm (Algorithm 1) with the "blending" be set by $b = 2$ (Eqs. 27 & 28).

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

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

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

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

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

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

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

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

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

⊕ *Section 10.5*

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

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

(b) Assume the data comes from a normal distribution with known variance, but unknown mean, i.e., $p(x) \sim N(\mu, 2)$ — that is, the unknown parameter $\boldsymbol{\theta}$ in Eq. 37 is simply the scalar $\mu$. Assume a wide prior for the parameter: $p(\mu)$ is uniform in the range $-10 \leq \mu \leq +10$. Plot posterior probabilities for $k = 0, 1, 2, 3, 4, 5, 10, 15, 20, 25, 30$ points from $\mathcal{D}$.

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

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

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

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

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

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

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

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

⊕ *Section 10.6*

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

(a) First, generate a two-dimensional data set consisting of two parts: $\mathcal{D}_1$ contains 100 points whose distance from the origin is chosen uniformly in the range $3 \leq r \leq 5$, and angular position uniform in the range $0 \leq \phi < 2\pi$; likewise, $\mathcal{D}_2$ consists of 50 points of distance $0 \leq r \leq 2$ and angle $0 \leq \phi < 2\pi$. The full data set used below is $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$.

(b) Write a simple clustering algorithm that links any two points $\mathbf{x}$ and $\mathbf{x}'$ if $d(\mathbf{x}, \mathbf{x}') < \theta$, where $\theta$ is a threshold selected by the user, and distance is calculated by means of a general Minkowski metric (Eq. 44),

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

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

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

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

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

⊕ *Section 10.7*

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

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

(b) Write a program to search through each of the cluster configurations, and for each compute the following criteria: $J_e$ (Eq. 49), $J_d$ (Eq. 63), $\sum_{i=1}^{d} \lambda_i$ (Eq. 64), $J_f = tr\mathbf{S}_T^{-1}\mathbf{S}_W$ (Eq. 65) and $|\mathbf{S}|/|\mathbf{S}_T|$ (Eq. 66). show the optimal clusters for each of your four criteria.

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

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

⊕ *Section 10.8*

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

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

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

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

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

(e) Explain why your final answers differ.

⊕ *Section 10.9*

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

(a) $d_{min}$ (Eq. 74)

(b) $d_{max}$ (Eq. 75)

(c) $d_{avg}$ (Eq. 76)

(d) $d_{mean}$ (Eq. 77)

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

(a) Write a program to perform hierarchical clustering and display a dendrogram, using measure of distance to be selected from the Eqs. 74 – 77.

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

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

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

⊕ *Section 10.10*

⊕ *Section 10.12*

**13.** xxx

⊕ *Section 10.11*

**14.** Implement a basic competitive learning clustering algorithm (Algorithm 6) and apply it to the three-dimensional data in the table above as follows.

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

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

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

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

⊕ *Section 10.13*

**15.** PCA exercise

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

(a) Generate 100 points for $t = 1, \ldots, 100$ for $x_1(t) = xxx$ and $x_2(t) = xxx$. Generate 100 points each for three sensors according to:

$$x_1(t) = xxx$$
$$x_2(t) = xxx$$

and three sensors:

$$s_1(t) = xxx$$
$$s_2(t) = xxx$$
$$s_3(t) = xxx$$

(Of course, in this blind source separation task, neither the source signals nor the mixing parameters are known.)

(b) xxx

**17.** Repeat Computer exercise 16, but for three sources:

$$x_1(t) = xxx$$
$$x_2(t) = xxx$$
$$x_3(t) = xxx$$

and four sensors:

$$s_1(t) = xxx$$
$$s_2(t) = xxx$$
$$s_3(t) = xxx$$
$$s_4(t) = xxx$$

⊕ *Section 10.14*

**18.** Write a computer program that uses the general maximum-likelihood equation of Sect. **??** iteratively to estimate the unknown means, variances, and prior probabilities. Use this program to find maximum-likelihood estimates of these parameters for the data in Table **??**.

**19.** hill climbing for clustering. Start at BAD and at GOOD starting places. Note that do not get same answer.

**20.** Write a program to perform the minimization of in Problem 28.

**21.** what if you have the wrong number of clusters?....xxx

# Bibliography


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

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

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

[4] James C. Bezdek. *Fuzzy mathematics in pattern classification*. Ph.D. thesis, Cornell University, Applied Mathematics Center, Ithaca, NY, 1973.

[5] James C. Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Plenum Press, New York, NY, 1981.

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

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

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

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

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

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

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

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

[14] Peter L. Forey, Christopher J. Humphries, Ian J. Kitching, Robert W. Scotland, Darrell J. Siebert, and David M. Williams. *Cladistics: A practical course in systematics*. Clarendon Press, Oxford, UK, 1 edition, 1992.

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

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

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

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

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

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

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

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

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

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

[25] Tuevo Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, 3 edition, 1989.

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

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

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

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

[30] Barbara Moore. Art1 and pattern clustering. In David Touretzky, Geoffrey Hinton, and Terrence Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 174–185, San Mateo, CA, 1988. (Pittsburg 1988), Morgan Kaufmann.

[31] Barak A. Pearlmutter and Lucas C. Parra. A context-sensitive generalization of ICA. In *International Conference on Neural Information Processing*, pages 151–157, Hong Kong, Sept. 24–27 1996. Springer Verlag.

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

[33] James O. Ramsay. Maximum likelihood estimation in nultidimensional scaling. *Psychometricka*, 42:241–266, 1977.

[34] Susan S. Schiffman, Mark L. Reynolds, and F. W. Young. *Introduction to Multidimensional Scaling: Theory, Methods and Applications*. Academic Press, New York, NY, 1981.

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

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

[37] Henry Teicher. Identifiability of mixtures. *Annals of Mathematical Statistics*, 32:244–248, 1961.

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

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

# Index