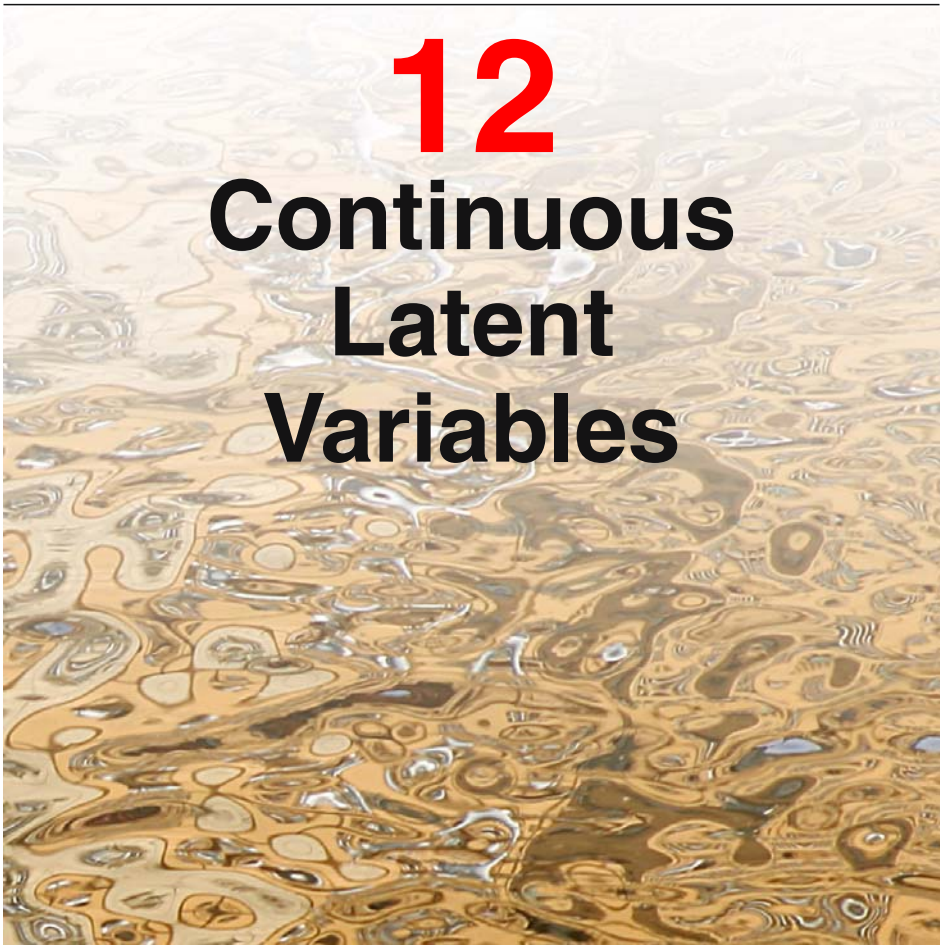


12 Continuous Latent Variables

In Chapter 9, we discussed probabilistic models having discrete latent variables, such as the mixture of Gaussians. We now explore models in which some, or all, of the latent variables are continuous. An important motivation for such models is that many data sets have the property that the data points all lie close to a manifold of much lower dimensionality than that of the original data space. To see why this might arise, consider an artificial data set constructed by taking one of the off-line digits, represented by a 64×64 pixel grey-level image, and embedding it in a larger image of size 100×100 by padding with pixels having the value zero (corresponding to white pixels) in which the location and orientation of the digit is varied at random, as illustrated in Figure 12.1. Each of the resulting images is represented by a point in the $100 \times 100 = 10,000$-dimensional data space. However, across a data set of such images, there are only three *degrees of freedom* of variability, corresponding to the vertical and horizontal translations and the rotations. The data points will therefore live on a subspace of the data space whose *intrinsic dimensionality* is three. Note

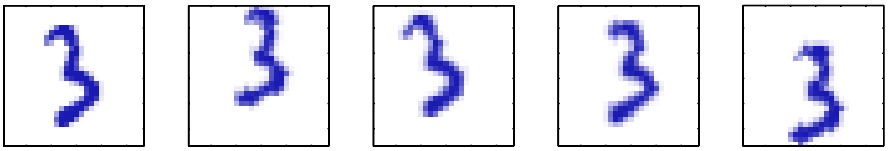


Figure 12.1 A synthetic data set obtained by taking one of the off-line digit images and creating multiple copies in each of which the digit has undergone a random displacement and rotation within some larger image field. The resulting images each have $100 \times 100 = 10,000$ pixels.

that the manifold will be nonlinear because, for instance, if we translate the digit past a particular pixel, that pixel value will go from zero (white) to one (black) and back to zero again, which is clearly a nonlinear function of the digit position. In this example, the translation and rotation parameters are latent variables because we observe only the image vectors and are not told which values of the translation or rotation variables were used to create them.

For real digit image data, there will be a further degree of freedom arising from scaling. Moreover there will be multiple additional degrees of freedom associated with more complex deformations due to the variability in an individual's writing as well as the differences in writing styles between individuals. Nevertheless, the number of such degrees of freedom will be small compared to the dimensionality of the data set.

Appendix A Another example is provided by the oil flow data set, in which (for a given geometrical configuration of the gas, water, and oil phases) there are only two degrees of freedom of variability corresponding to the fraction of oil in the pipe and the fraction of water (the fraction of gas then being determined). Although the data space comprises 12 measurements, a data set of points will lie close to a two-dimensional manifold embedded within this space. In this case, the manifold comprises several distinct segments corresponding to different flow regimes, each such segment being a (noisy) continuous two-dimensional manifold. If our goal is data compression, or density modelling, then there can be benefits in exploiting this manifold structure. In practice, the data points will not be confined precisely to a smooth low-dimensional manifold, and we can interpret the departures of data points from the manifold as 'noise'. This leads naturally to a generative view of such models in which we first select a point within the manifold according to some latent variable distribution and then generate an observed data point by adding noise, drawn from some conditional distribution of the data variables given the latent variables.

Section 8.1.4 The simplest continuous latent variable model assumes Gaussian distributions for both the latent and observed variables and makes use of a linear-Gaussian dependence of the observed variables on the state of the latent variables. This leads to a probabilistic formulation of the well-known technique of principal component analysis (PCA), as well as to a related model called factor analysis.

Section 12.1 In this chapter we will begin with a standard, nonprobabilistic treatment of PCA, and then we show how PCA arises naturally as the maximum likelihood solution to

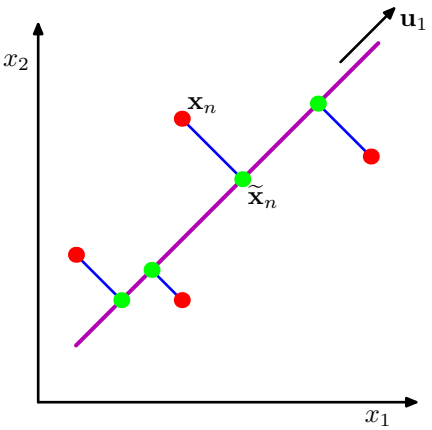


Figure 12.2 Principal component analysis seeks a space of lower dimensionality, known as the principal subspace and denoted by the magenta line, such that the orthogonal projection of the data points (red dots) onto this subspace maximizes the variance of the projected points (green dots). An alternative definition of PCA is based on minimizing the sum-of-squares of the projection errors, indicated by the blue lines.

Section 12.2 a particular form of linear-Gaussian latent variable model. This probabilistic reformulation brings many advantages, such as the use of EM for parameter estimation, principled extensions to mixtures of PCA models, and Bayesian formulations that allow the number of principal components to be determined automatically from the data. Finally, we discuss briefly several generalizations of the latent variable concept that go beyond the linear-Gaussian assumption including non-Gaussian latent variables, which leads to the framework of *independent component analysis*, as well as *Section 12.4* models having a nonlinear relationship between latent and observed variables.

12.1. Principal Component Analysis

Principal component analysis, or PCA, is a technique that is widely used for applications such as dimensionality reduction, lossy data compression, feature extraction, and data visualization (Jolliffe, 2002). It is also known as the *Karhunen-Loève* transform.

There are two commonly used definitions of PCA that give rise to the same algorithm. PCA can be defined as the orthogonal projection of the data onto a lower dimensional linear space, known as the *principal subspace*, such that the variance of the projected data is maximized (Hotelling, 1933). Equivalently, it can be defined as the linear projection that minimizes the average projection cost, defined as the mean squared distance between the data points and their projections (Pearson, 1901). The process of orthogonal projection is illustrated in Figure 12.2. We consider each of these definitions in turn.

12.1.1 Maximum variance formulation

Consider a data set of observations $\{\mathbf{x}_n\}$ where $n = 1, \ldots, N$, and $\mathbf{x}_n$ is a Euclidean variable with dimensionality D. Our goal is to project the data onto a space having dimensionality $M < D$ while maximizing the variance of the projected data. For the moment, we shall assume that the value of M is given. Later in this

chapter, we shall consider techniques to determine an appropriate value of M from the data.

To begin with, consider the projection onto a one-dimensional space ($M = 1$). We can define the direction of this space using a D-dimensional vector $\mathbf{u}_1$, which for convenience (and without loss of generality) we shall choose to be a unit vector so that $\mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1 = 1$ (note that we are only interested in the direction defined by $\mathbf{u}_1$, not in the magnitude of $\mathbf{u}_1$ itself). Each data point $\mathbf{x}_n$ is then projected onto a scalar value $\mathbf{u}_1^{\mathrm{T}}\mathbf{x}_n$. The mean of the projected data is $\mathbf{u}_1^{\mathrm{T}}\overline{\mathbf{x}}$ where $\overline{\mathbf{x}}$ is the sample set mean given by

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n \tag{12.1}$$

and the variance of the projected data is given by

$$\frac{1}{N}\sum_{n=1}^{N}\left\{\mathbf{u}_1^{\mathrm{T}}\mathbf{x}_n - \mathbf{u}_1^{\mathrm{T}}\overline{\mathbf{x}}\right\}^2 = \mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1 \tag{12.2}$$

where $\mathbf{S}$ is the data covariance matrix defined by

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^{\mathrm{T}}. \tag{12.3}$$

We now maximize the projected variance $\mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1$ with respect to $\mathbf{u}_1$. Clearly, this has to be a constrained maximization to prevent $\|\mathbf{u}_1\| \to \infty$. The appropriate constraint comes from the normalization condition $\mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1 = 1$. To enforce this constraint, we introduce a Lagrange multiplier that we shall denote by λ_1, and then make an unconstrained maximization of

Appendix E

$$\mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1 + \lambda_1\left(1 - \mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1\right). \tag{12.4}$$

By setting the derivative with respect to $\mathbf{u}_1$ equal to zero, we see that this quantity will have a stationary point when

$$\mathbf{S}\mathbf{u}_1 = \lambda_1\mathbf{u}_1 \tag{12.5}$$

which says that $\mathbf{u}_1$ must be an eigenvector of $\mathbf{S}$. If we left-multiply by $\mathbf{u}_1^{\mathrm{T}}$ and make use of $\mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1 = 1$, we see that the variance is given by

$$\mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1 = \lambda_1 \tag{12.6}$$

and so the variance will be a maximum when we set $\mathbf{u}_1$ equal to the eigenvector having the largest eigenvalue λ_1. This eigenvector is known as the first principal component.

We can define additional principal components in an incremental fashion by choosing each new direction to be that which maximizes the projected variance

amongst all possible directions orthogonal to those already considered. If we consider the general case of an M-dimensional projection space, the optimal linear projection for which the variance of the projected data is maximized is now defined by the M eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_M$ of the data covariance matrix $\mathbf{S}$ corresponding to the M largest eigenvalues $\lambda_1, \ldots, \lambda_M$. This is easily shown using proof by induction.

数学归纳法证明

Exercise 12.1

To summarize, principal component analysis involves evaluating the mean $\overline{\mathbf{x}}$ and the covariance matrix $\mathbf{S}$ of the data set and then finding the M eigenvectors of $\mathbf{S}$ corresponding to the M largest eigenvalues. Algorithms for finding eigenvectors and eigenvalues, as well as additional theorems related to eigenvector decomposition, can be found in Golub and Van Loan (1996). Note that the computational cost of computing the full eigenvector decomposition for a matrix of size $D \times D$ is $O(D^3)$. If we plan to project our data onto the first M principal components, then we only need to find the first M eigenvalues and eigenvectors. This can be done with more efficient techniques, such as the *power method* (Golub and Van Loan, 1996), that scale like $O(MD^2)$, or alternatively we can make use of the EM algorithm.

Section 12.2.2

12.1.2 Minimum-error formulation

We now discuss an alternative formulation of PCA based on projection error minimization. To do this, we introduce a complete orthonormal set of D-dimensional basis vectors $\{\mathbf{u}_i\}$ where $i = 1, \ldots, D$ that satisfy

Appendix C

$$\mathbf{u}_i^{\mathrm{T}} \mathbf{u}_j = \delta_{ij}. \tag{12.7}$$

Because this basis is complete, each data point can be represented exactly by a linear combination of the basis vectors

$$\mathbf{x}_n = \sum_{i=1}^{D} \alpha_{ni} \mathbf{u}_i \tag{12.8}$$

where the coefficients α_{ni} will be different for different data points. This simply corresponds to a rotation of the coordinate system to a new system defined by the $\{\mathbf{u}_i\}$, and the original D components $\{x_{n1}, \ldots, x_{nD}\}$ are replaced by an equivalent set $\{\alpha_{n1}, \ldots, \alpha_{nD}\}$. Taking the inner product with $\mathbf{u}_j$, and making use of the orthonormality property, we obtain $\alpha_{nj} = \mathbf{x}_n^{\mathrm{T}} \mathbf{u}_j$, and so without loss of generality we can write

$$\mathbf{x}_n = \sum_{i=1}^{D} \left(\mathbf{x}_n^{\mathrm{T}} \mathbf{u}_i\right) \mathbf{u}_i. \tag{12.9}$$

Our goal, however, is to approximate this data point using a representation involving a restricted number $M < D$ of variables corresponding to a projection onto a lower-dimensional subspace. The M-dimensional linear subspace can be represented, without loss of generality, by the first M of the basis vectors, and so we approximate each data point $\mathbf{x}_n$ by

给出了超平面的方向

给出了超平面的偏移量

$$\widetilde{\mathbf{x}}_n = \sum_{i=1}^{M} z_{ni} \mathbf{u}_i + \left(\sum_{i=M+1}^{D} b_i \mathbf{u}_i\right) \tag{12.10}$$

$\mathbf{x}_n$在M维空间/超平面上对应的点，也就是投影点

由式(12.10)可知，当$\{\mathbf{x}_n\}$已归一化时，$b_i = 0$ $(i = M+1, \ldots, D)$

where the $\{z_{ni}\}$ depend on the particular data point, whereas the $\{b_i\}$ are constants that are the same for all data points. We are free to choose the $\{\mathbf{u}_i\}$, the $\{z_{ni}\}$, and the $\{b_i\}$ so as to minimize the distortion introduced by the reduction in dimensionality. As our distortion measure, we shall use the squared distance between the original data point $\mathbf{x}_n$ and its approximation $\widetilde{\mathbf{x}}_n$, averaged over the data set, so that our goal is to minimize

$$J = \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{x}_n - \widetilde{\mathbf{x}}_n\|^2. \tag{12.11}$$

Consider first of all the minimization with respect to the quantities $\{z_{ni}\}$. Substituting for $\widetilde{\mathbf{x}}_n$, setting the derivative with respect to z_{nj} to zero, and making use of the orthonormality conditions, we obtain

$$z_{nj} = \mathbf{x}_n^{\mathrm{T}}\mathbf{u}_j \tag{12.12}$$

where $j = 1, \ldots, M$. Similarly, setting the derivative of J with respect to b_i to zero, and again making use of the orthonormality relations, gives

$$b_j = \overline{\mathbf{x}}^{\mathrm{T}}\mathbf{u}_j \tag{12.13}$$

where $j = M+1, \ldots, D$. If we substitute for z_{ni} and b_i, and make use of the general expansion (12.9), we obtain

$$\mathbf{x}_n - \widetilde{\mathbf{x}}_n = \sum_{i=M+1}^{D} \left\{(\mathbf{x}_n - \overline{\mathbf{x}})^{\mathrm{T}}\mathbf{u}_i\right\}\mathbf{u}_i \tag{12.14}$$

from which we see that the displacement vector from $\mathbf{x}_n$ to $\widetilde{\mathbf{x}}_n$ lies in the space orthogonal to the principal subspace, because it is a linear combination of $\{\mathbf{u}_i\}$ for $i = M+1, \ldots, D$, as illustrated in Figure 12.2. This is to be expected because the projected points $\widetilde{\mathbf{x}}_n$ must lie within the principal subspace, but we can move them freely within that subspace, and so the minimum error is given by the orthogonal projection.

We therefore obtain an expression for the distortion measure J as a function purely of the $\{\mathbf{u}_i\}$ in the form

$$J = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D}\left(\mathbf{x}_n^{\mathrm{T}}\mathbf{u}_i - \overline{\mathbf{x}}^{\mathrm{T}}\mathbf{u}_i\right)^2 = \sum_{i=M+1}^{D}\mathbf{u}_i^{\mathrm{T}}\mathbf{S}\mathbf{u}_i. \tag{12.15}$$

There remains the task of minimizing J with respect to the $\{\mathbf{u}_i\}$, which must be a constrained minimization otherwise we will obtain the vacuous result $\mathbf{u}_i = 0$. The constraints arise from the orthonormality conditions and, as we shall see, the solution will be expressed in terms of the eigenvector expansion of the covariance matrix. Before considering a formal solution, let us try to obtain some intuition about the result by considering the case of a two-dimensional data space $D = 2$ and a one-dimensional principal subspace $M = 1$. We have to choose a direction $\mathbf{u}_2$ so as to

minimize $J = \mathbf{u}_2^{\mathrm{T}}\mathbf{S}\mathbf{u}_2$, subject to the normalization constraint $\mathbf{u}_2^{\mathrm{T}}\mathbf{u}_2 = 1$. Using a Lagrange multiplier λ_2 to enforce the constraint, we consider the minimization of

$$\widetilde{J} = \mathbf{u}_2^{\mathrm{T}}\mathbf{S}\mathbf{u}_2 + \lambda_2\left(1 - \mathbf{u}_2^{\mathrm{T}}\mathbf{u}_2\right). \tag{12.16}$$

Setting the derivative with respect to $\mathbf{u}_2$ to zero, we obtain $\mathbf{S}\mathbf{u}_2 = \lambda_2\mathbf{u}_2$ so that $\mathbf{u}_2$ is an eigenvector of $\mathbf{S}$ with eigenvalue λ_2. Thus any eigenvector will define a stationary point of the distortion measure. To find the value of J at the minimum, we back-substitute the solution for $\mathbf{u}_2$ into the distortion measure to give $J = \lambda_2$. We therefore obtain the minimum value of J by choosing $\mathbf{u}_2$ to be the eigenvector corresponding to the smaller of the two eigenvalues. Thus we should choose the principal subspace to be aligned with the eigenvector having the *larger* eigenvalue. This result accords with our intuition that, in order to minimize the average squared projection distance, we should choose the principal component subspace to pass through the mean of the data points and to be aligned with the directions of maximum variance. For the case when the eigenvalues are equal, any choice of principal direction will give rise to the same value of J.

Exercise 12.2

The general solution to the minimization of J for arbitrary D and arbitrary $M < D$ is obtained by choosing the $\{\mathbf{u}_i\}$ to be eigenvectors of the covariance matrix given by

$$\mathbf{S}\mathbf{u}_i = \lambda_i\mathbf{u}_i \tag{12.17}$$

where $i = 1, \ldots, D$, and as usual the eigenvectors $\{\mathbf{u}_i\}$ are chosen to be orthonormal. The corresponding value of the distortion measure is then given by

$$J = \sum_{i=M+1}^{D} \lambda_i \tag{12.18}$$

which is simply the sum of the eigenvalues of those eigenvectors that are orthogonal to the principal subspace. We therefore obtain the minimum value of J by selecting these eigenvectors to be those having the $D - M$ smallest eigenvalues, and hence the eigenvectors defining the principal subspace are those corresponding to the M largest eigenvalues.

Although we have considered $M < D$, the PCA analysis still holds if $M = D$, in which case there is no dimensionality reduction but simply a rotation of the coordinate axes to align with principal components.

Finally, it is worth noting that there exists a closely related linear dimensionality reduction technique called *canonical correlation analysis*, or *CCA* (Hotelling, 1936; Bach and Jordan, 2002). Whereas PCA works with a single random variable, CCA considers two (or more) variables and tries to find a corresponding pair of linear subspaces that have high cross-correlation, so that each component within one of the subspaces is correlated with a single component from the other subspace. Its solution can be expressed in terms of a generalized eigenvector problem.

12.1.3 Applications of PCA

We can illustrate the use of PCA for data compression by considering the off-line digits data set. Because each eigenvector of the covariance matrix is a vector

Appendix A

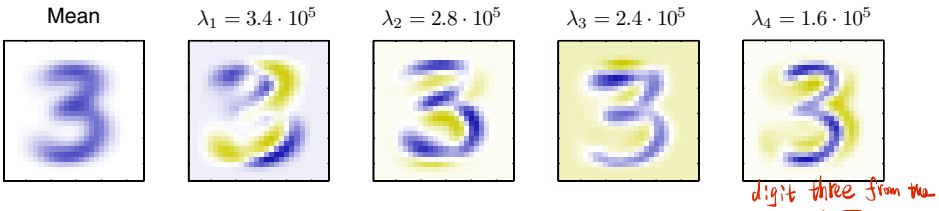


Figure 12.3 The mean vector $\overline{\mathbf{x}}$ along with the first four PCA eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_4$ for the off-line digits data set, together with the corresponding eigenvalues.

in the original D-dimensional space, we can represent the eigenvectors as images of the same size as the data points. The first five eigenvectors, along with the corresponding eigenvalues, are shown in Figure 12.3. A plot of the complete spectrum of eigenvalues, sorted into decreasing order, is shown in Figure 12.4(a). The distortion measure J associated with choosing a particular value of M is given by the sum of the eigenvalues from $M + 1$ up to D and is plotted for different values of M in Figure 12.4(b).

If we substitute (12.12) and (12.13) into (12.10), we can write the PCA approximation to a data vector $\mathbf{x}_n$ in the form

$$\widetilde{\mathbf{x}}_n = \sum_{i=1}^{M} (\mathbf{x}_n^{\mathrm{T}}\mathbf{u}_i)\mathbf{u}_i + \sum_{i=M+1}^{D} (\overline{\mathbf{x}}^{\mathrm{T}}\mathbf{u}_i)\mathbf{u}_i \tag{12.19}$$

$$= \overline{\mathbf{x}} + \sum_{i=1}^{M} \left(\mathbf{x}_n^{\mathrm{T}}\mathbf{u}_i - \overline{\mathbf{x}}^{\mathrm{T}}\mathbf{u}_i\right)\mathbf{u}_i \tag{12.20}$$

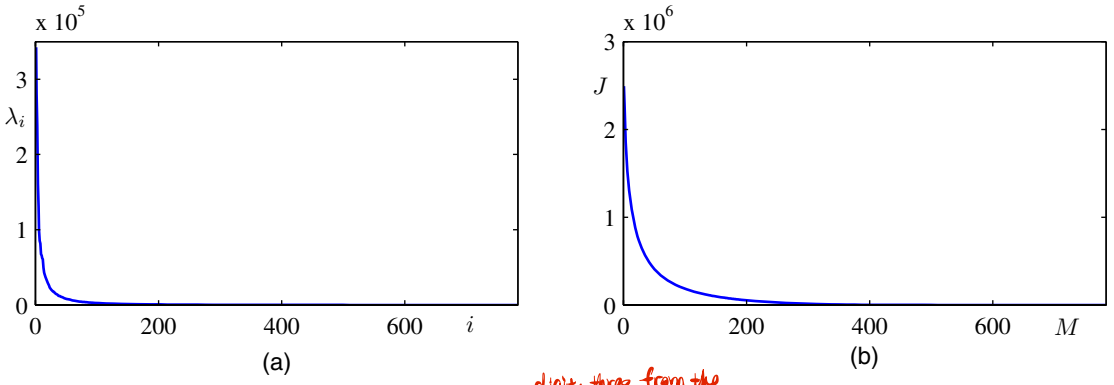


Figure 12.4 (a) Plot of the eigenvalue spectrum for the off-line digits data set. (b) Plot of the sum of the discarded eigenvalues, which represents the sum-of-squares distortion J introduced by projecting the data onto a principal component subspace of dimensionality M.

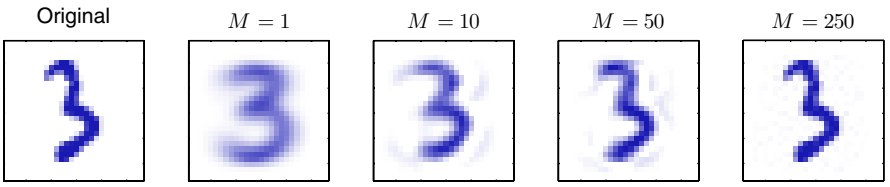


Figure 12.5 An original example from the off-line digits data set together with its PCA reconstructions obtained by retaining M principal components for various values of M. As M increases the reconstruction becomes more accurate and would become perfect when $M = D = 28 \times 28 = 784$.

where we have made use of the relation

$$\overline{\mathbf{x}} = \sum_{i=1}^{D} \left(\overline{\mathbf{x}}^{\mathrm{T}} \mathbf{u}_i\right) \mathbf{u}_i \tag{12.21}$$

which follows from the completeness of the $\{\mathbf{u}_i\}$. This represents a compression of the data set, because for each data point we have replaced the D-dimensional vector $\mathbf{x}_n$ with an M-dimensional vector having components $\left(\mathbf{x}_n^{\mathrm{T}} \mathbf{u}_i - \overline{\mathbf{x}}^{\mathrm{T}} \mathbf{u}_i\right)$. The smaller the value of M, the greater the degree of compression. Examples of ~~PCA~~ reconstructions of ~~data points for~~ the digits data set are shown in Figure 12.5.

Another application of principal component analysis is to data pre-processing. In this case, the goal is not dimensionality reduction but rather the transformation of a data set in order to standardize certain of its properties. This can be important in allowing subsequent pattern recognition algorithms to be applied successfully to the data set. Typically, it is done when the original variables are measured in various different units or have significantly different variability. For instance in the Old Faithful data set, the time between eruptions is typically an order of magnitude greater than the duration of an eruption. When we applied the K-means algorithm to this data set, we first made a separate linear re-scaling of the individual variables such that each variable had zero mean and unit variance. This is known as *standardizing* the data, and the covariance matrix for the standardized data has components

$$\rho_{ij} = \frac{1}{N} \sum_{n=1}^{N} \frac{(x_{ni} - \overline{x}_i)}{\sigma_i} \frac{(x_{nj} - \overline{x}_j)}{\sigma_j} \tag{12.22}$$

where σ_i is the ~~variance~~ of x_i. This is known as the *correlation* matrix of the original data and has the property that if two components x_i and x_j of the data are perfectly correlated, then $\rho_{ij} = 1$, and if they are uncorrelated, then $\rho_{ij} = 0$.

However, using PCA we can make a more substantial normalization of the data to give it zero mean and unit covariance, so that different variables become decorrelated. To do this, we first write the eigenvector equation (12.17) in the form

$$\mathbf{SU} = \mathbf{UL} \tag{12.23}$$

Appendix A

Section 9.1

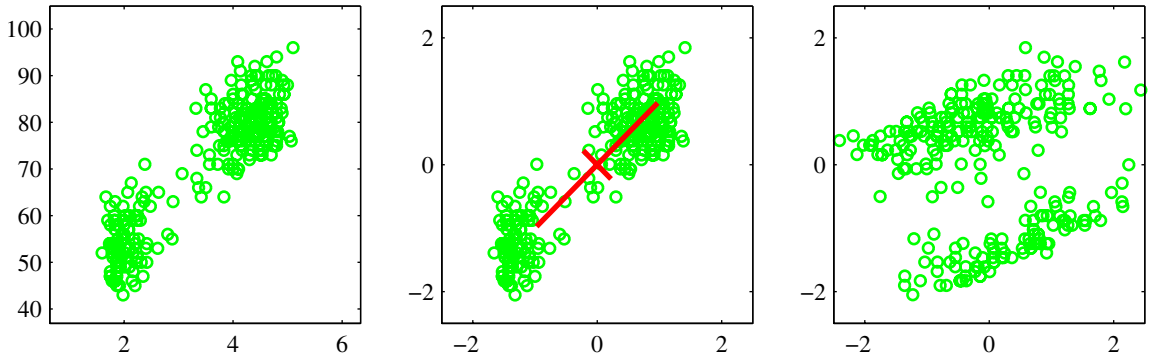


Figure 12.6 Illustration of the effects of linear pre-processing applied to the Old Faithful data set. The plot on the left shows the original data. The centre plot shows the result of standardizing the individual variables to zero mean and unit variance. Also shown are the principal axes of this normalized data set, plotted over the range $\pm\lambda_i^{1/2}$. The plot on the right shows the result of whitening of the data to give it zero mean and unit covariance.

where $\mathbf{L}$ is a $D \times D$ diagonal matrix with elements λ_i, and $\mathbf{U}$ is a $D \times D$ orthogonal matrix with columns given by $\mathbf{u}_i$. Then we define, for each data point $\mathbf{x}_n$, a transformed value given by

$$\mathbf{y}_n = \mathbf{L}^{-1/2}\mathbf{U}^{\mathrm{T}}(\mathbf{x}_n - \overline{\mathbf{x}}) \tag{12.24}$$

where $\overline{\mathbf{x}}$ is the sample mean defined by (12.1). Clearly, the set $\{\mathbf{y}_n\}$ has zero mean, and its covariance is given by the identity matrix because

$$\begin{aligned}\frac{1}{N}\sum_{n=1}^{N}\mathbf{y}_n\mathbf{y}_n^{\mathrm{T}} &= \frac{1}{N}\sum_{n=1}^{N}\mathbf{L}^{-1/2}\mathbf{U}^{\mathrm{T}}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^{\mathrm{T}}\mathbf{U}\mathbf{L}^{-1/2} \\ &= \mathbf{L}^{-1/2}\mathbf{U}^{\mathrm{T}}\mathbf{S}\mathbf{U}\mathbf{L}^{-1/2} = \mathbf{L}^{-1/2}\mathbf{L}\mathbf{L}^{-1/2} = \mathbf{I}. \end{aligned} \tag{12.25}$$

This operation is known as *whitening* or *sphereing* the data and is illustrated for the Old Faithful data set in Figure 12.6.

Appendix A

It is interesting to compare PCA with the Fisher linear discriminant which was discussed in Section 4.1.4. Both methods can be viewed as techniques for linear dimensionality reduction. However, PCA is unsupervised and depends only on the values $\mathbf{x}_n$ whereas Fisher linear discriminant also uses class-label information. This difference is highlighted by the example in Figure 12.7.

Another common application of principal component analysis is to data visualization. Here each data point is projected onto a two-dimensional ($M = 2$) principal subspace, so that a data point $\mathbf{x}_n$ is plotted at Cartesian coordinates given by $\mathbf{x}_n^{\mathrm{T}}\mathbf{u}_1$ and $\mathbf{x}_n^{\mathrm{T}}\mathbf{u}_2$, where $\mathbf{u}_1$ and $\mathbf{u}_2$ are the eigenvectors corresponding to the largest and second largest eigenvalues. An example of such a plot, for the oil flow data set, is shown in Figure 12.8.

Appendix A

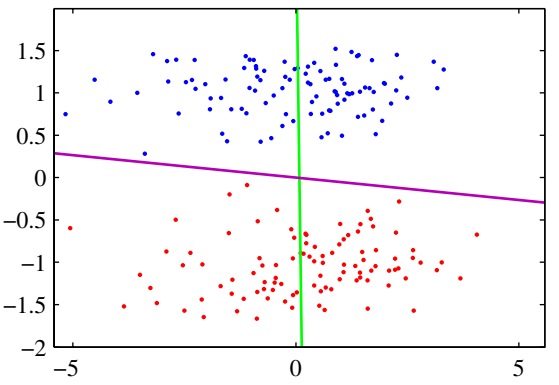


Figure 12.7 A comparison of principal component analysis with Fisher's linear discriminant for linear dimensionality reduction. Here the data in two dimensions, belonging to two classes shown in red and blue, is to be projected onto a single dimension. PCA chooses the direction of maximum variance, shown by the magenta curve, which leads to strong class overlap, whereas the Fisher linear discriminant takes account of the class labels and leads to a projection onto the green curve giving much better class separation.

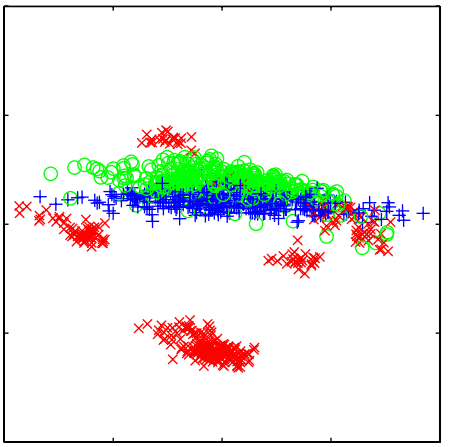


Figure 12.8 Visualization of the oil flow data set obtained by projecting the data onto the first two principal components. The red, blue, and green points correspond to the 'laminar', 'homogeneous', and 'annular' flow configurations respectively.

12.1.4 PCA for high-dimensional data

D:原始空间维度
M:低维空间维度
N:数据量
这部分是讨论D>N时如何更高效地进行PCA

In some applications of principal component analysis, the number of data points is smaller than the dimensionality of the data space. For example, we might want to apply PCA to a data set of a few hundred images, each of which corresponds to a vector in a space of potentially several million dimensions (corresponding to three colour values for each of the pixels in the image). Note that in a D-dimensional space a set of N points, where $N < D$, defines a linear subspace whose dimensionality is at most $N - 1$, and so there is little point in applying PCA for values of M that are greater than $N - 1$. Indeed, if we perform PCA we will find that at least $D - N + 1$ of the eigenvalues are zero, corresponding to eigenvectors along whose directions the data set has zero variance. Furthermore, typical algorithms for finding the eigenvectors of a $D \times D$ matrix have a computational cost that scales like $O(D^3)$, and so for applications such as the image example, a direct application of PCA will be computationally infeasible.

We can resolve this problem as follows. First, let us define $\mathbf{X}$ to be the $(N \times D)$-

dimensional centred data matrix, whose n^{th} row is given by $(\mathbf{x}_n - \overline{\mathbf{x}})^{\mathrm{T}}$. The covariance matrix (12.3) can then be written as $\mathbf{S} = N^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{X}$, and the corresponding eigenvector equation becomes

$$\frac{1}{N}\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{u}_i = \lambda_i \mathbf{u}_i. \tag{12.26}$$

本质上就是奇异值分解：
$X_{NXD} = V_{NXN} \Sigma_{NXD} U^{T}_{DXD}$

Now pre-multiply both sides by $\mathbf{X}$ to give

$$\frac{1}{N}\mathbf{X}\mathbf{X}^{\mathrm{T}}(\mathbf{X}\mathbf{u}_i) = \lambda_i(\mathbf{X}\mathbf{u}_i). \tag{12.27}$$

If we now define $\mathbf{v}_i = \mathbf{X}\mathbf{u}_i$, we obtain

$$\frac{1}{N}\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{12.28}$$

which is an eigenvector equation for the $N \times N$ matrix $N^{-1}\mathbf{X}\mathbf{X}^{\mathrm{T}}$. We see that this has the same $N-1$ eigenvalues as the original covariance matrix (which itself has an additional $D - N + 1$ eigenvalues of value zero). Thus we can solve the eigenvector problem in spaces of lower dimensionality with computational cost $O(N^3)$ instead of $O(D^3)$. In order to determine the eigenvectors, we multiply both sides of (12.28) by $\mathbf{X}^{\mathrm{T}}$ to give

$$\left(\frac{1}{N}\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)(\mathbf{X}^{\mathrm{T}}\mathbf{v}_i) = \lambda_i(\mathbf{X}^{\mathrm{T}}\mathbf{v}_i) \tag{12.29}$$

from which we see that $(\mathbf{X}^{\mathrm{T}}\mathbf{v}_i)$ is an eigenvector of $\mathbf{S}$ with eigenvalue λ_i. Note, however, that these eigenvectors need not be normalized. To determine the appropriate normalization, we re-scale $\mathbf{u}_i \propto \mathbf{X}^{\mathrm{T}}\mathbf{v}_i$ by a constant such that $\|\mathbf{u}_i\| = 1$, which, assuming $\mathbf{v}_i$ has been normalized to unit length, gives

$$\mathbf{u}_i = \frac{1}{(N\lambda_i)^{1/2}}\mathbf{X}^{\mathrm{T}}\mathbf{v}_i. \tag{12.30}$$

In summary, to apply this approach we first evaluate $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ and then find its eigenvectors and eigenvalues and then compute the eigenvectors in the original data space using (12.30).

12.2. Probabilistic PCA

The formulation of PCA discussed in the previous section was based on a linear projection of the data onto a subspace of lower dimensionality than the original data space. We now show that PCA can also be expressed as the maximum likelihood solution of a probabilistic latent variable model. This reformulation of PCA, known as *probabilistic PCA*, brings several advantages compared with conventional PCA:

- Probabilistic PCA represents a constrained form of the Gaussian distribution in which the number of free parameters can be restricted while still allowing the model to capture the dominant correlations in a data set.

- We can derive an EM algorithm for PCA that is computationally efficient in situations where only a few leading eigenvectors are required and that avoids having to evaluate the data covariance matrix as an intermediate step.

Section 12.2.2

- The combination of a probabilistic model and EM allows us to deal with missing values in the data set. P179 第3点提了一嘴
- Mixtures of probabilistic PCA models can be formulated in a principled way and trained using the EM algorithm.
- Probabilistic PCA forms the basis for a Bayesian treatment of PCA in which the dimensionality of the principal subspace can be found automatically from the data.

Section 12.2.3

和PPCA不同，Conventional PCA压根就不会对x的分布作假设，它只关注x与z之间的转化，x可服从任意分布。可见二者基于不同的假设，模有可比性，因此这一条不能这么算。

- The existence of a likelihood function allows direct comparison with other probabilistic density models. By contrast, conventional PCA will assign a low reconstruction cost to data points that are close to the principal subspace even if they lie arbitrarily far from the training data.
- Probabilistic PCA can be used to model class-conditional densities and hence be applied to classification problems. 书中未做，但估计是利用类似GMM混合模型的式
- The probabilistic PCA model can be run generatively to provide samples from the distribution.

This formulation of PCA as a probabilistic model was proposed independently by Tipping and Bishop (1997, 1999b) and by Roweis (1998). As we shall see later, it is closely related to *factor analysis* (Basilevsky, 1994).

Section 8.1.4 介绍模型

Probabilistic PCA is a simple example of the linear-Gaussian framework, in which all of the marginal and conditional distributions are Gaussian. We can formulate probabilistic PCA by first introducing an explicit latent variable $\mathbf{z}$ corresponding to the principal-component subspace. Next we define a Gaussian prior distribution $p(\mathbf{z})$ over the latent variable, together with a Gaussian conditional distribution $p(\mathbf{x}|\mathbf{z})$ for the observed variable $\mathbf{x}$ conditioned on the value of the latent variable. Specifically, the prior distribution over $\mathbf{z}$ is given by a zero-mean unit-covariance Gaussian

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}). \tag{12.31}$$

注意，不要认为PPCA是CPCA(Conventional PCA)在概率论框架下的表述。CPCA只讨论x与z之间的变换，而对x服从什么分布没有要求，可以服从任意分布，或者是满足某个确定函数f(x)=0，对其同样也是如此；而PPCA则基于x,z为高斯分布的情况下展开。当然，PPCA之所以也叫PCA，是因为基于ML估计，在极限状态($\sigma^2 \to 0$)下能得到和CPCA相同的x与z之间的变换关系，即式(12.50)。

Section 8.2.2

Similarly, the conditional distribution of the observed variable $\mathbf{x}$, conditioned on the value of the latent variable $\mathbf{z}$, is again Gaussian, of the form

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I}) \tag{12.32}$$

in which the mean of $\mathbf{x}$ is a general linear function of $\mathbf{z}$ governed by the $D \times M$ matrix $\mathbf{W}$ and the D-dimensional vector $\boldsymbol{\mu}$. Note that this factorizes with respect to the elements of $\mathbf{x}$, in other words this is an example of the naive Bayes model. As we shall see shortly, the columns of $\mathbf{W}$ span a linear subspace within the data space that corresponds to the principal subspace. The other parameter in this model is the scalar σ^2 governing the variance of the conditional distribution. Note that there is no

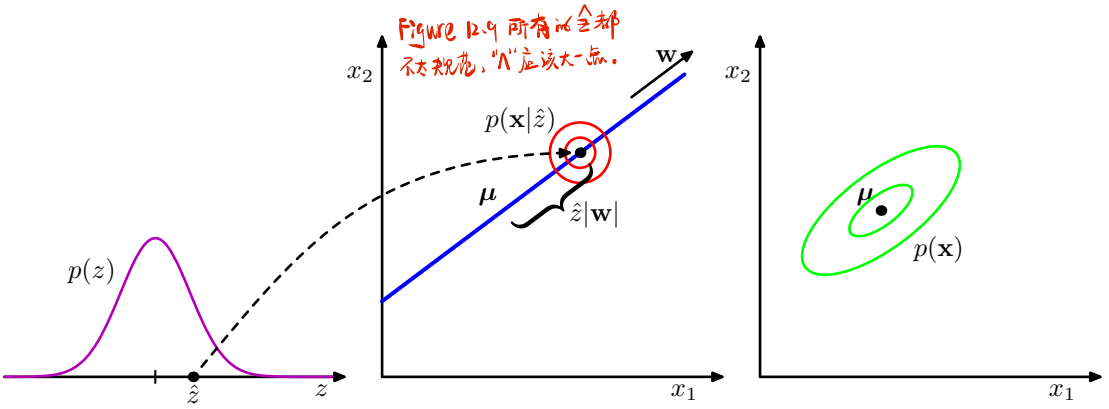


Figure 12.9 An illustration of the generative view of the probabilistic PCA model for a two-dimensional data space and a one-dimensional latent space. An observed data point $\mathbf{x}$ is generated by first drawing a value $\widehat{z}$ for the latent variable from its prior distribution $p(z)$ and then drawing a value for $\mathbf{x}$ from an isotropic Gaussian distribution (illustrated by the red circles) having mean $\mathbf{w}\widehat{z} + \boldsymbol{\mu}$ and covariance $\sigma^2\mathbf{I}$. The green ellipses show the density contours for the marginal distribution $p(\mathbf{x})$.

给学长注明

loss of generality in assuming a zero mean, unit covariance Gaussian for the latent distribution $p(\mathbf{z})$ because a more general Gaussian distribution would give rise to an equivalent probabilistic model.

Exercise 12.4

We can view the probabilistic PCA model from a generative viewpoint in which a sampled value of the observed variable is obtained by first choosing a value for the latent variable and then sampling the observed variable conditioned on this latent value. Specifically, the D-dimensional observed variable $\mathbf{x}$ is defined by a linear transformation of the M-dimensional latent variable $\mathbf{z}$ plus additive Gaussian 'noise', so that

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon} \tag{12.33}$$

where $\mathbf{z}$ is an M-dimensional Gaussian latent variable, and $\boldsymbol{\epsilon}$ is a D-dimensional zero-mean Gaussian-distributed noise variable with covariance $\sigma^2\mathbf{I}$. This generative process is illustrated in Figure 12.9. Note that this framework is based on a mapping from latent space to data space, in contrast to the more conventional view of PCA discussed above. The reverse mapping, from data space to the latent space, will be obtained shortly using Bayes' theorem.

为MLE作准备

Suppose we wish to determine the values of the parameters $\mathbf{W}$, $\boldsymbol{\mu}$ and σ^2 using maximum likelihood. To write down the likelihood function, we need an expression for the marginal distribution $p(\mathbf{x})$ of the observed variable. This is expressed, from the sum and product rules of probability, in the form

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\,\mathrm{d}\mathbf{z}. \tag{12.34}$$

Exercise 12.7

Because this corresponds to a linear-Gaussian model, this marginal distribution is again Gaussian, and is given by

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{C}) \tag{12.35}$$

where the $D \times D$ covariance matrix $\mathbf{C}$ is defined by

$$\mathbf{C} = \mathbf{W}\mathbf{W}^{\mathrm{T}} + \sigma^2\mathbf{I}. \tag{12.36}$$

This result can also be derived more directly by noting that the predictive distribution will be Gaussian and then evaluating its mean and covariance using (12.33). This gives

$$\begin{aligned} \mathbb{E}[\mathbf{x}] &= \mathbb{E}[\mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}] = \boldsymbol{\mu} && (12.37) \\ \mathrm{cov}[\mathbf{x}] &= \mathbb{E}\left[(\mathbf{W}\mathbf{z} + \boldsymbol{\epsilon})(\mathbf{W}\mathbf{z} + \boldsymbol{\epsilon})^{\mathrm{T}}\right] \\ &= \mathbb{E}\left[\mathbf{W}\mathbf{z}\mathbf{z}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\right] + \mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^{\mathrm{T}}] = \mathbf{W}\mathbf{W}^{\mathrm{T}} + \sigma^2\mathbf{I} && (12.38) \end{aligned}$$

where we have used the fact that $\mathbf{z}$ and $\boldsymbol{\epsilon}$ are independent random variables and hence are uncorrelated.

Intuitively, we can think of the distribution $p(\mathbf{x})$ as being defined by taking an isotropic Gaussian 'spray can' and moving it across the principal subspace spraying Gaussian ink with density determined by σ^2 and weighted by the prior distribution. The accumulated ink density gives rise to a 'pancake' shaped distribution representing the marginal density $p(\mathbf{x})$.

The predictive distribution $p(\mathbf{x})$ is governed by the parameters $\boldsymbol{\mu}$, $\mathbf{W}$, and σ^2. However, there is redundancy in this parameterization corresponding to rotations of the latent space coordinates. To see this, consider a matrix $\widetilde{\mathbf{W}} = \mathbf{W}\mathbf{R}$ where $\mathbf{R}$ is an orthogonal matrix. Using the orthogonality property $\mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{I}$, we see that the quantity $\widetilde{\mathbf{W}}\widetilde{\mathbf{W}}^{\mathrm{T}}$ that appears in the covariance matrix $\mathbf{C}$ takes the form

$$\widetilde{\mathbf{W}}\widetilde{\mathbf{W}}^{\mathrm{T}} = \mathbf{W}\mathbf{R}\mathbf{R}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}} = \mathbf{W}\mathbf{W}^{\mathrm{T}} \tag{12.39}$$

and hence is independent of $\mathbf{R}$. Thus there is a whole family of matrices $\widetilde{\mathbf{W}}$ all of which give rise to the same predictive distribution. This invariance can be understood in terms of rotations within the latent space. We shall return to a discussion of the number of independent parameters in this model later.

When we evaluate the predictive distribution, we require $\mathbf{C}^{-1}$, which involves the inversion of a $D \times D$ matrix. The computation required to do this can be reduced by making use of the matrix inversion identity (C.7) to give

$$\mathbf{C}^{-1} = \sigma^{-2}\mathbf{I} - \sigma^{-2}\mathbf{W}\mathbf{M}^{-1}\mathbf{W}^{\mathrm{T}} \tag{12.40}$$

where the $M \times M$ matrix $\mathbf{M}$ is defined by

$$\mathbf{M} = \mathbf{W}^{\mathrm{T}}\mathbf{W} + \sigma^2\mathbf{I}. \tag{12.41}$$

Because we invert $\mathbf{M}$ rather than inverting $\mathbf{C}$ directly, the cost of evaluating $\mathbf{C}^{-1}$ is reduced from $O(D^3)$ to $O(M^3)$.

As well as the predictive distribution $p(\mathbf{x})$, we will also require the posterior distribution $p(\mathbf{z}|\mathbf{x})$, which can again be written down directly using the result (2.116) for linear-Gaussian models to give

Exercise 12.8

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}\left(\mathbf{z}|\mathbf{M}^{-1}\mathbf{W}^{\mathrm{T}}(\mathbf{x} - \boldsymbol{\mu}), \sigma^{-2}\mathbf{M}\right). \tag{12.42}$$

Note that the posterior mean depends on $\mathbf{x}$, whereas the posterior covariance is independent of $\mathbf{x}$.

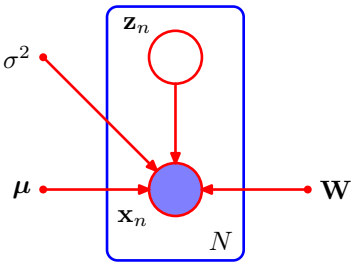


Figure 12.10 The probabilistic PCA model for a data set of N observations of $\mathbf{x}$ can be expressed as a directed graph in which each observation $\mathbf{x}_n$ is associated with a value $\mathbf{z}_n$ of the latent variable.

12.2.1 Maximum likelihood PCA

We next consider the determination of the model parameters using maximum likelihood. Given a data set $\mathbf{X} = \{\mathbf{x}_n\}$ of observed data points, the probabilistic PCA model can be expressed as a directed graph, as shown in Figure 12.10. The corresponding log likelihood function is given, from (12.35), by

$$\begin{aligned}\ln p(\mathbf{X}|\boldsymbol{\mu}, \mathbf{W}, \sigma^2) &= \sum_{n=1}^{N} \ln p(\mathbf{x}_n|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) \\ &= -\frac{ND}{2}\ln(2\pi) - \frac{N}{2}\ln|\mathbf{C}| - \frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}). \quad (12.43)\end{aligned}$$

Setting the derivative of the log likelihood with respect to $\boldsymbol{\mu}$ equal to zero gives the expected result $\boldsymbol{\mu} = \overline{\mathbf{x}}$ where $\overline{\mathbf{x}}$ is the data mean defined by (12.1). Back-substituting we can then write the log likelihood function in the form

$$\ln p(\mathbf{X}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) = -\frac{N}{2}\left\{D\ln(2\pi) + \ln|\mathbf{C}| + \mathrm{Tr}\left(\mathbf{C}^{-1}\mathbf{S}\right)\right\} \quad (12.44)$$

where $\mathbf{S}$ is the data covariance matrix defined by (12.3). Because the log likelihood is a quadratic function of $\boldsymbol{\mu}$, this solution represents the unique maximum, as can be confirmed by computing second derivatives.

Maximization with respect to $\mathbf{W}$ and σ^2 is more complex but nonetheless has an exact closed-form solution. It was shown by Tipping and Bishop (1999b) that all of the stationary points of the log likelihood function can be written as

$$\mathbf{W}_{\mathrm{ML}} = \mathbf{U}_M(\mathbf{L}_M - \sigma^2\mathbf{I})^{1/2}\mathbf{R} \quad (12.45)$$

where $\mathbf{U}_M$ is a $D \times M$ matrix whose columns are given by any subset (of size M) of the eigenvectors of the data covariance matrix $\mathbf{S}$, the $M \times M$ diagonal matrix $\mathbf{L}_M$ has elements given by the corresponding eigenvalues λ_i, and $\mathbf{R}$ is an arbitrary $M \times M$ orthogonal matrix.

Furthermore, Tipping and Bishop (1999b) showed that the *maximum* of the likelihood function is obtained when the M eigenvectors are chosen to be those whose eigenvalues are the M largest (all other solutions being saddle points). A similar result was conjectured independently by Roweis (1998), although no proof was given.

Again, we shall assume that the eigenvectors have been arranged in order of decreasing values of the corresponding eigenvalues, so that the M principal eigenvectors are $\mathbf{u}_1, \ldots, \mathbf{u}_M$. In this case, the columns of $\mathbf{W}$ define the principal subspace of standard PCA. The corresponding maximum likelihood solution for σ^2 is then given by

$$\sigma^2_{\mathrm{ML}} = \frac{1}{D-M} \sum_{i=M+1}^{D} \lambda_i \tag{12.46}$$

so that σ^2_{ML} is the average variance associated with the discarded dimensions.

Because $\mathbf{R}$ is orthogonal, it can be interpreted as a rotation matrix in the M latent space. If we substitute the solution for $\mathbf{W}$ into the expression for $\mathbf{C}$, and make use of the orthogonality property $\mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{I}$, we see that $\mathbf{C}$ is independent of $\mathbf{R}$. This simply says that the predictive density is unchanged by rotations in the latent space as discussed earlier. For the particular case of $\mathbf{R} = \mathbf{I}$, we see that the columns of $\mathbf{W}$ are the principal component eigenvectors scaled by the variance parameters $\sqrt{\lambda_i - \sigma^2}$. The interpretation of these scaling factors is clear once we recognize that for a convolution of independent Gaussian distributions (in this case the latent space distribution and the noise model) the variances are additive. Thus the variance λ_i in the direction of an eigenvector $\mathbf{u}_i$ is composed of the sum of a contribution $\lambda_i - \sigma^2$ from the projection of the unit-variance latent space distribution into data space through the corresponding column of $\mathbf{W}$, plus an isotropic contribution of variance σ^2 which is added in all directions by the noise model.

It is worth taking a moment to study the form of the covariance matrix given by (12.36). Consider the variance of the predictive distribution along some direction specified by the unit vector $\mathbf{v}$, where $\mathbf{v}^{\mathrm{T}}\mathbf{v} = 1$, which is given by $\mathbf{v}^{\mathrm{T}}\mathbf{C}\mathbf{v}$. First suppose that $\mathbf{v}$ is orthogonal to the principal subspace, in other words it is given by some linear combination of the discarded eigenvectors. Then $\mathbf{v}^{\mathrm{T}}\mathbf{U} = \mathbf{0}$ and hence $\mathbf{v}^{\mathrm{T}}\mathbf{C}\mathbf{v} = \sigma^2$. Thus the model predicts a noise variance orthogonal to the principal subspace, which, from (12.46), is just the average of the discarded eigenvalues. Now suppose that $\mathbf{v} = \mathbf{u}_i$ where $\mathbf{u}_i$ is one of the retained eigenvectors defining the principal subspace. Then $\mathbf{v}^{\mathrm{T}}\mathbf{C}\mathbf{v} = (\lambda_i - \sigma^2) + \sigma^2 = \lambda_i$. In other words, this model correctly captures the variance of the data along the principal axes, and approximates the variance in all remaining directions with a single average value σ^2.

One way to construct the maximum likelihood density model would simply be to find the eigenvectors and eigenvalues of the data covariance matrix and then to evaluate $\mathbf{W}$ and σ^2 using the results given above. In this case, we would choose $\mathbf{R} = \mathbf{I}$ for convenience. However, if the maximum likelihood solution is found by numerical optimization of the likelihood function, for instance using an algorithm such as conjugate gradients (Fletcher, 1987; Nocedal and Wright, 1999; Bishop and Nabney, 2008) or through the EM algorithm, then the resulting value of $\mathbf{R}$ is essentially arbitrary. This implies that the columns of $\mathbf{W}$ need not be orthogonal. If an orthogonal basis is required, the matrix $\mathbf{W}$ can be post-processed appropriately (Golub and Van Loan, 1996). Alternatively, the EM algorithm can be modified in such a way as to yield orthonormal principal directions, sorted in descending order of the corresponding eigenvalues, directly (Ahn and Oh, 2003).

Section 12.2.2

The rotational invariance in latent space represents a form of statistical nonidentifiability, analogous to that encountered for mixture models in the case of discrete latent variables. Here there is a continuum of parameters all of which lead to the same predictive density, in contrast to the discrete nonidentifiability associated with component re-labelling in the mixture setting.

If we consider the case of $M = D$, so that there is no reduction of dimensionality, then $\mathbf{U}_M = \mathbf{U}$ and $\mathbf{L}_M = \mathbf{L}$. Making use of the orthogonality properties $\mathbf{U}\mathbf{U}^{\mathrm{T}} = \mathbf{I}$ and $\mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{I}$, we see that the covariance $\mathbf{C}$ of the marginal distribution for $\mathbf{x}$ becomes

$$\mathbf{C} = \mathbf{U}(\mathbf{L} - \sigma^2\mathbf{I})^{1/2}\mathbf{R}\mathbf{R}^{\mathrm{T}}(\mathbf{L} - \sigma^2\mathbf{I})^{1/2}\mathbf{U}^{\mathrm{T}} + \sigma^2\mathbf{I} = \mathbf{U}\mathbf{L}\mathbf{U}^{\mathrm{T}} = \mathbf{S} \tag{12.47}$$

and so we obtain the standard maximum likelihood solution for an unconstrained Gaussian distribution in which the covariance matrix is given by the sample covariance.

Conventional PCA is generally formulated as a projection of points from the D-dimensional data space onto an M-dimensional linear subspace. Probabilistic PCA, however, is most naturally expressed as a mapping from the latent space into the data space via (12.33). For applications such as visualization and data compression, we can reverse this mapping using Bayes' theorem. Any point $\mathbf{x}$ in data space can then be summarized by its posterior mean and covariance in latent space. From (12.42) the mean is given by

$$\mathbb{E}[\mathbf{z}|\mathbf{x}] = \mathbf{M}^{-1}\mathbf{W}_{\mathrm{ML}}^{\mathrm{T}}(\mathbf{x} - \overline{\mathbf{x}}) \tag{12.48}$$

where $\mathbf{M}$ is given by (12.41). This projects to a point in data space given by

$$\mathbf{W}\mathbb{E}[\mathbf{z}|\mathbf{x}] + \boldsymbol{\mu}. \tag{12.49}$$

Section 3.3.1

Note that this takes the same form as the equations for regularized linear regression and is a consequence of maximizing the likelihood function for a linear Gaussian model. Similarly, the posterior covariance is given from (12.42) by $\sigma^2\mathbf{M}^{-1}$ and is independent of $\mathbf{x}$.

If we take the limit $\sigma^2 \to 0$, then the posterior mean reduces to

$$(\mathbf{W}_{\mathrm{ML}}^{\mathrm{T}}\mathbf{W}_{\mathrm{ML}})^{-1}\mathbf{W}_{\mathrm{ML}}^{\mathrm{T}}(\mathbf{x} - \overline{\mathbf{x}}) \tag{12.50}$$

which represents an orthogonal projection of the data point onto the latent space, and so we recover the standard PCA model. The posterior covariance in this limit is zero, however, and the density becomes singular. For $\sigma^2 > 0$, the latent projection is shifted towards the origin, relative to the orthogonal projection.

Exercise 12.11

Exercise 12.12

Finally, we note that an important role for the probabilistic PCA model is in defining a multivariate Gaussian distribution in which the number of degrees of freedom, in other words the number of independent parameters, can be controlled whilst still allowing the model to capture the dominant correlations in the data. Recall that a general Gaussian distribution has $D(D + 1)/2$ independent parameters in its covariance matrix (plus another D parameters in its mean). Thus the number of parameters scales quadratically with D and can become excessive in spaces of high

Section 2.3

If we take M = D, then we obtain the standard maximum likelihood solution for an unconstrained Gaussian distribution in which the covariance matrix is given by the sample covariance.

If we take M = D – 1, then we recover the standard result for a full covariance Gaussian.

由式(12.51)式(12.36)和式(12.46)，M=D时，自由度为 $\frac{D(D+1)}{2}+1$，C=S；M=D-1时，自由度为 $\frac{D(D+1)}{2}$，C=S。一个协方差矩阵的自由度最多为 $\frac{D(D+1)}{2}$，采用 $\frac{D(D+1)}{2}+1$ 与 $\frac{D(D+1)}{2}$ 的模型去拟合数据均对应 unconstrained 的情况，即 C=S。可以看到，M=D 和 M=D-1 时，ML 得到的分布 p(x)（即式(12.35)）是相同的，M=D-1时最后一个主方向的方差为0，由 σ^2 去捕获。然而 M=D 与 M=D-1 时所得 p(x) 相同，但是这两种，即式(12.44)是相同的，二者并不看。

dimensionality. If we restrict the covariance matrix to be diagonal, then it has only D independent parameters, and so the number of parameters now grows linearly with dimensionality. However, it now treats the variables as if they were independent and hence can no longer express any correlations between them. Probabilistic PCA provides an elegant compromise in which the M most significant correlations can be captured while still ensuring that the total number of parameters grows only linearly with D. We can see this by evaluating the number of degrees of freedom in the PPCA model as follows. The covariance matrix $\mathbf{C}$ depends on the parameters $\mathbf{W}$, which has size $D \times M$, and σ^2, giving a total parameter count of $DM+1$. However, we have seen that there is some redundancy in this parameterization associated with rotations of the coordinate system in the latent space. The orthogonal matrix $\mathbf{R}$ that expresses these rotations has size $M \times M$. In the first column of this matrix there are $M-1$ independent parameters, because the column vector must be normalized to unit length. In the second column there are $M-2$ independent parameters, because the column must be normalized and also must be orthogonal to the previous column, and so on. Summing this arithmetic series, we see that $\mathbf{R}$ has a total of $M(M-1)/2$ independent parameters. Thus the number of degrees of freedom in the covariance matrix $\mathbf{C}$ is given by

PPCA能够兼顾二者，即能捕捉数据的特性，又使模型参数较少。

$$DM+1-M(M-1)/2. \tag{12.51}$$

The number of independent parameters in this model therefore only grows linearly with D, for fixed M. If we take $M = D-1$, then we recover the standard result for a full covariance Gaussian. In this case, the variance along $D-1$ linearly independent directions is controlled by the columns of $\mathbf{W}$, and the variance along the remaining direction is given by σ^2. If $M = 0$, the model is equivalent to the isotropic covariance case.

full covariance Gaussian 和 isotropic covariance Gaussian 可视为 PPCA 的特例

Exercise 12.14

12.2.2 EM algorithm for PCA

As we have seen, the probabilistic PCA model can be expressed in terms of a marginalization over a continuous latent space $\mathbf{z}$ in which for each data point $\mathbf{x}_n$, there is a corresponding latent variable $\mathbf{z}_n$. We can therefore make use of the EM algorithm to find maximum likelihood estimates of the model parameters. This may seem rather pointless because we have already obtained an exact closed-form solution for the maximum likelihood parameter values. However, in spaces of high dimensionality, there may be computational advantages in using an iterative EM procedure rather than working directly with the sample covariance matrix. This EM procedure can also be extended to the factor analysis model, for which there is no closed-form solution. Finally, it allows missing data to be handled in a principled way.

虽然有直接解析解，但使用EM算法也是有好处的

Section 12.2.4

We can derive the EM algorithm for probabilistic PCA by following the general framework for EM. Thus we write down the complete-data log likelihood and take its expectation with respect to the posterior distribution of the latent ~~distribution~~ variable evaluated using 'old' parameter values. Maximization of this expected complete-data log likelihood then yields the 'new' parameter values. Because the data points

Section 9.4

are assumed independent, the complete-data log likelihood function takes the form

$$\ln p\left(\mathbf{X}, \mathbf{Z}|\boldsymbol{\mu}, \mathbf{W}, \sigma^2\right) = \sum_{n=1}^{N} \{\ln p(\mathbf{x}_n|\mathbf{z}_n) + \ln p(\mathbf{z}_n)\} \tag{12.52}$$

where the n^{th} row of the matrix $\mathbf{Z}$ is given by $\mathbf{z}_n$. We already know that the exact maximum likelihood solution for $\boldsymbol{\mu}$ is given by the sample mean $\overline{\mathbf{x}}$ defined by (12.1), and it is convenient to substitute for $\boldsymbol{\mu}$ at this stage. Making use of the expressions (12.31) and (12.32) for the latent and conditional distributions, respectively, and taking the expectation with respect to the posterior distribution over the latent variables, we obtain

$$\begin{aligned}\mathbb{E}[\ln p\left(\mathbf{X}, \mathbf{Z}|\boldsymbol{\mu}, \mathbf{W}, \sigma^2\right)] = -\sum_{n=1}^{N} \Bigg\{ & \frac{D}{2}\ln(2\pi\sigma^2) + \frac{1}{2}\mathrm{Tr}\left(\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}]\right) \\ & + \frac{1}{2\sigma^2}\|\mathbf{x}_n - \boldsymbol{\mu}\|^2 - \frac{1}{\sigma^2}\mathbb{E}[\mathbf{z}_n]^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}(\mathbf{x}_n - \boldsymbol{\mu}) \\ & + \frac{1}{2\sigma^2}\mathrm{Tr}\left(\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}]\mathbf{W}^{\mathrm{T}}\mathbf{W}\right)\Bigg\}. \end{aligned} \tag{12.53}$$

Note that this depends on the posterior distribution only through the sufficient statistics of the Gaussian. Thus in the E step, we use the old parameter values to evaluate

$$\mathbb{E}[\mathbf{z}_n] = \mathbf{M}^{-1}\mathbf{W}^{\mathrm{T}}(\mathbf{x}_n - \overline{\mathbf{x}}) \tag{12.54}$$

$$\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}] = \sigma^2\mathbf{M}^{-1} + \mathbb{E}[\mathbf{z}_n]\mathbb{E}[\mathbf{z}_n]^{\mathrm{T}} \tag{12.55}$$

which follow directly from the posterior distribution (12.42) together with the standard result $\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}] = \mathrm{cov}[\mathbf{z}_n] + \mathbb{E}[\mathbf{z}_n]\mathbb{E}[\mathbf{z}_n]^{\mathrm{T}}$. Here $\mathbf{M}$ is defined by (12.41).

Exercise 12.15 In the M step, we maximize with respect to $\mathbf{W}$ and σ^2, keeping the posterior statistics fixed. Maximization with respect to σ^2 is straightforward. For the maximization with respect to $\mathbf{W}$ we make use of (C.24), and obtain the M step equations

$$\mathbf{W}_{\text{new}} = \left[\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})\mathbb{E}[\mathbf{z}_n]^{\mathrm{T}}\right]\left[\sum_{n=1}^{N}\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}]\right]^{-1} \tag{12.56}$$

$$\begin{aligned}\sigma^2_{\text{new}} = \frac{1}{ND}\sum_{n=1}^{N}\Big\{ & \|\mathbf{x}_n - \overline{\mathbf{x}}\|^2 - 2\mathbb{E}[\mathbf{z}_n]^{\mathrm{T}}\mathbf{W}_{\text{new}}^{\mathrm{T}}(\mathbf{x}_n - \overline{\mathbf{x}}) \\ & + \mathrm{Tr}\left(\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}]\mathbf{W}_{\text{new}}^{\mathrm{T}}\mathbf{W}_{\text{new}}\right)\Big\}. \end{aligned} \tag{12.57}$$

The EM algorithm for probabilistic PCA proceeds by initializing the parameters and then alternately computing the sufficient statistics of the latent space posterior distribution using (12.54) and (12.55) in the E step and revising the parameter values using (12.56) and (12.57) in the M step.

One of the benefits of the EM algorithm for PCA is computational efficiency for large-scale applications (Roweis, 1998). Unlike conventional PCA based on an

eigenvector decomposition of the sample covariance matrix, the EM approach is iterative and so might appear to be less attractive. However, each cycle of the EM algorithm can be computationally much more efficient than conventional PCA in spaces of high dimensionality. To see this, we note that the eigendecomposition of the covariance matrix requires $O(D^3)$ computation. Often we are interested only in the first M eigenvectors and their corresponding eigenvalues, in which case we can use algorithms that are $O(MD^2)$. However, the evaluation of the covariance matrix itself takes $O(ND^2)$ computations, where N is the number of data points. Algorithms such as the snapshot method (Sirovich, 1987), which assume that the eigenvectors are linear combinations of the data vectors, avoid direct evaluation of the covariance matrix but are $O(N^3)$ and hence unsuited to large data sets. The EM algorithm described here also does not construct the covariance matrix explicitly. Instead, the most computationally demanding steps are those involving sums over the data set that are $O(NDM)$. For large D, and $M \ll D$, this can be a significant saving compared to $O(ND^2)$ and can offset the iterative nature of the EM algorithm.

Note that this EM algorithm can be implemented in an on-line form in which each D-dimensional data point is read in and processed and then discarded before the next data point is considered. To see this, note that the quantities evaluated in the E step (an M-dimensional vector and an $M \times M$ matrix) can be computed for each data point separately, and in the M step we need to accumulate sums over data points, which we can do incrementally. This approach can be advantageous if both N and D are large.

Because we now have a fully probabilistic model for PCA, we can deal with missing data, provided that it is missing at random, by marginalizing over the distribution of the unobserved variables. Again these missing values can be treated using the EM algorithm. We give an example of the use of this approach for data visualization in Figure 12.11.

Another elegant feature of the EM approach is that we can take the limit $\sigma^2 \to 0$, corresponding to standard PCA, and still obtain a valid EM-like algorithm (Roweis, 1998). From (12.55), we see that the only quantity we need to compute in the E step is $\mathbb{E}[\mathbf{z}_n]$. Furthermore, the M step is simplified because $\mathbf{M} = \mathbf{W}^{\mathrm{T}}\mathbf{W}$. To emphasize the simplicity of the algorithm, let us define $\widetilde{\mathbf{X}}$ to be a matrix of size $N \times D$ whose n^{th} row is given by the vector $\mathbf{x}_n - \overline{\mathbf{x}}$ and similarly define $\mathbf{\Omega}$ to be a matrix of size ~~$D \times M$~~ whose n^{th} ~~row~~ column is given by the vector $\mathbb{E}[\mathbf{z}_n]$. The E step (12.54) of the EM algorithm for PCA then becomes

$$\mathbf{\Omega} = (\mathbf{W}_{\mathrm{old}}^{\mathrm{T}}\mathbf{W}_{\mathrm{old}})^{-1}\mathbf{W}_{\mathrm{old}}^{\mathrm{T}}\widetilde{\mathbf{X}}^{\mathrm{T}} \tag{12.58}$$

and the M step (12.56) takes the form

$$\mathbf{W}_{\mathrm{new}} = \widetilde{\mathbf{X}}^{\mathrm{T}}\mathbf{\Omega}^{\mathrm{T}}(\mathbf{\Omega}\mathbf{\Omega}^{\mathrm{T}})^{-1}. \tag{12.59}$$

Again these can be implemented in an on-line form. These equations have a simple interpretation as follows. From our earlier discussion, we see that the E step involves an orthogonal projection of the data points onto the current estimate for the principal subspace. Correspondingly, the M step represents a re-estimation of the principal

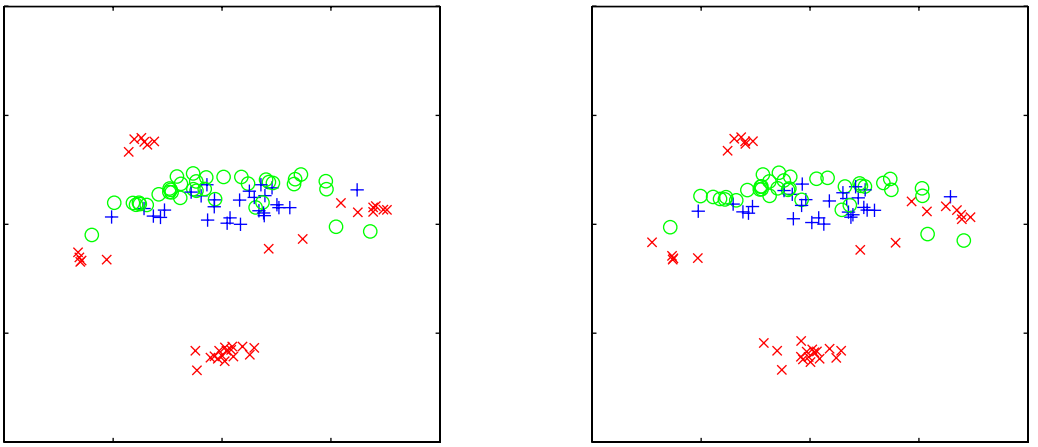


Figure 12.11 Probabilistic PCA visualization of a portion of the oil flow data set for the first 100 data points. The left-hand plot shows the posterior mean projections of the data points on the principal subspace. The right-hand plot is obtained by first randomly omitting 30% of the variable values and then using EM to handle the missing values. Note that each data point then has at least one missing measurement but that the plot is very similar to the one obtained without missing values.

subspace to minimize the squared reconstruction error in which the projections are fixed.

Exercise 12.17

We can give a simple physical analogy for this EM algorithm, which is easily visualized for $D = 2$ and $M = 1$. Consider a collection of data points in two dimensions, and let the one-dimensional principal subspace be represented by a solid rod. Now attach each data point to the rod via a spring obeying Hooke's law (stored energy is proportional to the square of the spring's length). In the E step, we keep the rod fixed and allow the attachment points to slide up and down the rod so as to minimize the energy. This causes each attachment point (independently) to position itself at the orthogonal projection of the corresponding data point onto the rod. In the M step, we keep the attachment points fixed and then release the rod and allow it to move to the minimum energy position. The E and M steps are then repeated until a suitable convergence criterion is satisfied, as is illustrated in Figure 12.12.

12.2.3 Bayesian PCA

So far in our discussion of PCA, we have assumed that the value M for the dimensionality of the principal subspace is given. In practice, we must choose a suitable value according to the application. For visualization, we generally choose $M = 2$, whereas for other applications the appropriate choice for M may be less clear. One approach is to plot the eigenvalue spectrum for the data set, analogous to the example in Figure 12.4 for the off-line digits data set, and look to see if the eigenvalues naturally form two groups comprising a set of small values separated by a significant gap from a set of relatively large values, indicating a natural choice for M. In practice, such a gap is often not seen.

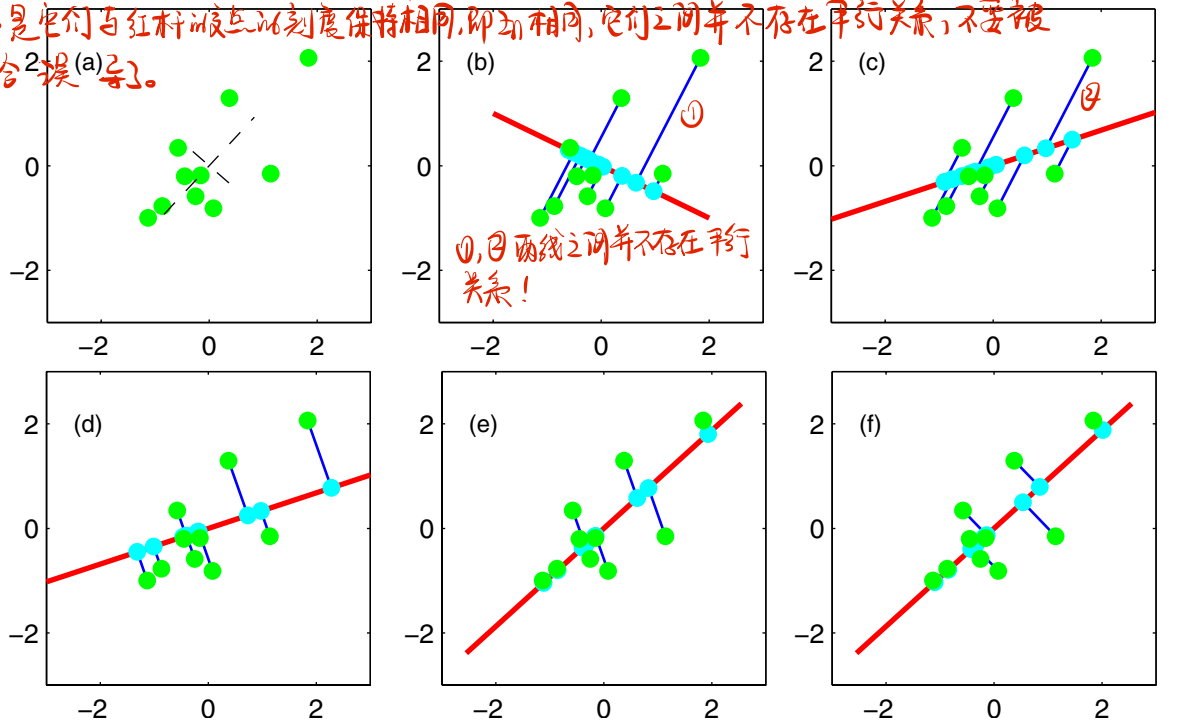


Figure 12.12 Synthetic data illustrating the EM algorithm for PCA defined by (12.58) and (12.59). (a) A data set $\mathbf{X}$ with the data points shown in green, together with the true principal components (shown as eigenvectors scaled by the square roots of the eigenvalues). (b) Initial configuration of the principal subspace defined by $\mathbf{W}$, shown in red, together with the projections of the latent points $\mathbf{Z}$ into the data space, given by $\mathbf{ZW}^{\mathrm{T}}$, shown in cyan. (c) After one M step, the latent space has been updated with $\mathbf{Z}$ held fixed. (d) After the successive E step, the values of $\mathbf{Z}$ have been updated, giving orthogonal projections, with $\mathbf{W}$ held fixed. (e) After the second M step. (f) After the second E step.

Section 1.3

Because the probabilistic PCA model has a well-defined likelihood function, we could employ cross-validation to determine the value of dimensionality by selecting the largest log likelihood on a validation data set. Such an approach, however, can become computationally costly, particularly if we consider a probabilistic mixture of PCA models (Tipping and Bishop, 1999a) in which we seek to determine the appropriate dimensionality separately for each component in the mixture.

Given that we have a probabilistic formulation of PCA, it seems natural to seek a Bayesian approach to model selection. To do this, we need to marginalize out the model parameters $\boldsymbol{\mu}$, $\mathbf{W}$, and σ^2 with respect to appropriate prior distributions. This can be done by using a variational framework to approximate the analytically intractable marginalizations (Bishop, 1999b). The marginal likelihood values, given by the variational lower bound, can then be compared for a range of different values of M and the value giving the largest marginal likelihood selected.

Here we consider a simpler approach ~~introduced by~~ based on the *evidence ap-*

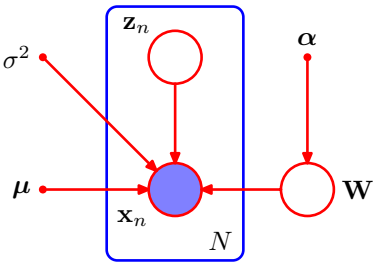


Figure 12.13 Probabilistic graphical model for Bayesian PCA in which the distribution over the parameter matrix $\mathbf{W}$ is governed by a vector $\boldsymbol{\alpha}$ of hyperparameters.

proximation, which is appropriate when the number of data points is relatively large and the corresponding posterior distribution is tightly peaked (Bishop, 1999a). It involves a specific choice of prior over $\mathbf{W}$ that allows surplus dimensions in the principal subspace to be pruned out of the model. This corresponds to an example of *automatic relevance determination*, or *ARD*, discussed in Section 7.2.2. Specifically, we define an independent Gaussian prior over each column of $\mathbf{W}$, which represent the vectors defining the principal subspace. Each such Gaussian has an independent variance governed by a precision hyperparameter α_i so that

$$p(\mathbf{W}|\boldsymbol{\alpha}) = \prod_{i=1}^{M} \left(\frac{\alpha_i}{2\pi}\right)^{D/2} \exp\left\{-\frac{1}{2}\alpha_i \mathbf{w}_i^{\mathrm{T}}\mathbf{w}_i\right\} \tag{12.60}$$

where $\mathbf{w}_i$ is the i^{th} column of $\mathbf{W}$. The resulting model can be represented using the directed graph shown in Figure 12.13.

The values for α_i will be found iteratively by maximizing the marginal likelihood function in which $\mathbf{W}$ has been integrated out. As a result of this optimization, some of the α_i may be driven to infinity, with the corresponding parameters vector $\mathbf{w}_i$ being driven to zero (the posterior distribution becomes a delta function at the origin) giving a sparse solution. The effective dimensionality of the principal subspace is then determined by the number of finite α_i values, and the corresponding vectors $\mathbf{w}_i$ can be thought of as 'relevant' for modelling the data distribution. In this way, the Bayesian approach is automatically making the trade-off between improving the fit to the data, by using a larger number of vectors $\mathbf{w}_i$ with their corresponding eigenvalues λ_i each tuned to the data, and reducing the complexity of the model by suppressing some of the $\mathbf{w}_i$ vectors. The origins of this sparsity were discussed earlier in the context of relevance vector machines.

Section 7.2

The values of α_i are re-estimated during training by maximizing the ~~log~~ marginal likelihood given by

$$p(\mathbf{X}|\boldsymbol{\alpha}, \boldsymbol{\mu}, \sigma^2) = \int p(\mathbf{X}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) p(\mathbf{W}|\boldsymbol{\alpha})\, \mathrm{d}\mathbf{W} \tag{12.61}$$

where the log of $p(\mathbf{X}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2)$ is given by (12.43). Note that for simplicity we also treat $\boldsymbol{\mu}$ and σ^2 as parameters to be estimated, rather than defining priors over these parameters.

Section 4.4

Section 3.5.3

Because this integration is intractable, we make use of the Laplace approximation. If we assume that the posterior distribution is sharply peaked, as will occur for sufficiently large data sets, then the re-estimation equations obtained by maximizing the marginal likelihood with respect to α_i take the simple form

$$\alpha_i^{\text{new}} = \frac{D}{\mathbf{w}_i^{\mathrm{T}}\mathbf{w}_i} \tag{12.62}$$

which follows from (3.98), noting that the dimensionality of $\mathbf{w}_i$ is D. These re-estimations are interleaved with the EM algorithm updates for determining $\mathbf{W}$ and σ^2. The E-step equations are again given by (12.54) and (12.55). Similarly, the M-step equation for σ^2 is again given by (12.57). The only change is to the M-step equation for $\mathbf{W}$, which is modified to give

$$\mathbf{W}_{\text{new}} = \left[\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})\mathbb{E}[\mathbf{z}_n]^{\mathrm{T}}\right]\left[\sum_{n=1}^{N}\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}] + \sigma^2\mathbf{A}\right]^{-1} \tag{12.63}$$

where $\mathbf{A} = \text{diag}(\alpha_i)$. The value of $\boldsymbol{\mu}$ is given by the sample mean, as before.

If we choose $M = D - 1$ then, if all α_i values are finite, the model represents a full-covariance Gaussian, while if all the α_i go to infinity the model is equivalent to an isotropic Gaussian, and so the model can encompass all permissible values for the effective dimensionality of the principal subspace. It is also possible to consider smaller values of M, which will save on computational cost but which will limit the maximum dimensionality of the subspace. A comparison of the results of this algorithm with standard probabilistic PCA is shown in Figure 12.14.

Bayesian PCA provides an opportunity to illustrate the Gibbs sampling algorithm discussed in Section 11.3. Figure 12.15 shows an example of the samples from the hyperparameters $\ln \alpha_i$ for a data set in $D = 4$ dimensions in which the dimensionality of the latent space is $M = 3$ but in which the data set is generated from a probabilistic PCA model having one direction of high variance, with the remaining directions comprising low variance noise. This result shows clearly the presence of three distinct modes in the posterior distribution. At each step of the iteration, one of the hyperparameters has a small value and the remaining two have large values, so that two of the three latent variables are suppressed. During the course of the Gibbs sampling, the solution makes sharp transitions between the three modes.

The model described here involves a prior only over the matrix $\mathbf{W}$. A fully Bayesian treatment of PCA, including priors over $\boldsymbol{\mu}$, σ^2, and $\boldsymbol{\alpha}$, and solved using variational methods, is described in Bishop (1999b). For a discussion of various Bayesian approaches to determining the appropriate dimensionality for a PCA model, see Minka (2001c).

12.2.4 Factor analysis

Factor analysis is a linear-Gaussian latent variable model that is closely related to probabilistic PCA. Its definition differs from that of probabilistic PCA only in that the conditional distribution of the observed variable $\mathbf{x}$ given the latent variable $\mathbf{z}$ is

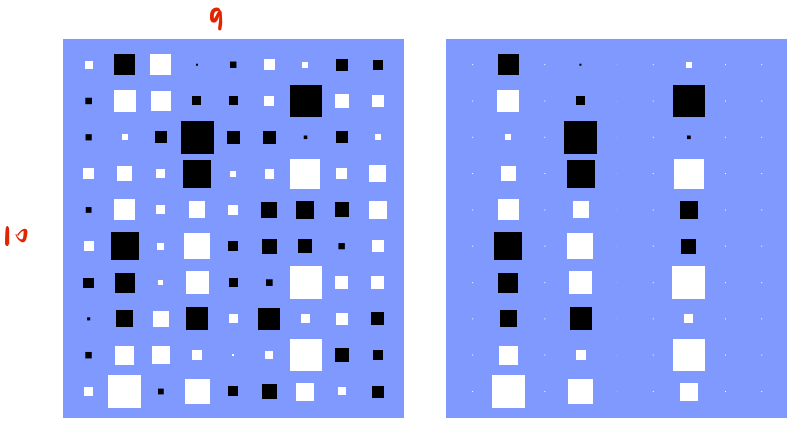


Figure 12.14 'Hinton' diagrams of the matrix $\mathbf{W}$ in which each element of the matrix is depicted as a square (white for positive and black for negative values) whose area is proportional to the magnitude of that element. The synthetic data set comprises 300 data points in $D = 10$ dimensions sampled from a Gaussian distribution having standard deviation 1.0 in 3 directions and standard deviation 0.5 in the remaining 7 directions for a data set in $D = 10$ dimensions having $M = 3$ directions with larger variance than the remaining 7 directions. The left-hand plot shows the result from maximum likelihood probabilistic PCA, and the ~~left~~ right-hand plot shows the corresponding result from Bayesian PCA. We see how the Bayesian model is able to discover the appropriate dimensionality by suppressing the 6 surplus degrees of freedom.

taken to have a diagonal rather than an isotropic covariance so that

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \boldsymbol{\Psi}) \tag{12.64}$$

where $\boldsymbol{\Psi}$ is a $D \times D$ diagonal matrix. Note that the factor analysis model, in common with probabilistic PCA, assumes that the observed variables $x_1, \ldots, x_D$ are independent, given the latent variable $\mathbf{z}$. In essence, the factor analysis model is explaining the observed covariance structure of the data by representing the independent variance associated with each coordinate in the matrix $\boldsymbol{\Psi}$ and capturing the covariance between variables in the matrix $\mathbf{W}$. In the factor analysis literature, the columns of $\mathbf{W}$, which capture the correlations between observed variables, are called *factor loadings*, and the diagonal elements of $\boldsymbol{\Psi}$, which represent the independent noise variances for each of the variables, are called *uniquenesses*.

The origins of factor analysis are as old as those of PCA, and discussions of factor analysis can be found in the books by Everitt (1984), Bartholomew (1987), and Basilevsky (1994). Links between factor analysis and PCA were investigated by Lawley (1953) and Anderson (1963) who showed that at stationary points of the likelihood function, for a factor analysis model with $\boldsymbol{\Psi} = \sigma^2\mathbf{I}$, the columns of $\mathbf{W}$ are scaled eigenvectors of the sample covariance matrix, and σ^2 is the average of the discarded eigenvalues. Later, Tipping and Bishop (1999b) showed that the maximum of the log likelihood function occurs when the eigenvectors comprising $\mathbf{W}$ are chosen to be the principal eigenvectors.

Making use of (2.115), we see that the marginal distribution for the observed

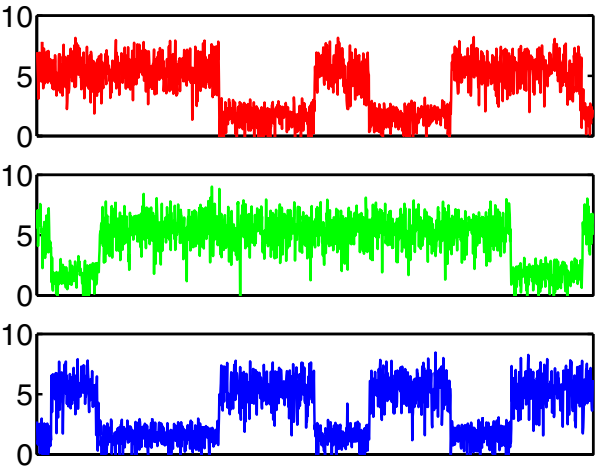


Figure 12.15 Gibbs sampling for Bayesian PCA showing plots of $\ln \alpha_i$ versus iteration number for three α values, showing transitions between the three modes of the posterior distribution.

variable is given by $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{C})$ where now

$$\mathbf{C} = \mathbf{W}\mathbf{W}^{\mathrm{T}} + \boldsymbol{\Psi}. \tag{12.65}$$

Exercise 12.19 As with probabilistic PCA, this model is invariant to rotations in the latent space.

Historically, factor analysis has been the subject of controversy when attempts have been made to place an interpretation on the individual factors (the coordinates in $\mathbf{z}$-space), which has proven problematic due to the nonidentifiability of factor analysis associated with rotations in this space. From our perspective, however, we shall view factor analysis as a form of latent variable density model, in which the form of the latent space is of interest but not the particular choice of coordinates used to describe it. If we wish to remove the degeneracy associated with latent space rotations, we must consider non-Gaussian latent variable distributions, giving rise to independent component analysis (ICA) models. *Section 12.4*

We can determine the parameters $\boldsymbol{\mu}$, $\mathbf{W}$, and $\boldsymbol{\Psi}$ in the factor analysis model by maximum likelihood. The solution for $\boldsymbol{\mu}$ is again given by the sample mean. However, unlike probabilistic PCA, there is no longer a closed-form maximum likelihood solution for $\mathbf{W}$, which must therefore be found iteratively. Because factor analysis is a latent variable model, this can be done using an EM algorithm (Rubin and Thayer, 1982) that is analogous to the one used for probabilistic PCA. Specifically, the E-step equations are given by *Exercise 12.21*

$$\mathbb{E}[\mathbf{z}_n] = \mathbf{G}\mathbf{W}^{\mathrm{T}}\boldsymbol{\Psi}^{-1}(\mathbf{x}_n - \overline{\mathbf{x}}) \tag{12.66}$$

$$\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}] = \mathbf{G} + \mathbb{E}[\mathbf{z}_n]\mathbb{E}[\mathbf{z}_n]^{\mathrm{T}} \tag{12.67}$$

where we have defined

$$\mathbf{G} = (\mathbf{I} + \mathbf{W}^{\mathrm{T}}\boldsymbol{\Psi}^{-1}\mathbf{W})^{-1}. \tag{12.68}$$

Note that this is expressed in a form that involves inversion of matrices of size $M \times M$ rather than $D \times D$ (except for the $D \times D$ diagonal matrix $\boldsymbol{\Psi}$ whose inverse is trivial

to compute in $O(D)$ steps), which is convenient because often $M \ll D$. Similarly, the M-step equations take the form

Exercise 12.22

$$\mathbf{W}_{\text{new}} = \left[\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})\mathbb{E}[\mathbf{z}_n]^{\mathrm{T}}\right]\left[\sum_{n=1}^{N}\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}]\right]^{-1} \tag{12.69}$$

$$\mathbf{\Psi}_{\text{new}} = \text{diag}\left\{\mathbf{S} - \mathbf{W}_{\text{new}}\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[\mathbf{z}_n](\mathbf{x}_n - \bar{\mathbf{x}})^{\mathrm{T}}\right\} \tag{12.70}$$

where the 'diag' operator sets all of the nondiagonal elements of a matrix to zero. A Bayesian treatment of the factor analysis model can be obtained by a straightforward application of the techniques discussed in this book.

Exercise 12.25

Another difference between probabilistic PCA and factor analysis concerns their different behaviour under transformations of the data set. For PCA and probabilistic PCA, if we rotate the coordinate system in data space, then we obtain exactly the same fit to the data but with the $\mathbf{W}$ matrix transformed by the corresponding rotation matrix. However, for factor analysis, the analogous property is that if we make a component-wise re-scaling of the data vectors, then this is absorbed into a corresponding re-scaling of the elements of $\mathbf{\Psi}$.

12.3. Kernel PCA

In Chapter 6, we saw how the technique of kernel substitution allows us to take an algorithm expressed in terms of scalar products of the form $\mathbf{x}^{\mathrm{T}}\mathbf{x}'$ and generalize that algorithm by replacing the scalar products with a nonlinear kernel. Here we apply this technique of kernel substitution to principal component analysis, thereby obtaining a nonlinear generalization called *kernel PCA* (Schölkopf *et al.*, 1998).

Consider a data set $\{\mathbf{x}_n\}$ of observations, where $n = 1, \ldots, N$, in a space of dimensionality D. In order to keep the notation uncluttered, we shall assume that we have already subtracted the sample mean from each of the vectors $\mathbf{x}_n$, so that $\sum_n \mathbf{x}_n = \mathbf{0}$. The first step is to express conventional PCA in such a form that the data vectors $\{\mathbf{x}_n\}$ appear only in the form of the scalar products $\mathbf{x}_n^{\mathrm{T}}\mathbf{x}_m$. Recall that the principal components are defined by the eigenvectors $\mathbf{u}_i$ of the covariance matrix

$$\mathbf{S}\mathbf{u}_i = \lambda_i\mathbf{u}_i \tag{12.71}$$

where $i = 1, \ldots, D$. Here the $D \times D$ sample covariance matrix $\mathbf{S}$ is defined by

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^{\mathrm{T}}, \tag{12.72}$$

and the eigenvectors are normalized such that $\mathbf{u}_i^{\mathrm{T}}\mathbf{u}_i = 1$.

Now consider a nonlinear transformation $\boldsymbol{\phi}(\mathbf{x})$ into an M-dimensional feature space, so that each data point $\mathbf{x}_n$ is thereby projected onto a point $\boldsymbol{\phi}(\mathbf{x}_n)$. We can

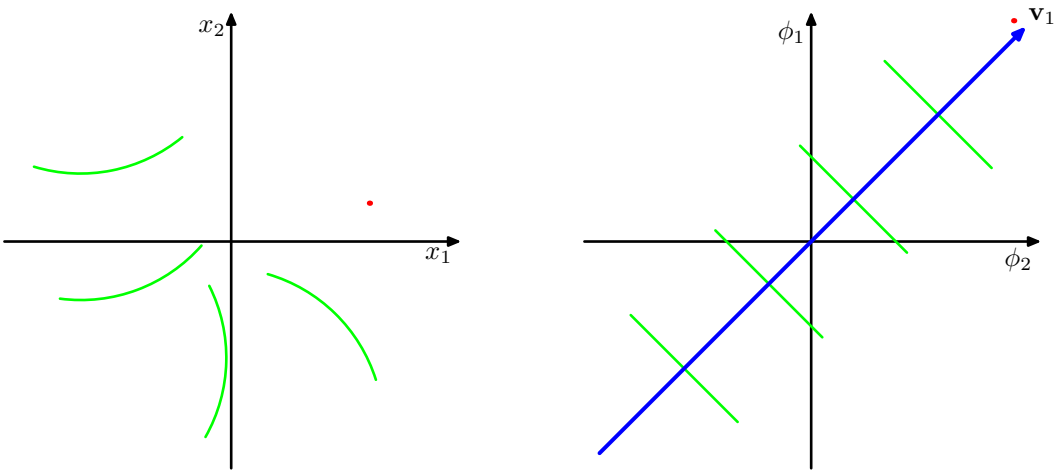


Figure 12.16 Schematic illustration of kernel PCA. A data set in the original data space (left-hand plot) is projected by a nonlinear transformation $\phi(\mathbf{x})$ into a feature space (right-hand plot). By performing PCA in the feature space, we obtain the principal components, of which the first is shown in blue and is denoted by the vector $\mathbf{v}_1$. The green lines in feature space indicate the linear projections onto the first principal component, which correspond to nonlinear projections in the original data space. Note that in general it is not possible to represent the nonlinear principal component by a vector in $\mathbf{x}$ space.

now perform standard PCA in the feature space, which implicitly defines a nonlinear principal component model in the original data space, as illustrated in Figure 12.16. For the moment, let us assume that the projected data set also has zero mean, so that $\sum_n \boldsymbol{\phi}(\mathbf{x}_n) = \mathbf{0}$. We shall return to this point shortly. The $M \times M$ sample covariance matrix in feature space is given by

$$\mathbf{C} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\phi}(\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}} \tag{12.73}$$

and its eigenvector expansion is defined by

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{12.74}$$

$i = 1, \ldots, M$. Our goal is to solve this eigenvalue problem without having to work explicitly in the feature space. From the definition of $\mathbf{C}$, the eigenvector equations tells us that $\mathbf{v}_i$ satisfies

$$\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\phi}(\mathbf{x}_n)\left\{\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\mathbf{v}_i\right\} = \lambda_i \mathbf{v}_i \tag{12.75}$$

and so we see that (provided $\lambda_i > 0$) the vector $\mathbf{v}_i$ is given by a linear combination of the $\boldsymbol{\phi}(\mathbf{x}_n)$ and so can be written in the form

$$\mathbf{v}_i = \sum_{n=1}^{N} a_{in}\boldsymbol{\phi}(\mathbf{x}_n). \tag{12.76}$$

Substituting this expansion back into the eigenvector equation, we obtain

$$\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{\phi}(\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\sum_{m=1}^{N}a_{im}\boldsymbol{\phi}(\mathbf{x}_m)=\lambda_i\sum_{n=1}^{N}a_{in}\boldsymbol{\phi}(\mathbf{x}_n). \tag{12.77}$$

The key step is now to express this in terms of the kernel function $k(\mathbf{x}_n,\mathbf{x}_m)=\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_m)$, which we do by multiplying both sides by $\boldsymbol{\phi}(\mathbf{x}_l)^{\mathrm{T}}$ to give

$$\frac{1}{N}\sum_{n=1}^{N}k(\mathbf{x}_l,\mathbf{x}_n)\sum_{m=1}^{N}a_{im}k(\mathbf{x}_n,\mathbf{x}_m)=\lambda_i\sum_{n=1}^{N}a_{in}k(\mathbf{x}_l,\mathbf{x}_n). \tag{12.78}$$

This can be written in matrix notation as

$$\mathbf{K}^2\mathbf{a}_i=\lambda_i N\mathbf{K}\mathbf{a}_i \tag{12.79}$$

where $\mathbf{a}_i$ is an N-dimensional column vector with elements a_{in} for $n=1,\ldots,N$. We can find solutions for $\mathbf{a}_i$ by solving the following eigenvalue problem

$$\mathbf{K}\mathbf{a}_i=\lambda_i N\mathbf{a}_i \tag{12.80}$$

in which we have removed a factor of $\mathbf{K}$ from both sides of (12.79). Note that the solutions of (12.79) and (12.80) differ only by eigenvectors of $\mathbf{K}$ having zero eigenvalues that do not affect the principal components projection.

Exercise 12.26

The normalization condition for the coefficients $\mathbf{a}_i$ is obtained by requiring that the eigenvectors in feature space be normalized. Using (12.76) and (12.80), we have

$$1=\mathbf{v}_i^{\mathrm{T}}\mathbf{v}_i=\sum_{n=1}^{N}\sum_{m=1}^{N}a_{in}a_{im}\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_m)=\mathbf{a}_i^{\mathrm{T}}\mathbf{K}\mathbf{a}_i=\lambda_i N\mathbf{a}_i^{\mathrm{T}}\mathbf{a}_i. \tag{12.81}$$

Having solved the eigenvector problem, the resulting principal component projections can then also be cast in terms of the kernel function so that, using (12.76), the projection of a point $\mathbf{x}$ onto eigenvector i is given by

$$y_i(\mathbf{x})=\boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\mathbf{v}_i=\sum_{n=1}^{N}a_{in}\boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)=\sum_{n=1}^{N}a_{in}k(\mathbf{x},\mathbf{x}_n) \tag{12.82}$$

and so again is expressed in terms of the kernel function.

In the original D-dimensional $\mathbf{x}$ space there are D orthogonal eigenvectors and hence we can find at most D linear principal components. The dimensionality M of the feature space, however, can be much larger than D (even infinite), and thus we can find a number of nonlinear principal components that can exceed D. Note, however, that the number of nonzero eigenvalues cannot exceed the number N of data points, because (even if $M>N$) the covariance matrix in feature space has rank at most equal to N. This is reflected in the fact that kernel PCA involves the eigenvector expansion of the $N\times N$ matrix $\mathbf{K}$.

So far we have assumed that the projected data set given by $\phi(\mathbf{x}_n)$ has zero mean, which in general will not be the case. We cannot simply compute and then subtract off the mean, since we wish to avoid working directly in feature space, and so again, we formulate the algorithm purely in terms of the kernel function. The projected data points after centralizing, denoted $\widetilde{\phi}(\mathbf{x}_n)$, are given by

$$\widetilde{\phi}(\mathbf{x}_n) = \phi(\mathbf{x}_n) - \frac{1}{N}\sum_{l=1}^{N}\phi(\mathbf{x}_l) \tag{12.83}$$

and the corresponding elements of the Gram matrix are given by

$$\begin{aligned}
\widetilde{K}_{nm} &= \widetilde{\phi}(\mathbf{x}_n)^{\mathrm{T}}\widetilde{\phi}(\mathbf{x}_m) \\
&= \phi(\mathbf{x}_n)^{\mathrm{T}}\phi(\mathbf{x}_m) - \frac{1}{N}\sum_{l=1}^{N}\phi(\mathbf{x}_n)^{\mathrm{T}}\phi(\mathbf{x}_l) \\
&\quad -\frac{1}{N}\sum_{l=1}^{N}\phi(\mathbf{x}_l)^{\mathrm{T}}\phi(\mathbf{x}_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{l=1}^{N}\phi(\mathbf{x}_j)^{\mathrm{T}}\phi(\mathbf{x}_l) \\
&= k(\mathbf{x}_n,\mathbf{x}_m) - \frac{1}{N}\sum_{l=1}^{N}k(\mathbf{x}_l,\mathbf{x}_m) \\
&\quad -\frac{1}{N}\sum_{l=1}^{N}k(\mathbf{x}_n,\mathbf{x}_l) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{l=1}^{N}k(\mathbf{x}_j,\mathbf{x}_l).
\end{aligned} \tag{12.84}$$

This can be expressed in matrix notation as

$$\widetilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_N\mathbf{K} - \mathbf{K}\mathbf{1}_N + \mathbf{1}_N\mathbf{K}\mathbf{1}_N \tag{12.85}$$

where $\mathbf{1}_N$ denotes the $N \times N$ matrix in which every element takes the value $1/N$. Thus we can evaluate $\widetilde{\mathbf{K}}$ using only the kernel function and then use $\widetilde{\mathbf{K}}$ to determine the eigenvalues and eigenvectors. Note that the standard PCA algorithm is recovered as a special case if we use a linear kernel $k(\mathbf{x},\mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{x}'$. Figure 12.17 shows an example of kernel PCA applied to a synthetic data set (Schölkopf *et al.*, 1998). Here a 'Gaussian' kernel of the form

Exercise 12.27

$$k(\mathbf{x},\mathbf{x}') = \exp(-\|\mathbf{x}-\mathbf{x}'\|^2/0.1) \tag{12.86}$$

is applied to a synthetic data set. The lines correspond to contours along which the projection onto the corresponding principal component, defined by

$$\phi(\mathbf{x})^{\mathrm{T}}\mathbf{v}_i = \sum_{n=1}^{N}a_{in}k(\mathbf{x},\mathbf{x}_n) \tag{12.87}$$

is constant.

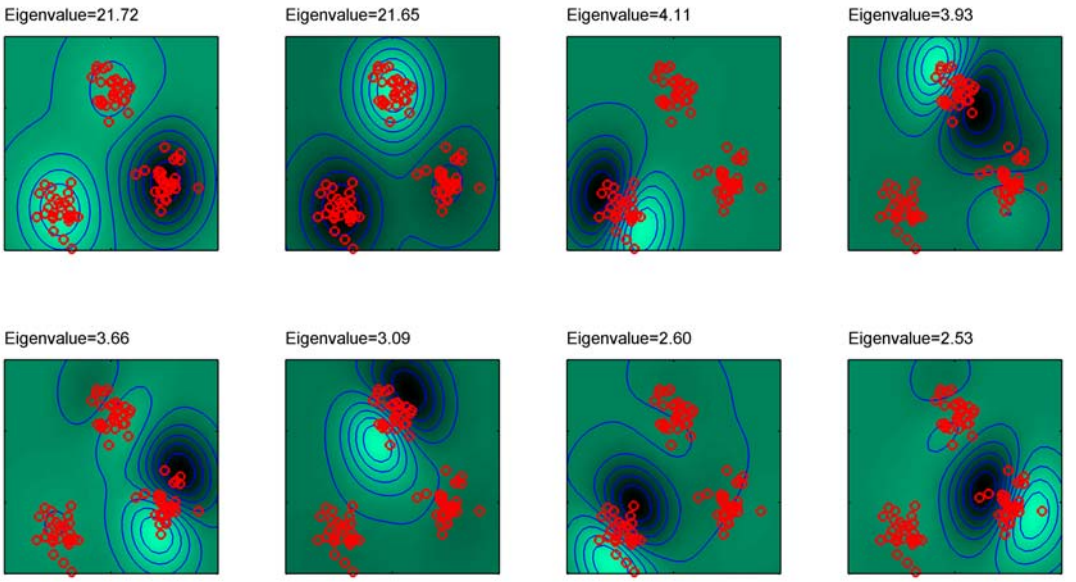


Figure 12.17 Example of kernel PCA, with a Gaussian kernel applied to a synthetic data set in two dimensions, showing the first eight eigenfunctions along with their eigenvalues. The contours are lines along which the projection onto the corresponding principal component is constant. Note how the first two eigenvectors separate the three clusters, the next three eigenvectors split each of the cluster into halves, and the following three eigenvectors again split the clusters into halves along directions orthogonal to the previous splits.

One obvious disadvantage of kernel PCA is that it involves finding the eigenvectors of the $N \times N$ matrix $\widetilde{\mathbf{K}}$ rather than the $D \times D$ matrix $\mathbf{S}$ of conventional linear PCA, and so in practice for large data sets approximations are often used.

Finally, we note that in standard linear PCA, we often retain some reduced number $L < D$ of eigenvectors and then approximate a data vector $\mathbf{x}_n$ by its projection $\widehat{\mathbf{x}}_n$ onto the L-dimensional principal subspace, defined by

$$\widehat{\mathbf{x}}_n = \sum_{i=1}^{L} \left(\mathbf{x}_n^{\mathrm{T}} \mathbf{u}_i\right) \mathbf{u}_i. \tag{12.88}$$

x_n PCA,投影点 $\widehat{x}_n$，则 $\widehat{x}$ 是 x_n 的近似

但对KPCA，可以这样理解：$x_n \xrightarrow{\phi} \phi(x_n)$，记为 y_n，PCA投影后 $\widehat{y}_n \xrightarrow{\phi^{-1}} \widehat{x}_n$。虽然 $\widehat{y}_n$ 是 y_n 的近似，但 $\widehat{x}_n$ 却不是 x_n 的近似。

In kernel PCA, this will in general not be possible. To see this, note that the mapping $\phi(\mathbf{x})$ maps the D-dimensional $\mathbf{x}$ space into a D-dimensional *manifold* in the M-dimensional feature space ϕ. The vector $\mathbf{x}$ is known as the *pre-image* of the corresponding point $\phi(\mathbf{x})$. However, the projection of points in feature space onto the linear PCA subspace in that space will typically not lie on the nonlinear D-dimensional manifold and so will not have a corresponding pre-image in data space. Techniques have therefore been proposed for finding approximate pre-images (Bakir *et al.*, 2004).

指 feature space

12.4. Nonlinear Latent Variable Models

In this chapter, we have focussed on the simplest class of models having continuous latent variables, namely those based on linear-Gaussian distributions. As well as having great practical importance, these models are relatively easy to analyse and to fit to data and can also be used as components in more complex models. Here we consider briefly some generalizations of this framework to models that are either nonlinear or non-Gaussian, or both.

In fact, the issues of nonlinearity and non-Gaussianity are related because a general probability density can be obtained from a simple fixed reference density, such as a Gaussian, by making a nonlinear change of variables. This idea forms the basis of several practical latent variable models as we shall see shortly.

Exercise 12.28

12.4.1 Independent component analysis

We begin by considering models in which the observed variables are related linearly to the latent variables, but for which the latent distribution is non-Gaussian. An important class of such models, known as *independent component analysis*, or *ICA*, arises when we consider a distribution over the latent variables that factorizes, so that

$$p(\mathbf{z}) = \prod_{j=1}^{M} p(z_j). \tag{12.89}$$

To understand the role of such models, consider a situation in which two people are talking at the same time, and we record their voices using two microphones. If we ignore effects such as time delay and echoes, then the signals received by the microphones at any point in time will be given by linear combinations of the amplitudes of the two voices. The coefficients of this linear combination will be constant, and if we can infer their values from sample data, then we can invert the mixing process (assuming it is nonsingular) and thereby obtain two clean signals each of which contains the voice of just one person. This is an example of a problem called *blind source separation* in which 'blind' refers to the fact that we are given only the mixed data, and neither the original sources nor the mixing coefficients are observed (Cardoso, 1998).

This type of problem is sometimes addressed using the following approach (MacKay, 2003) in which we ignore the temporal nature of the signals and treat the successive samples as i.i.d. We consider a generative model in which there are two latent variables corresponding to the unobserved speech signal amplitudes, and there are two observed variables given by the signal values at the microphones. The latent variables have a joint distribution that factorizes as above, and the observed variables are given by a linear combination of the latent variables. There is no need to include a noise distribution because the number of latent variables equals the number of observed variables, and therefore the marginal distribution of the observed variables will not in general be singular, so the observed variables are simply deterministic linear combinations of the latent variables. Given a data set of observations, the

likelihood function for this model is a function of the coefficients in the linear combination. The log likelihood can be maximized using gradient-based optimization giving rise to a particular version of independent component analysis.

The success of this approach requires that the latent variables have non-Gaussian distributions. To see this, recall that in probabilistic PCA (and in factor analysis) the latent-space distribution is given by a zero-mean isotropic Gaussian. The model therefore cannot distinguish between two different choices for the latent variables where these differ simply by a rotation in latent space. This can be verified directly by noting that the marginal density (12.35), and hence the likelihood function, is unchanged if we make the transformation $\mathbf{W} \rightarrow \mathbf{WR}$ where $\mathbf{R}$ is an orthogonal matrix satisfying $\mathbf{RR}^{\mathrm{T}} = \mathbf{I}$, because the matrix $\mathbf{C}$ given by (12.36) is itself invariant. Extending the model to allow more general Gaussian latent distributions does not change this conclusion because, as we have seen, such a model is equivalent to the zero-mean isotropic Gaussian latent variable model.

Another way to see why a Gaussian latent variable distribution in a linear model is insufficient to find independent components is to note that the principal components represent a rotation of the coordinate system in data space such as to diagonalize the covariance matrix, so that the data distribution in the new coordinates is then uncorrelated. Although zero correlation is a necessary condition for independence

Exercise 12.29

it is not, however, sufficient. In practice, a common choice for the latent-variable distribution is given by

$$p(z_j) = \frac{1}{\pi \cosh(z_j)} = \frac{1}{\pi(e^{z_j} + e^{-z_j})} \tag{12.90}$$

which has heavy tails compared to a Gaussian, reflecting the observation that many real-world distributions also exhibit this property.

The original ICA model (Bell and Sejnowski, 1995) was based on the optimization of an objective function defined by information maximization. One advantage of a probabilistic latent variable formulation is that it helps to motivate and formulate generalizations of basic ICA. For instance, *independent factor analysis* (Attias, 1999a) considers a model in which the number of latent and observed variables can differ, the observed variables are noisy, and the individual latent variables have flexible distributions modelled by mixtures of Gaussians. The log likelihood for this model is maximized using EM, and the reconstruction of the latent variables is approximated using a variational approach. Many other types of model have been considered, and there is now a huge literature on ICA and its applications (Jutten and Herault, 1991; Comon *et al.*, 1991; Amari *et al.*, 1996; Pearlmutter and Parra, 1997; Hyvärinen and Oja, 1997; Hinton *et al.*, 2001; Miskin and MacKay, 2001; Hojen-Sorensen *et al.*, 2002; Choudrey and Roberts, 2003; Chan *et al.*, 2003; Stone, 2004).

12.4.2 Autoassociative neural networks

In Chapter 5 we considered neural networks in the context of supervised learning, where the role of the network is to predict the output variables given values

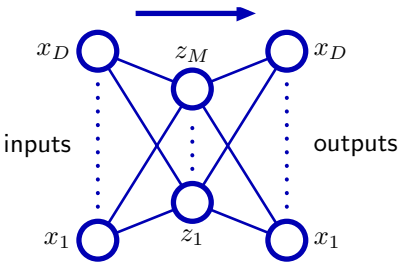


Figure 12.18 An autoassociative multilayer perceptron having two layers of weights. Such a network is trained to map input vectors onto themselves by minimization of a sum-of-squares error. Even with nonlinear units in the hidden layer, such a network is equivalent to linear principal component analysis. Links representing bias parameters have been omitted for clarity.

for the input variables. However, neural networks have also been applied to unsupervised learning where they have been used for dimensionality reduction. This is achieved by using a network having the same number of outputs as inputs, and optimizing the weights so as to minimize some measure of the reconstruction error between inputs and outputs with respect to a set of training data.

Consider first a multilayer perceptron of the form shown in Figure 12.18, having D inputs, D output units and M hidden units, with $M < D$. The targets used to train the network are simply the input vectors themselves, so that the network is attempting to map each input vector onto itself. Such a network is said to form an *autoassociative* mapping. Since the number of hidden units is smaller than the number of inputs, a perfect reconstruction of all input vectors is not in general possible. We therefore determine the network parameters $\mathbf{w}$ by minimizing an error function which captures the degree of mismatch between the input vectors and their reconstructions. In particular, we shall choose a sum-of-squares error of the form

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{x}_n\|^2. \tag{12.91}$$

If the hidden units have linear activation functions, then it can be shown that the error function has a unique global minimum, and that at this minimum the network performs a projection onto the M-dimensional subspace which is spanned by the first M principal components of the data (Bourlard and Kamp, 1988; Baldi and Hornik, 1989). Thus, the vectors of weights which lead into the hidden units in Figure 12.18 form a basis set which spans the principal subspace. Note, however, that these vectors need not be orthogonal or normalized. This result is unsurprising, since both principal component analysis and the neural network are using linear dimensionality reduction and are minimizing the same sum-of-squares error function.

It might be thought that the limitations of a linear dimensionality reduction could be overcome by using nonlinear (sigmoidal) activation functions for the hidden units in the network in Figure 12.18. However, even with nonlinear hidden units, the minimum error solution is again given by the projection onto the principal component subspace (Bourlard and Kamp, 1988). There is therefore no advantage in using two-layer neural networks to perform dimensionality reduction. Standard techniques for principal component analysis (based on singular value decomposition) are guaranteed to give the correct solution in finite time, and they also generate an ordered set of eigenvalues with corresponding orthonormal eigenvectors.

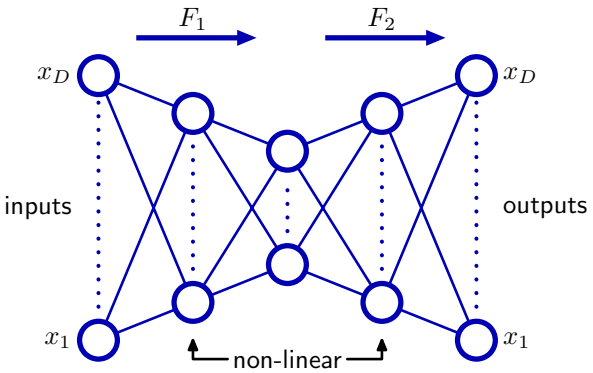


Figure 12.19 Addition of extra hidden layers of nonlinear units gives an autoassociative network which can perform a nonlinear dimensionality reduction.

The situation is different, however, if additional hidden layers are permitted in the network. Consider the four-layer autoassociative network shown in Figure 12.19. Again the output units are linear, and the M units in the second hidden layer can also be linear, however, the first and third hidden layers have sigmoidal nonlinear activation functions. The network is again trained by minimization of the error function (12.91). We can view this network as two successive functional mappings $\mathbf{F}_1$ and $\mathbf{F}_2$, as indicated in Figure 12.19. The first mapping $\mathbf{F}_1$ projects the original D-dimensional data onto an M-dimensional subspace $\mathcal{S}$ defined by the activations of the units in the second hidden layer. Because of the presence of the first hidden layer of nonlinear units, this mapping is very general, and in particular is not restricted to being linear. Similarly, the second half of the network defines an arbitrary functional mapping from the M-dimensional space back into the original D-dimensional input space. This has a simple geometrical interpretation, as indicated for the case $D = 3$ and $M = 2$ in Figure 12.20.

Such a network effectively performs a nonlinear principal component analysis.

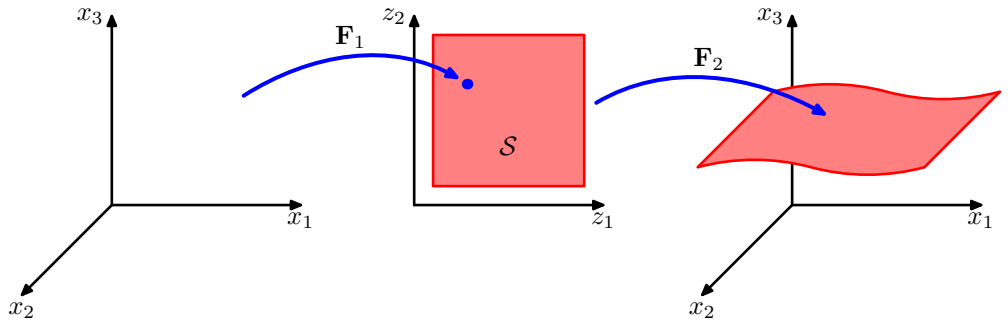


Figure 12.20 Geometrical interpretation of the mappings performed by the network in Figure 12.19 for the case of $D = 3$ inputs and $M = 2$ units in the middle hidden layer. The function $\mathbf{F}_2$ maps from an M-dimensional space $\mathcal{S}$ into a D-dimensional space and therefore defines the way in which the space $\mathcal{S}$ is embedded within the original $\mathbf{x}$-space. Since the mapping $\mathbf{F}_2$ can be nonlinear, the embedding of $\mathcal{S}$ can be nonplanar, as indicated in the figure. The mapping $\mathbf{F}_1$ then defines a projection of points in the original D-dimensional space into the M-dimensional subspace $\mathcal{S}$.

It has the advantage of not being limited to linear transformations, although it contains standard principal component analysis as a special case. However, training the network now involves a nonlinear optimization problem, since the error function (12.91) is no longer a quadratic function of the network parameters. Computationally intensive nonlinear optimization techniques must be used, and there is the risk of finding a suboptimal local minimum of the error function. Also, the dimensionality of the subspace must be specified before training the network.

12.4.3 Modelling nonlinear manifolds

As we have already noted, many natural sources of data correspond to low-dimensional, possibly noisy, nonlinear manifolds embedded within the higher dimensional observed data space. Capturing this property explicitly can lead to improved density modelling compared with more general methods. Here we consider briefly a range of techniques that attempt to do this.

One way to model the nonlinear structure is through a combination of linear models, so that we make a piece-wise linear approximation to the manifold. This can be obtained, for instance, by using a clustering technique such as K-means based on Euclidean distance to partition the data set into local groups with standard PCA applied to each group. A better approach is to use the reconstruction error for cluster assignment (Kambhatla and Leen, 1997; Hinton *et al.*, 1997) as then a common cost function is being optimized in each stage. However, these approaches still suffer from limitations due to the absence of an overall density model. By using probabilistic PCA it is straightforward to define a fully probabilistic model simply by considering a mixture distribution in which the components are probabilistic PCA models (Tipping and Bishop, 1999a). Such a model has both discrete latent variables, corresponding to the discrete mixture, as well as continuous latent variables, and the likelihood function can be maximized using the EM algorithm. A fully Bayesian treatment, based on variational inference (Bishop and Winn, 2000), allows the number of components in the mixture, as well as the effective dimensionalities of the individual models, to be inferred from the data. There are many variants of this model in which parameters such as the $\mathbf{W}$ matrix or the noise variances are tied across components in the mixture, or in which the isotropic noise distributions are replaced by diagonal ones, giving rise to a mixture of factor analysers (Ghahramani and Hinton, 1996a; Ghahramani and Beal, 2000). The mixture of probabilistic PCA models can also be extended hierarchically to produce an interactive data visualization algorithm (Bishop and Tipping, 1998).

An alternative to considering a mixture of linear models is to consider a single nonlinear model. Recall that conventional PCA finds a linear subspace that passes close to the data in a least-squares sense. This concept can be extended to one-dimensional nonlinear surfaces in the form of *principal curves* (Hastie and Stuetzle, 1989). We can describe a curve in a D-dimensional data space using a vector-valued function $\mathbf{f}(\lambda)$, which is a vector each of whose elements is a function of the scalar λ. There are many possible ways to parameterize the curve, of which a natural choice is the arc length along the curve. For any given point $\widehat{\mathbf{x}}$ in data space, we can find the point on the curve that is closest in Euclidean distance. We denote this point by

$\lambda = g_{\mathbf{f}}(\mathbf{x})$ because it depends on the particular curve $\mathbf{f}(\lambda)$. For a continuous data density $p(\mathbf{x})$, a principal curve is defined as one for which every point on the curve is the mean of all those points in data space that project to it, so that

$$\mathbb{E}\left[\mathbf{x}|g_{\mathbf{f}}(\mathbf{x}) = \lambda\right] = \mathbf{f}(\lambda). \tag{12.92}$$

For a given continuous density, there can be many principal curves. In practice, we are interested in finite data sets, and we also wish to restrict attention to smooth curves. Hastie and Stuetzle (1989) propose a two-stage iterative procedure for finding such principal curves, somewhat reminiscent of the EM algorithm for PCA. The curve is initialized using the first principal component, and then the algorithm alternates between a data projection step and curve re-estimation step. In the projection step, each data point is assigned to a value of λ corresponding to the closest point on the curve. Then in the re-estimation step, each point on the curve is given by a weighted average of those points that project to nearby points on the curve, with points closest on the curve given the greatest weight. In the case where the subspace is constrained to be linear, the procedure converges to the first principal component and is equivalent to the power method for finding the largest eigenvector of the covariance matrix. Principal curves can be generalized to multidimensional manifolds called *principal surfaces* although these have found limited use due to the difficulty of data smoothing in higher dimensions even for two-dimensional manifolds.

MDS

PCA is often used to project a data set onto a lower-dimensional space, for example two dimensional, for the purposes of visualization. Another linear technique with a similar aim is *multidimensional scaling*, or *MDS* (Cox and Cox, 2000). It finds a low-dimensional projection of the data such as to preserve, as closely as possible, the pairwise distances between data points, and involves finding the eigenvectors of the distance matrix. In the case where the distances are Euclidean, it gives equivalent results to PCA. The MDS concept can be extended to a wide variety of data types specified in terms of a similarity matrix, giving *nonmetric* MDS.

LLE

Two other nonprobabilistic methods for dimensionality reduction and data visualization are worthy of mention. *Locally linear embedding*, or *LLE* (Roweis and Saul, 2000) first computes the set of coefficients that best reconstructs each data point from its neighbours. These coefficients are arranged to be invariant to rotations, translations, and scalings of that data point and its neighbours, and hence they characterize the local geometrical properties of the neighbourhood. LLE then maps the high-dimensional data points down to a lower dimensional space while preserving these neighbourhood coefficients. If the local neighbourhood for a particular data point can be considered linear, then the transformation can be achieved using a combination of translation, rotation, and scaling, such as to preserve the angles formed between the data points and their neighbours. Because the weights are invariant to these transformations, we expect the same weight values to reconstruct the data points in the low-dimensional space as in the high-dimensional data space. In spite of the nonlinearity, the optimization for LLE does not exhibit local minima.

isomap

In *isometric feature mapping*, or *isomap* (Tenenbaum *et al.*, 2000), the goal is to project the data to a lower-dimensional space using MDS, but where the dissimilarities are defined in terms of the *geodesic distances* measured along the mani-

fold. For instance, if two points lie on a circle, then the geodesic is the arc-length distance measured around the circumference of the circle not the straight line distance measured along the chord connecting them. The algorithm first defines the neighbourhood for each data point, either by finding the K nearest neighbours or by finding all points within a sphere of radius ϵ. A graph is then constructed by linking all neighbouring points and labelling them with their Euclidean distance. The geodesic distance between any pair of points is then approximated by the sum of the arc lengths along the shortest path connecting them (which itself is found using standard algorithms). Finally, metric MDS is applied to the geodesic distance matrix to find the low-dimensional projection.

latent trait models

Our focus in this chapter has been on models for which the observed variables are continuous. We can also consider models having continuous latent variables together with discrete observed variables, giving rise to *latent trait* models (Bartholomew, 1987). In this case, the marginalization over the continuous latent variables, even for a linear relationship between latent and observed variables, cannot be performed analytically, and so more sophisticated techniques are required. Tipping (1999) uses variational inference in a model with a two-dimensional latent space, allowing a binary data set to be visualized analogously to the use of PCA to visualize continuous data. Note that this model is the dual of the Bayesian logistic regression problem discussed in Section 4.5. In the case of logistic regression we have N observations of the feature vector $\boldsymbol{\phi}_n$ which are parameterized by a single parameter vector $\mathbf{w}$, whereas in the latent space visualization model there is a single latent space variable $\mathbf{x}$ (analogous to $\boldsymbol{\phi}$) and N copies of the latent variable $\mathbf{w}_n$. A generalization of probabilistic latent variable models to general exponential family distributions is described in Collins *et al.* (2002).

density network

We have already noted that an arbitrary distribution can be formed by taking a Gaussian random variable and transforming it through a suitable nonlinearity. This is exploited in a general latent variable model called a *density network* (MacKay, 1995; MacKay and Gibbs, 1999) in which the nonlinear function is governed by a multilayered neural network. If the network has enough hidden units, it can approximate a given nonlinear function to any desired accuracy. The downside of having such a flexible model is that the marginalization over the latent variables, required in order to obtain the likelihood function, is no longer analytically tractable. Instead, the likelihood is approximated using Monte Carlo techniques by drawing samples from the Gaussian prior. The marginalization over the latent variables then becomes a simple sum with one term for each sample. However, because a large number of sample points may be required in order to give an accurate representation of the marginal, this procedure can be computationally costly.

Chapter 5

Chapter 11

GTM & SOM

If we consider more restricted forms for the nonlinear function, and make an appropriate choice of the latent variable distribution, then we can construct a latent variable model that is both nonlinear and efficient to train. The *generative topographic mapping*, or *GTM* (Bishop *et al.*, 1996; Bishop *et al.*, 1997a; Bishop *et al.*, 1998b) uses a latent distribution that is defined by a finite regular grid of delta functions over the (typically two-dimensional) latent space. Marginalization over the latent space then simply involves summing over the contributions from each of the grid locations.

GTM

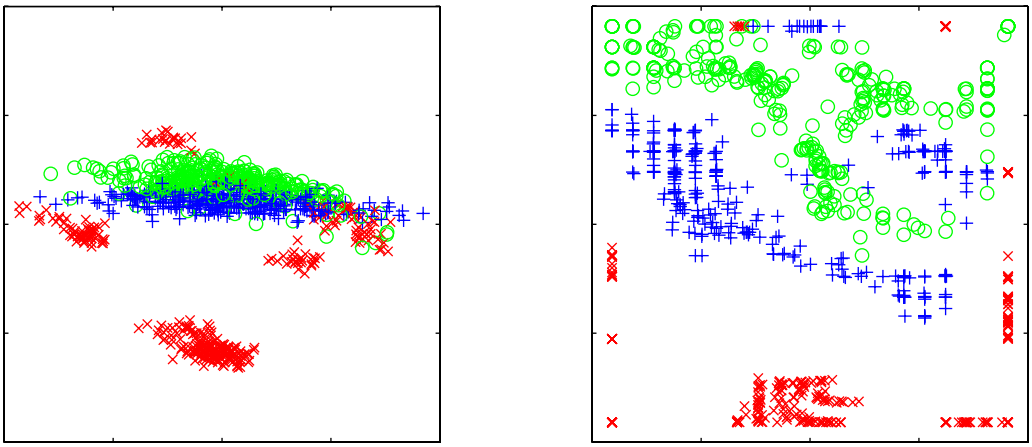


Figure 12.21 Plot of the oil flow data set visualized using PCA on the left and GTM on the right. For the GTM model, each data point is plotted at the mean of its posterior distribution in latent space. The nonlinearity of the GTM model allows the separation between the groups of data points to be seen more clearly.

Chapter 3

Section 1.4

The nonlinear mapping is given by a linear regression model that allows for general nonlinearity while being a linear function of the adaptive parameters. Note that the usual limitation of linear regression models arising from the curse of dimensionality does not arise in the context of the GTM since the manifold generally has two dimensions irrespective of the dimensionality of the data space. A consequence of these two choices is that the likelihood function can be expressed analytically in closed form and can be optimized efficiently using the EM algorithm. The resulting GTM model fits a two-dimensional nonlinear manifold to the data set, and by evaluating the posterior distribution over latent space for the data points, they can be projected back to the latent space for visualization purposes. Figure 12.21 shows a comparison of the oil data set visualized with linear PCA and with the nonlinear GTM.

The GTM can be seen as a probabilistic version of an earlier model called the *self organizing map*, or *SOM* (Kohonen, 1982; Kohonen, 1995), which also represents a two-dimensional nonlinear manifold as a regular array of discrete points. The SOM is somewhat reminiscent of the K-means algorithm in that data points are assigned to nearby prototype vectors that are then subsequently updated. Initially, the prototypes are distributed at random, and during the training process they 'self organize' so as to approximate a smooth manifold. Unlike K-means, however, the SOM is not optimizing any well-defined cost function (Erwin *et al.*, 1992) making it difficult to set the parameters of the model and to assess convergence. There is also no guarantee that the 'self-organization' will take place as this is dependent on the choice of appropriate parameter values for any particular data set.

By contrast, GTM optimizes the log likelihood function, and the resulting model defines a probability density in data space. In fact, it corresponds to a constrained mixture of Gaussians in which the components share a common variance, and the means are constrained to lie on a smooth two-dimensional manifold. This proba-

bilistic foundation also makes it very straightforward to define generalizations of GTM (Bishop *et al.*, 1998a) such as a Bayesian treatment, dealing with missing values, a principled extension to discrete variables, the use of Gaussian processes to define the manifold, or a hierarchical GTM model (Tino and Nabney, 2002).

Section 6.4

Because the manifold in GTM is defined as a continuous surface, not just at the prototype vectors as in the SOM, it is possible to compute the *magnification factors* corresponding to the local expansions and compressions of the manifold needed to fit the data set (Bishop *et al.*, 1997b) as well as the *directional curvatures* of the manifold (Tino *et al.*, 2001). These can be visualized along with the projected data and provide additional insight into the model.

Exercises

12.1 (⋆⋆) **www** In this exercise, we use proof by induction to show that the linear projection onto an M-dimensional subspace that maximizes the variance of the projected data is defined by the M eigenvectors of the data covariance matrix $\mathbf{S}$, given by (12.3), corresponding to the M largest eigenvalues. In Section 12.1, this result was proven for the case of $M = 1$. Now suppose the result holds for some general value of M and show that it consequently holds for dimensionality $M + 1$. To do this, first set the derivative of the variance of the projected data with respect to a vector $\mathbf{u}_{M+1}$ defining the new direction in data space equal to zero. This should be done subject to the constraints that $\mathbf{u}_{M+1}$ be orthogonal to the existing vectors $\mathbf{u}_1, \ldots, \mathbf{u}_M$, and also that it be normalized to unit length. Use Lagrange multipliers to enforce these constraints. Then make use of the orthonormality properties of the vectors $\mathbf{u}_1, \ldots, \mathbf{u}_M$ to show that the new vector $\mathbf{u}_{M+1}$ is an eigenvector of $\mathbf{S}$. Finally, show that the variance is maximized if the eigenvector is chosen to be the one corresponding to eigenvector λ_{M+1} where the eigenvalues have been ordered in decreasing value.

Appendix E

12.2 (⋆⋆) Show that the minimum value of the PCA distortion measure J given by (12.15) with respect to the $\mathbf{u}_i$, subject to the orthonormality constraints (12.7), is obtained when the $\mathbf{u}_i$ are eigenvectors of the data covariance matrix $\mathbf{S}$. To do this, introduce a matrix $\mathbf{H}$ of Lagrange multipliers, one for each constraint, so that the modified distortion measure, in matrix notation reads

$$\widetilde{J} = \mathrm{Tr}\left\{\widehat{\mathbf{U}}^{\mathrm{T}}\mathbf{S}\widehat{\mathbf{U}}\right\} + \mathrm{Tr}\left\{\mathbf{H}(\mathbf{I} - \widehat{\mathbf{U}}^{\mathrm{T}}\widehat{\mathbf{U}})\right\} \tag{12.93}$$

where $\widehat{\mathbf{U}}$ is a matrix of dimension $D \times (D - M)$ whose columns are given by $\mathbf{u}_i$. Now minimize $\widetilde{J}$ with respect to $\widehat{\mathbf{U}}$ and show that the solution satisfies $\mathbf{S}\widehat{\mathbf{U}} = \widehat{\mathbf{U}}\mathbf{H}$. Clearly, one possible solution is that the columns of $\widehat{\mathbf{U}}$ are eigenvectors of $\mathbf{S}$, in which case $\mathbf{H}$ is a diagonal matrix containing the corresponding eigenvalues. To obtain the general solution, show that $\mathbf{H}$ can be assumed to be a symmetric matrix, and by using its eigenvector expansion show that the general solution to $\mathbf{S}\widehat{\mathbf{U}} = \widehat{\mathbf{U}}\mathbf{H}$ gives the same value for $\widetilde{J}$ as the specific solution in which the columns of $\widehat{\mathbf{U}}$ are

the eigenvectors of $\mathbf{S}$. Because these solutions are all equivalent, it is convenient to choose the eigenvector solution.

12.3 ($\star$) Verify that the eigenvectors defined by (12.30) are normalized to unit length, assuming that the eigenvectors $\mathbf{v}_i$ have unit length.

12.4 ($\star$) **www** Suppose we replace the zero-mean, unit-covariance latent space distribution (12.31) in the probabilistic PCA model by a general Gaussian distribution of the form $\mathcal{N}(\mathbf{z}|\mathbf{m}, \mathbf{\Sigma})$. By redefining the parameters of the model, show that this leads to an identical model for the marginal distribution $p(\mathbf{x})$ over the observed variables for any valid choice of $\mathbf{m}$ and $\mathbf{\Sigma}$.

12.5 ($\star\star$) Let $\mathbf{x}$ be a D-dimensional random variable having a Gaussian distribution given by $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{\Sigma})$, and consider the M-dimensional random variable given by $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$ where $\mathbf{A}$ is an $M \times D$ matrix. Show that $\mathbf{y}$ also has a Gaussian distribution, and find expressions for its mean and covariance. Discuss the form of this Gaussian distribution for $M < D$, for $M = D$, and for $M > D$.

12.6 ($\star$) **www** Draw a directed probabilistic graph for the probabilistic PCA model described in Section 12.2 in which the components of the observed variable $\mathbf{x}$ are shown explicitly as separate nodes. Hence verify that the probabilistic PCA model has the same independence structure as the naive Bayes model discussed in Section 8.2.2.

12.7 ($\star\star$) By making use of the results (2.270) and (2.271) for the mean and covariance of a general distribution, derive the result (12.35) for the marginal distribution $p(\mathbf{x})$ in the probabilistic PCA model.

12.8 ($\star\star$) **www** By making use of the result (2.116), show that the posterior distribution $p(\mathbf{z}|\mathbf{x})$ for the probabilistic PCA model is given by (12.42).

12.9 ($\star$) Verify that maximizing the log likelihood (12.43) for the probabilistic PCA model with respect to the parameter $\boldsymbol{\mu}$ gives the result $\boldsymbol{\mu}_{\mathrm{ML}} = \overline{\mathbf{x}}$ where $\overline{\mathbf{x}}$ is the mean of the data vectors.

12.10 ($\star\star$) By evaluating the second derivatives of the log likelihood function (12.43) for the probabilistic PCA model with respect to the parameter $\boldsymbol{\mu}$, show that the stationary point $\boldsymbol{\mu}_{\mathrm{ML}} = \overline{\mathbf{x}}$ represents the unique maximum.

12.11 ($\star\star$) **www** Show that in the limit $\sigma^2 \to 0$, the posterior mean for the probabilistic PCA model becomes an orthogonal projection onto the principal subspace, as in conventional PCA.

12.12 ($\star\star$) For $\sigma^2 > 0$ show that the posterior mean in the probabilistic PCA model is shifted towards the origin relative to the orthogonal projection.

12.13 ($\star\star$) Show that the optimal reconstruction of a data point under probabilistic PCA, according to the least squares projection cost of conventional PCA, is given by

$$\widetilde{\mathbf{x}} = \mathbf{W}_{\mathrm{ML}}(\mathbf{W}_{\mathrm{ML}}^{\mathrm{T}}\mathbf{W}_{\mathrm{ML}})^{-1}\mathbf{M}\mathbb{E}[\mathbf{z}|\mathbf{x}]. \tag{12.94}$$

12.14 ($\star$) The number of independent parameters in the covariance matrix for the probabilistic PCA model with an M-dimensional latent space and a D-dimensional data space is given by (12.51). Verify that in the case of $M = D - 1$, the number of independent parameters is the same as in a general covariance Gaussian, whereas for $M = 0$ it is the same as for a Gaussian with an isotropic covariance.

12.15 ($\star\star$) **www** Derive the M-step equations (12.56) and (12.57) for the probabilistic PCA model by maximization of the expected complete-data log likelihood function given by (12.53).

12.16 ($\star\star\star$) In Figure 12.11, we showed an application of probabilistic PCA to a data set in which some of the data values were missing at random. Derive the EM algorithm for maximizing the likelihood function for the probabilistic PCA model in this situation. Note that the $\{\mathbf{z}_n\}$, as well as the missing data values that are components of the vectors $\{\mathbf{x}_n\}$, are now latent variables. Show that in the special case in which all of the data values are observed, this reduces to the EM algorithm for probabilistic PCA derived in Section 12.2.2.

12.17 ($\star\star$) **www** Let $\mathbf{W}$ be a $D \times M$ matrix whose columns define a linear subspace of dimensionality M embedded within a data space of dimensionality D, and let $\boldsymbol{\mu}$ be a D-dimensional vector. Given a data set $\{\mathbf{x}_n\}$ where $n = 1, \ldots, N$, we can approximate the data points using a linear mapping from a set of M-dimensional vectors $\{\mathbf{z}_n\}$, so that $\mathbf{x}_n$ is approximated by $\mathbf{W}\mathbf{z}_n + \boldsymbol{\mu}$. The associated sum-of-squares reconstruction cost is given by

$$J = \sum_{n=1}^{N} \|\mathbf{x}_n - \boldsymbol{\mu} - \mathbf{W}\mathbf{z}_n\|^2. \tag{12.95}$$

First show that minimizing J with respect to $\boldsymbol{\mu}$ leads to an analogous expression with $\mathbf{x}_n$ and $\mathbf{z}_n$ replaced by zero-mean variables $\mathbf{x}_n - \overline{\mathbf{x}}$ and $\mathbf{z}_n - \overline{\mathbf{z}}$, respectively, where $\overline{\mathbf{x}}$ and $\overline{\mathbf{z}}$ denote sample means. Then show that minimizing J with respect to $\mathbf{z}_n$, where $\mathbf{W}$ is kept fixed, gives rise to the PCA E step (12.58), and that minimizing J with respect to $\mathbf{W}$, where $\{\mathbf{z}_n\}$ is kept fixed, gives rise to the PCA M step (12.59).

12.18 ($\star$) Derive an expression for the number of independent parameters in the factor analysis model described in Section 12.2.4.

12.19 ($\star\star$) **www** Show that the factor analysis model described in Section 12.2.4 is invariant under rotations of the latent space coordinates.

12.20 ($\star\star$) By considering second derivatives, show that the only stationary point of the log likelihood function for the factor analysis model discussed in Section 12.2.4 with respect to the parameter $\boldsymbol{\mu}$ is given by the sample mean defined by (12.1). Furthermore, show that this stationary point is a maximum.

12.21 ($\star\star$) Derive the formulae (12.66) and (12.67) for the E step of the EM algorithm for factor analysis. Note that from the result of Exercise 12.20, the parameter $\boldsymbol{\mu}$ can be replaced by the sample mean $\overline{\mathbf{x}}$.

12.22 ($\star\star$) Write down an expression for the expected complete-data log likelihood function for the factor analysis model, and hence derive the corresponding M step equations (12.69) and (12.70).

12.23 ($\star$) **www** Draw a directed probabilistic graphical model representing a discrete mixture of probabilistic PCA models in which each PCA model has its own values of $\mathbf{W}$, $\boldsymbol{\mu}$, and σ^2. Now draw a modified graph in which these parameter values are shared between the components of the mixture.

12.24 ($\star\star\star$) We saw in Section 2.3.7 that Student's t-distribution can be viewed as an infinite mixture of Gaussians in which we marginalize with respect to a continuous latent variable. By exploiting this representation, formulate an EM algorithm for maximizing the log likelihood function for a multivariate Student's t-distribution given an observed set of data points, and derive the forms of the E and M step equations.

12.25 ($\star\star$) **www** Consider a linear-Gaussian latent-variable model having a latent space distribution $p(\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I})$ and a conditional distribution for the observed variable $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \boldsymbol{\Phi})$ where $\boldsymbol{\Phi}$ is an arbitrary symmetric, positive-definite noise covariance matrix. Now suppose that we make a nonsingular linear transformation of the data variables $\mathbf{x} \rightarrow \mathbf{A}\mathbf{x}$, where $\mathbf{A}$ is a $D \times D$ matrix. If $\boldsymbol{\mu}_{\mathrm{ML}}$, $\mathbf{W}_{\mathrm{ML}}$ and $\boldsymbol{\Phi}_{\mathrm{ML}}$ represent the maximum likelihood solution corresponding to the original untransformed data, show that $\mathbf{A}\boldsymbol{\mu}_{\mathrm{ML}}$, $\mathbf{A}\mathbf{W}_{\mathrm{ML}}$, and $\mathbf{A}\boldsymbol{\Phi}_{\mathrm{ML}}\mathbf{A}^{\mathrm{T}}$ will represent the corresponding maximum likelihood solution for the transformed data set. Finally, show that the form of the model is preserved in two cases: (i) $\mathbf{A}$ is a diagonal matrix and $\boldsymbol{\Phi}$ is a diagonal matrix. This corresponds to the case of factor analysis. The transformed $\boldsymbol{\Phi}$ remains diagonal, and hence factor analysis is *covariant* under component-wise re-scaling of the data variables; (ii) $\mathbf{A}$ is orthogonal and $\boldsymbol{\Phi}$ is proportional to the unit matrix so that $\boldsymbol{\Phi} = \sigma^2\mathbf{I}$. This corresponds to probabilistic PCA. The transformed $\boldsymbol{\Phi}$ matrix remains proportional to the unit matrix, and hence probabilistic PCA is covariant under a rotation of the axes of data space, as is the case for conventional PCA.

上面给出的结论，下面则给出的结论在PPCA和factor analysis中的应用

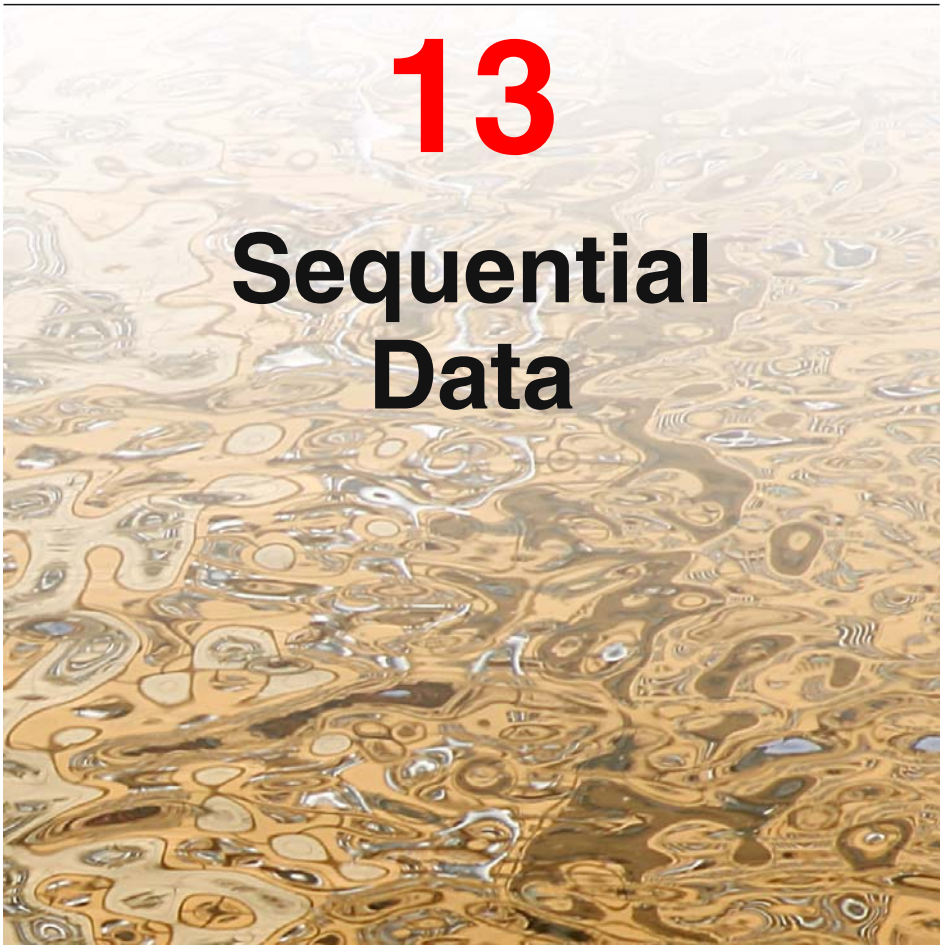
12.26 ($\star\star$) Show that any vector $\mathbf{a}_i$ that satisfies (12.80) will also satisfy (12.79). Also, show that for any solution of (12.80) having eigenvalue λ, we can add any multiple of an eigenvector of $\mathbf{K}$ having zero eigenvalue, and obtain a solution to (12.79) that also has eigenvalue λ. Finally, show that such modifications do not affect the principal-component projection given by (12.82).

12.27 ($\star\star$) Show that the conventional linear PCA algorithm is recovered as a special case of kernel PCA if we choose the linear kernel function given by $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{x}'$.

12.28 ($\star\star$) **www** Use the transformation property (1.27) of a probability density under a change of variable to show that any density $p(y)$ can be obtained from a fixed density $q(x)$ that is everywhere nonzero by making a nonlinear change of variable $y = f(x)$ in which $f(x)$ is a monotonic function so that $0 \leqslant f'(x) < \infty$. Write down the differential equation satisfied by $f(x)$ and draw a diagram illustrating the transformation of the density.

12.29 (⋆⋆) **www** Suppose that two variables z_1 and z_2 are independent so that $p(z_1, z_2) = p(z_1)p(z_2)$. Show that the covariance matrix between these variables is diagonal. This shows that independence is a sufficient condition for two variables to be uncorrelated. Now consider two variables y_1 and y_2 ~~in which $-1 \leqslant y_1 \leqslant 1$ and $y_2 = y_2^2$~~. Write down the conditional distribution $p(y_2|y_1)$ and observe that this is dependent on y_1, showing that the two variables are not independent. Now show that the covariance matrix between these two variables is again diagonal. To do this, use the relation $p(y_1, y_2) = p(y_1)p(y_2|y_1)$ to show that the off-diagonal terms are zero. This counter-example shows that zero correlation is not a sufficient condition for independence.

where y_1 is symmetrically distributed around 0 and $y_2 = y_1^2$.



13 Sequential Data

So far in this book, we have focussed primarily on sets of data points that were assumed to be independent and identically distributed (i.i.d.). This assumption allowed us to express the likelihood function as the product over all data points of the probability distribution evaluated at each data point. For many applications, however, the i.i.d. assumption will be a poor one. Here we consider a particularly important class of such data sets, namely those that describe sequential data. These often arise through measurement of time series, for example the rainfall measurements on successive days at a particular location, or the daily values of a currency exchange rate, or the acoustic features at successive time frames used for speech recognition. An example involving speech data is shown in Figure 13.1. Sequential data can also arise in contexts other than time series, for example the sequence of nucleotide base pairs along a strand of DNA or the sequence of characters in an English sentence. For convenience, we shall sometimes refer to 'past' and 'future' observations in a sequence. However, the models explored in this chapter are equally applicable to all

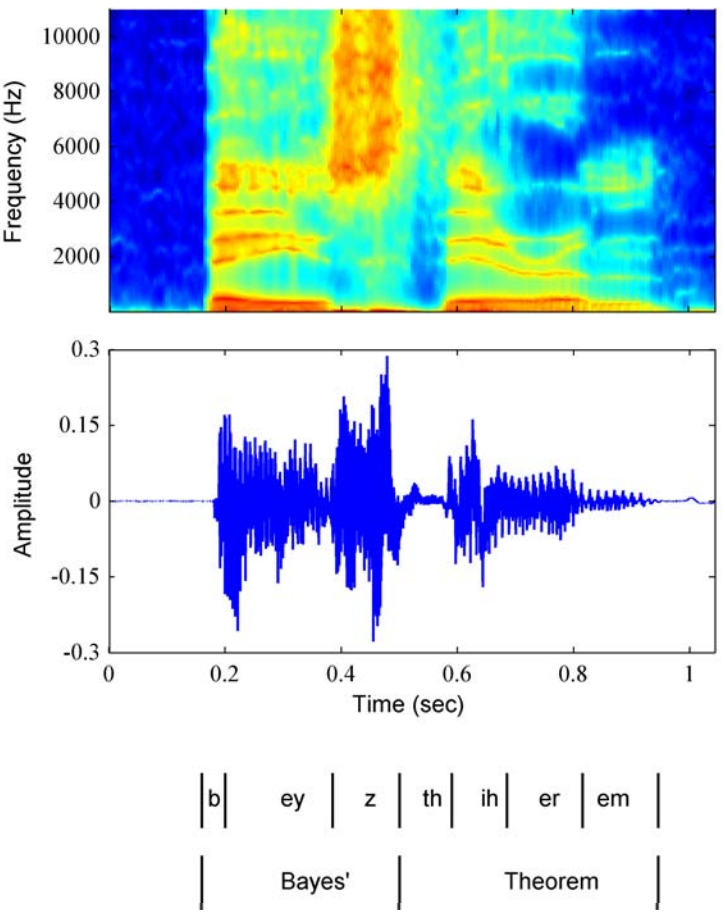


Figure 13.1 Example of a spectrogram of the spoken words "Bayes' theorem" showing a plot of the intensity of the spectral coefficients versus time index.

forms of sequential data, not just temporal sequences.

It is useful to distinguish between stationary and nonstationary sequential distributions. In the stationary case, the data evolves in time, but the distribution from which it is generated remains the same. For the more complex nonstationary situation, the generative distribution itself is evolving with time. Here we shall focus on the stationary case.

For many applications, such as financial forecasting, we wish to be able to predict the next value in a time series given observations of the previous values. Intuitively, we expect that recent observations are likely to be more informative than more historical observations in predicting future values. The example in Figure 13.1 shows that successive observations of the speech spectrum are indeed highly correlated. Furthermore, it would be impractical to consider a general dependence of future observations on all previous observations because the complexity of such a model would grow without limit as the number of observations increases. This leads us to consider *Markov models* in which we assume that future predictions are inde-

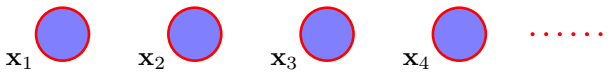


Figure 13.2 The simplest approach to modelling a sequence of observations is to treat them as independent, corresponding to a graph without links.

pendent of all but the most recent observations.

Although such models are tractable, they are also severely limited. We can obtain a more general framework, while still retaining tractability, by the introduction of latent variables, leading to *state space models*. As in Chapters 9 and 12, we shall see that complex models can thereby be constructed from simpler components (in particular, from distributions belonging to the exponential family) and can be readily characterized using the framework of probabilistic graphical models. Here we focus on the two most important examples of state space models, namely the *hidden Markov model*, in which the latent variables are discrete, and *linear dynamical systems*, in which the latent variables are Gaussian. Both models are described by directed graphs having a tree structure (no loops) for which inference can be performed efficiently using the sum-product algorithm.

13.1. Markov Models

The easiest way to treat sequential data would be simply to ignore the sequential aspects and treat the observations as i.i.d., corresponding to the graph in Figure 13.2. Such an approach, however, would fail to exploit the sequential patterns in the data, such as correlations between observations that are close in the sequence. Suppose, for instance, that we observe a binary variable denoting whether on a particular day it rained or not. Given a time series of recent observations of this variable, we wish to predict whether it will rain on the next day. If we treat the data as i.i.d., then the only information we can glean from the data is the relative frequency of rainy days. However, we know in practice that the weather often exhibits trends that may last for several days. Observing whether or not it rains today is therefore of significant help in predicting if it will rain tomorrow.

To express such effects in a probabilistic model, we need to relax the i.i.d. assumption, and one of the simplest ways to do this is to consider a *Markov model*. First of all we note that, without loss of generality, we can use the product rule to express the joint distribution for a sequence of observations in the form

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = p(\mathbf{x}_1) \prod_{n=2}^{N} p(\mathbf{x}_n | \mathbf{x}_1, \ldots, \mathbf{x}_{n-1}). \tag{13.1}$$

If we now assume that each of the conditional distributions on the right-hand side is independent of all previous observations except the most recent, we obtain the *first-order Markov chain*, which is depicted as a graphical model in Figure 13.3. The

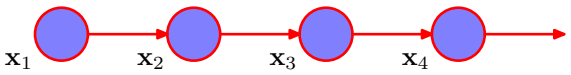


Figure 13.3 A first-order Markov chain of observations $\{\mathbf{x}_n\}$ in which the distribution $p(\mathbf{x}_n|\mathbf{x}_{n-1})$ of a particular observation $\mathbf{x}_n$ is conditioned on the value of the previous observation $\mathbf{x}_{n-1}$.

joint distribution for a sequence of N observations under this model is given by

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = p(\mathbf{x}_1) \prod_{n=2}^{N} p(\mathbf{x}_n|\mathbf{x}_{n-1}). \tag{13.2}$$

Section 8.2 From the d-separation property, we see that the conditional distribution for observation $\mathbf{x}_n$, given all of the observations up to time n, is given by

$$p(\mathbf{x}_n|\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}) = p(\mathbf{x}_n|\mathbf{x}_{n-1}) \tag{13.3}$$

Exercise 13.1 which is easily verified by direct evaluation starting from (13.2) and using the product rule of probability. Thus if we use such a model to predict the next observation in a sequence, the distribution of predictions will depend only on the value of the immediately preceding observation and will be independent of all earlier observations.

In most applications of such models, the conditional distributions $p(\mathbf{x}_n|\mathbf{x}_{n-1})$ that define the model will be constrained to be equal, corresponding to the assumption of a stationary time series. The model is then known as a *homogeneous* Markov chain. For instance, if the conditional distributions depend on adjustable parameters (whose values might be inferred from a set of training data), then all of the conditional distributions in the chain will share the same values of those parameters.

Although this is more general than the independence model, it is still very restrictive. For many sequential observations, we anticipate that the trends in the data over several successive observations will provide important information in predicting the next value. One way to allow earlier observations to have an influence is to move to higher-order Markov chains. If we allow the predictions to depend also on the previous-but-one value, we obtain a second-order Markov chain, represented by the graph in Figure 13.4. The joint distribution is now given by

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1) \prod_{n=3}^{N} p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{x}_{n-2}). \tag{13.4}$$

Again, using d-separation or by direct evaluation, we see that the conditional distribution of $\mathbf{x}_n$ given $\mathbf{x}_{n-1}$ and $\mathbf{x}_{n-2}$ is independent of all observations $\mathbf{x}_1, \ldots \mathbf{x}_{n-3}$.

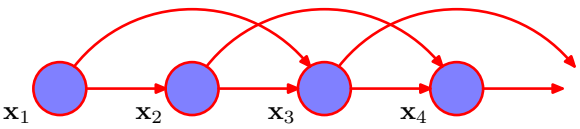


Figure 13.4 A second-order Markov chain, in which the conditional distribution of a particular observation $\mathbf{x}_n$ depends on the values of the two previous observations $\mathbf{x}_{n-1}$ and $\mathbf{x}_{n-2}$.

HMM和LDS的概率图均为Figure 13.5，其实和RNN很相似，处理的是时序数据，每个时间步用的参数均相同，不随时间变化。

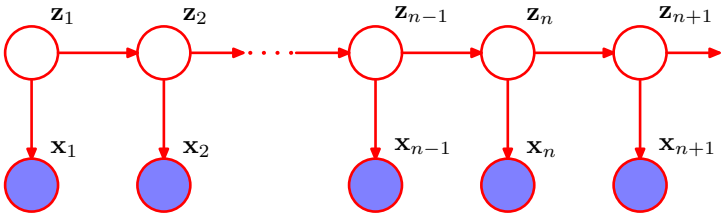


Figure 13.5 We can represent sequential data using a Markov chain of latent variables, with each observation conditioned on the state of the corresponding latent variable. This important graphical structure forms the foundation both for the hidden Markov model and for linear dynamical systems.

Each observation is now influenced by two previous observations. We can similarly consider extensions to an M^{th} order Markov chain in which the conditional distribution for a particular variable depends on the previous M variables. However, we have paid a price for this increased flexibility because the number of parameters in the model is now much larger. Suppose the observations are discrete variables having K states. Then the conditional distribution $p(\mathbf{x}_n|\mathbf{x}_{n-1})$ in a first-order Markov chain will be specified by a set of $K-1$ parameters for each of the K states of $\mathbf{x}_{n-1}$ giving a total of $K(K-1)$ parameters. Now suppose we extend the model to an M^{th} order Markov chain, so that the joint distribution is built up from conditionals $p(\mathbf{x}_n|\mathbf{x}_{n-M},\ldots,\mathbf{x}_{n-1})$. If the variables are discrete, and if the conditional distributions are represented by general conditional probability tables, then the number of parameters in such a model will have $K^M(K-1)$ parameters. Because this grows exponentially with M, it will often render this approach impractical for larger values of M.

For continuous variables, we can use linear-Gaussian conditional distributions in which each node has a Gaussian distribution whose mean is a linear function of its parents. This is known as an *autoregressive* or *AR* model (Box *et al.*, 1994; Thiesson *et al.*, 2004). An alternative approach is to use a parametric model for $p(\mathbf{x}_n|\mathbf{x}_{n-M},\ldots,\mathbf{x}_{n-1})$ such as a neural network. This technique is sometimes called a *tapped delay line* because it corresponds to storing (delaying) the previous M values of the observed variable in order to predict the next value. The number of parameters can then be much smaller than in a completely general model (for example it may grow linearly with M), although this is achieved at the expense of a restricted family of conditional distributions.

Suppose we wish to build a model for sequences that is not limited by the Markov assumption to any order and yet that can be specified using a limited number of free parameters. We can achieve this by introducing additional latent variables to permit a rich class of models to be constructed out of simple components, as we did with mixture distributions in Chapter 9 and with continuous latent variable models in Chapter 12. For each observation $\mathbf{x}_n$, we introduce a corresponding latent variable $\mathbf{z}_n$ (which may be of different type or dimensionality to the observed variable). We now assume that it is the latent variables that form a Markov chain, giving rise to the graphical structure known as a *state space model*, which is shown in Figure 13.5. It satisfies the key conditional independence property that $\mathbf{z}_{n-1}$ and $\mathbf{z}_{n+1}$ are independent given $\mathbf{z}_n$, so that

$$\mathbf{z}_{n+1} \perp\!\!\!\perp \mathbf{z}_{n-1} \,|\, \mathbf{z}_n. \tag{13.5}$$

The joint distribution for this model is given by

$$p(\mathbf{x}_1,\ldots,\mathbf{x}_N,\mathbf{z}_1,\ldots,\mathbf{z}_N) = p(\mathbf{z}_1)\left[\prod_{n=2}^{N} p(\mathbf{z}_n|\mathbf{z}_{n-1})\right]\prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{z}_n). \tag{13.6}$$

Using the d-separation criterion, we see that there is always a path connecting any two observed variables $\mathbf{x}_n$ and $\mathbf{x}_m$ via the latent variables, and that this path is never blocked. Thus the predictive distribution $p(\mathbf{x}_{n+1}|\mathbf{x}_1,\ldots,\mathbf{x}_n)$ for observation $\mathbf{x}_{n+1}$ given all previous observations does not exhibit any conditional independence properties, and so our predictions for $\mathbf{x}_{n+1}$ depends on all previous observations. The observed variables, however, do not satisfy the Markov property at any order. We shall discuss how to evaluate the predictive distribution in later sections of this chapter.

There are two important models for sequential data that are described by this graph. If the latent variables are discrete, then we obtain the *hidden Markov model*, or *HMM* (Elliott *et al.*, 1995). Note that the observed variables in an HMM may be discrete or continuous, and a variety of different conditional distributions can be used to model them. If both the latent and the observed variables are Gaussian (with a linear-Gaussian dependence of the conditional distributions on their parents), then we obtain the *linear dynamical system*.

Section 13.2

Section 13.3

13.2. Hidden Markov Models

The hidden Markov model can be viewed as a specific instance of the state space model of Figure 13.5 in which the latent variables are discrete. However, if we examine a single time slice of the model, we see that it corresponds to a mixture distribution, with component densities given by $p(\mathbf{x}|\mathbf{z})$. It can therefore also be interpreted as an extension of a mixture model in which the choice of mixture component for each observation is not selected independently but depends on the choice of component for the previous observation. The HMM is widely used in speech recognition (Jelinek, 1997; Rabiner and Juang, 1993), natural language modelling (Manning and Schütze, 1999), on-line handwriting recognition (Nag *et al.*, 1986), and for the analysis of biological sequences such as proteins and DNA (Krogh *et al.*, 1994; Durbin *et al.*, 1998; Baldi and Brunak, 2001).

As in the case of a standard mixture model, the latent variables are the discrete multinomial variables $\mathbf{z}_n$ describing which component of the mixture is responsible for generating the corresponding observation $\mathbf{x}_n$. Again, it is convenient to use a 1-of-K coding scheme, as used for mixture models in Chapter 9. We now allow the probability distribution of $\mathbf{z}_n$ to depend on the state of the previous latent variable $\mathbf{z}_{n-1}$ through a conditional distribution $p(\mathbf{z}_n|\mathbf{z}_{n-1})$. Because the latent variables are K-dimensional binary variables, this conditional distribution corresponds to a table of numbers that we denote by $\mathbf{A}$, the elements of which are known as *transition probabilities*. They are given by $A_{jk} \equiv p(z_{nk} = 1|z_{n-1,j} = 1)$, and because they are probabilities, they satisfy $0 \leqslant A_{jk} \leqslant 1$ with $\sum_k A_{jk} = 1$, so that the matrix $\mathbf{A}$

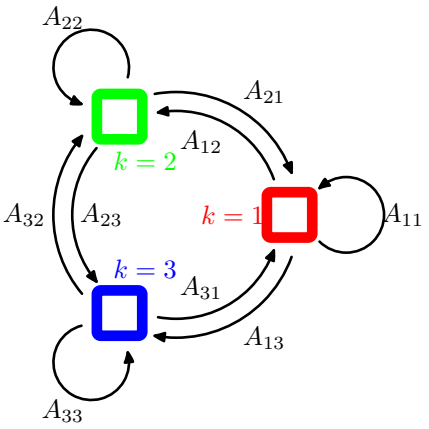


Figure 13.6 Transition diagram showing a model whose latent variables have three possible states corresponding to the three boxes. The black lines denote the elements of the transition matrix A_{jk}.

has $K(K-1)$ independent parameters. We can then write the conditional distribution explicitly in the form

$$p(\mathbf{z}_n|\mathbf{z}_{n-1},\mathbf{A}) = \prod_{k=1}^{K}\prod_{j=1}^{K} A_{jk}^{z_{n-1,j}z_{nk}}. \tag{13.7}$$

The initial latent node $\mathbf{z}_1$ is special in that it does not have a parent node, and so it has a marginal distribution $p(\mathbf{z}_1)$ represented by a vector of probabilities $\boldsymbol{\pi}$ with elements $\pi_k \equiv p(z_{1k} = 1)$, so that

$$p(\mathbf{z}_1|\boldsymbol{\pi}) = \prod_{k=1}^{K} \pi_k^{z_{1k}} \tag{13.8}$$

where $\sum_k \pi_k = 1$.

The transition matrix is sometimes illustrated diagrammatically by drawing the states as nodes in a state transition diagram as shown in Figure 13.6 for the case of $K = 3$. Note that this does not represent a probabilistic graphical model, because the nodes are not separate variables but rather states of a single variable, and so we have shown the states as boxes rather than circles.

It is sometimes useful to take a state transition diagram, of the kind shown in Figure 13.6, and unfold it over time. This gives an alternative representation of the transitions between latent states, known as a *lattice* or *trellis* diagram, and which is shown for the case of the hidden Markov model in Figure 13.7.

Section 8.4.5

The specification of the probabilistic model is completed by defining the conditional distributions of the observed variables $p(\mathbf{x}_n|\mathbf{z}_n, \boldsymbol{\phi})$, where $\boldsymbol{\phi}$ is a set of parameters governing the distribution. These are known as *emission probabilities*, and might for example be given by Gaussians of the form (9.11) if the elements of $\mathbf{x}$ are continuous variables, or by conditional probability tables if $\mathbf{x}$ is discrete. Because $\mathbf{x}_n$ is observed, the distribution $p(\mathbf{x}_n|\mathbf{z}_n, \boldsymbol{\phi})$ consists, for a given value of $\boldsymbol{\phi}$, of a vector of K numbers corresponding to the K possible states of the binary vector $\mathbf{z}_n$.

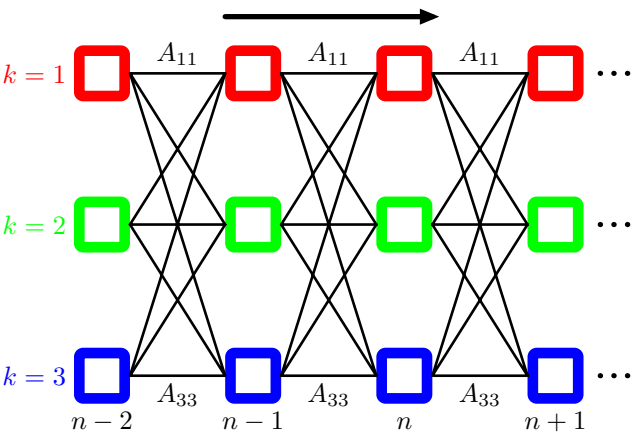


Figure 13.7 If we unfold the state transition diagram of Figure 13.6 over time, we obtain a lattice, or trellis, representation of the latent states. Each column of this diagram corresponds to one of the latent variables $\mathbf{z}_n$.

We can represent the emission probabilities in the form

$$p(\mathbf{x}_n|\mathbf{z}_n, \boldsymbol{\phi}) = \prod_{k=1}^{K} p(\mathbf{x}_n|\boldsymbol{\phi}_k)^{z_{nk}}. \tag{13.9}$$

We shall focus attention on *homogeneous* models for which all of the conditional distributions governing the latent variables share the same parameters $\mathbf{A}$, and similarly all of the emission distributions share the same parameters $\boldsymbol{\phi}$ (the extension to more general cases is straightforward). Note that a mixture model for an i.i.d. data set corresponds to the special case in which the parameters A_{jk} are the same for all values of j, so that the conditional distribution $p(\mathbf{z}_n|\mathbf{z}_{n-1})$ is independent of $\mathbf{z}_{n-1}$. This corresponds to deleting the horizontal links in the graphical model shown in Figure 13.5.

The joint probability distribution over both latent and observed variables is then given by

$$p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) = p(\mathbf{z}_1|\boldsymbol{\pi}) \left[\prod_{n=2}^{N} p(\mathbf{z}_n|\mathbf{z}_{n-1}, \mathbf{A})\right] \prod_{m=1}^{N} p(\mathbf{x}_m|\mathbf{z}_m, \boldsymbol{\phi}) \tag{13.10}$$

where $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$, and $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\phi}\}$ denotes the set of parameters governing the model. Most of our discussion of the hidden Markov model will be independent of the particular choice of the emission probabilities. Indeed, the model is tractable for a wide range of emission distributions including discrete tables, Gaussians, and mixtures of Gaussians. It is also possible to exploit discriminative models such as neural networks. These can be used to model the emission density $p(\mathbf{x}|\mathbf{z})$ directly, or to provide a representation for $p(\mathbf{z}|\mathbf{x})$ that can be converted into the required emission density $p(\mathbf{x}|\mathbf{z})$ using Bayes' theorem (Bishop *et al.*, 2004).

Exercise 13.4

We can gain a better understanding of the hidden Markov model by considering it from a generative point of view. Recall that to generate samples from a mixture of

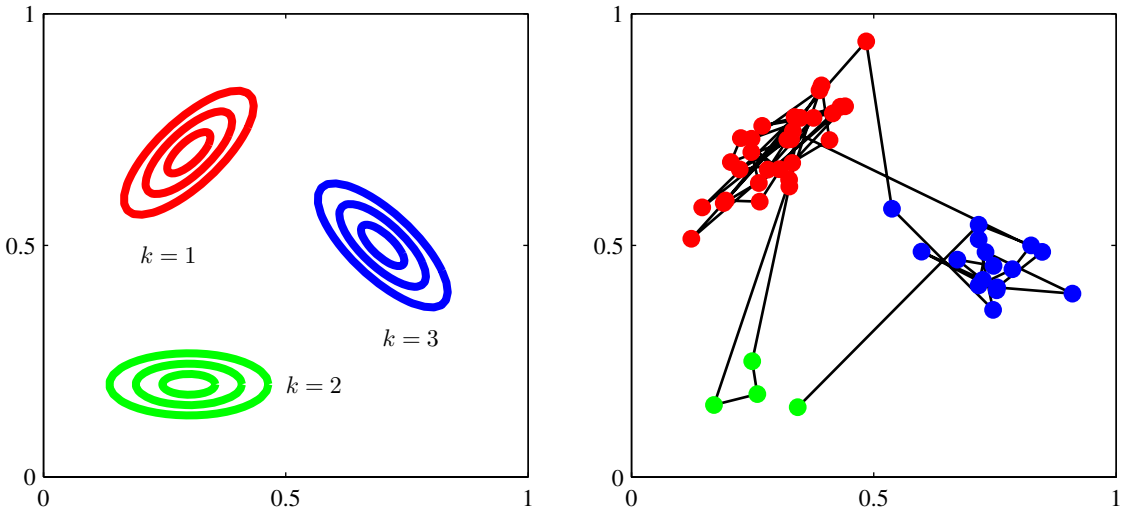


Figure 13.8 Illustration of sampling from a hidden Markov model having a 3-state latent variable $\mathbf{z}$ and a Gaussian emission model $p(\mathbf{x}|\mathbf{z})$ where $\mathbf{x}$ is 2-dimensional. (a) Contours of constant probability density for the emission distributions corresponding to each of the three states of the latent variable. (b) A sample of 50 points drawn from the hidden Markov model, colour coded according to the component that generated them and with lines connecting the successive observations. Here the transition matrix was fixed so that in any state there is a 5% probability of making a transition to each of the other states, and consequently a 90% probability of remaining in the same state.

Gaussians, we first chose one of the components at random with probability given by the mixing coefficients π_k and then generate a sample vector $\mathbf{x}$ from the corresponding Gaussian component. This process is repeated N times to generate a data set of N independent samples. In the case of the hidden Markov model, this procedure is modified as follows. We first choose the initial latent variable $\mathbf{z}_1$ with probabilities governed by the parameters π_k and then sample the corresponding observation $\mathbf{x}_1$. Now we choose the state of the variable $\mathbf{z}_2$ according to the transition probabilities $p(\mathbf{z}_2|\mathbf{z}_1)$ using the already instantiated value of $\mathbf{z}_1$. Thus suppose that the sample for $\mathbf{z}_1$ corresponds to state j. Then we choose the state k of $\mathbf{z}_2$ with probabilities A_{jk} for $k = 1, \ldots, K$. Once we know $\mathbf{z}_2$ we can draw a sample for $\mathbf{x}_2$ and also sample the next latent variable $\mathbf{z}_3$ and so on. This is an example of ancestral sampling for a directed graphical model. If, for instance, we have a model in which the diagonal transition elements A_{kk} are much larger than the off-diagonal elements, then a typical data sequence will have long runs of points generated from a single component, with infrequent transitions from one component to another. The generation of samples from a hidden Markov model is illustrated in Figure 13.8.

Section 8.1.2

left-to-right HMM

There are many variants of the standard HMM model, obtained for instance by imposing constraints on the form of the transition matrix $\mathbf{A}$ (Rabiner, 1989). Here we mention one of particular practical importance called the *left-to-right* HMM, which is obtained by setting the elements A_{jk} of $\mathbf{A}$ to zero if $k < j$, as illustrated in the

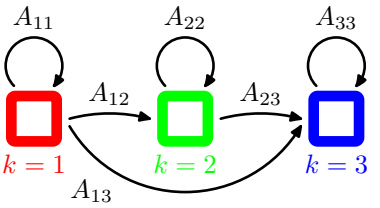


Figure 13.9 Example of the state transition diagram for a 3-state left-to-right hidden Markov model. Note that once a state has been vacated, it cannot later be re-entered.

state transition diagram for a 3-state HMM in Figure 13.9. Typically for such models the initial state probabilities for $p(\mathbf{z}_1)$ are modified so that $p(z_{11}) = 1$ and $p(z_{1j}) = 0$ for $j \neq 1$, in other words every sequence is constrained to start in state $j = 1$. The transition matrix may be further constrained to ensure that large changes in the state index do not occur, so that $A_{jk} = 0$ if $k > j + \Delta$. This type of model is illustrated using a lattice diagram in Figure 13.10.

Many applications of hidden Markov models, for example speech recognition, or on-line character recognition, make use of left-to-right architectures. As an illustration of the left-to-right hidden Markov model, we consider an example involving handwritten digits. This uses on-line data, meaning that each digit is represented by the trajectory of the pen as a function of time in the form of a sequence of pen coordinates, in contrast to the off-line digits data, discussed in Appendix A, which comprises static two-dimensional pixellated images of the ink. Examples of the on-line digits are shown in Figure 13.11. Here we train a hidden Markov model on a subset of data comprising 45 examples of the digit '2'. There are $K = 16$ states, each of which can generate a line segment of fixed length having one of 16 possible angles, and so the emission distribution is simply a 16×16 table of probabilities associated with the allowed angle values for each state index value. Transition probabilities are all set to zero except for those that keep the state index k the same or that increment it by 1, and the model parameters are optimized using 25 iterations of EM. We can gain some insight into the resulting model by running it generatively, as shown in Figure 13.11.

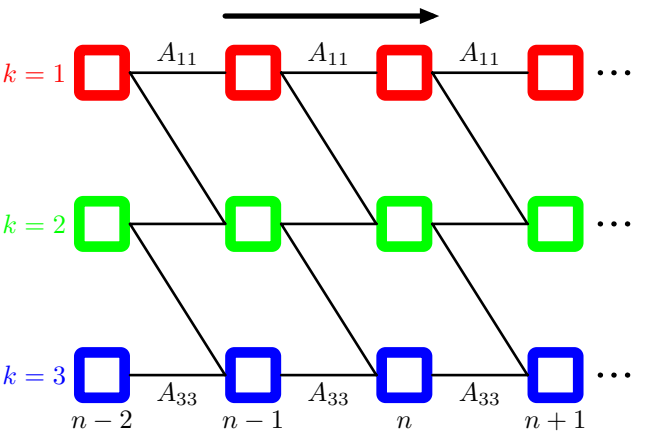


Figure 13.10 Lattice diagram for a 3-state left-to-right HMM in which the state index k is allowed to increase by at most 1 at each transition.

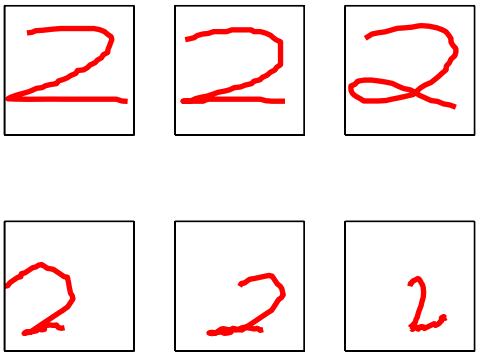


Figure 13.11 Top row: examples of on-line handwritten digits. Bottom row: synthetic digits sampled generatively from a left-to-right hidden Markov model that has been trained on a data set of 45 handwritten digits.

One of the most powerful properties of hidden Markov models is their ability to exhibit some degree of invariance to local warping (compression and stretching) of the time axis. To understand this, consider the way in which the digit '2' is written in the on-line handwritten digits example. A typical digit comprises two distinct sections joined at a cusp. The first part of the digit, which starts at the top left, has a sweeping arc down to the cusp or loop at the bottom left, followed by a second more-or-less straight sweep ending at the bottom right. Natural variations in writing style will cause the relative sizes of the two sections to vary, and hence the location of the cusp or loop within the temporal sequence will vary. From a generative perspective such variations can be accommodated by the hidden Markov model through changes in the number of transitions to the same state versus the number of transitions to the successive state. Note, however, that if a digit '2' is written in the reverse order, that is, starting at the bottom right and ending at the top left, then even though the pen tip coordinates may be identical to an example from the training set, the probability of the observations under the model will be extremely small. In the speech recognition context, warping of the time axis is associated with natural variations in the speed of speech, and again the hidden Markov model can accommodate such a distortion and not penalize it too heavily.

13.2.1 Maximum likelihood for the HMM

If we have observed a data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, we can determine the parameters of an HMM using maximum likelihood. The likelihood function is obtained from the joint distribution (13.10) by marginalizing over the latent variables

$$p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}). \tag{13.11}$$

似然函数的计算

Because the joint distribution $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ does not factorize over n (in contrast to the mixture distribution considered in Chapter 9), we cannot simply treat each of the summations over $\mathbf{z}_n$ independently. Nor can we perform the summations explicitly because there are N variables to be summed over, each of which has K states, resulting in a total of K^N terms. Thus the number of terms in the summation grows

exponentially with the length of the chain. In fact, the summation in (13.11) corresponds to summing over exponentially many paths through the lattice diagram in Figure 13.7.

We have already encountered a similar difficulty when we considered the inference problem for the simple chain of variables in Figure 8.32. There we were able to make use of the conditional independence properties of the graph to re-order the summations in order to obtain an algorithm whose cost scales linearly, instead of exponentially, with the length of the chain. We shall apply a similar technique to the hidden Markov model.

A further difficulty with the expression (13.11) for the likelihood function is that, because it corresponds to a generalization of a mixture distribution, it represents a summation over the emission models for different settings of the latent variables. Direct maximization of the likelihood function will therefore lead to complex expressions with no closed-form solutions, as was the case for simple mixture models (recall that a mixture model for i.i.d. data is a special case of the HMM).

Section 9.2

We therefore turn to the expectation maximization algorithm to find an efficient framework for maximizing the likelihood function in hidden Markov models. The EM algorithm starts with some initial selection for the model parameters, which we denote by $\boldsymbol{\theta}^{\text{old}}$. In the E step, we take these parameter values and find the posterior distribution of the latent variables $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$. We then use this posterior distribution to evaluate the expectation of the logarithm of the complete-data likelihood function, as a function of the parameters $\boldsymbol{\theta}$, to give the function $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ defined by

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}). \tag{13.12}$$

At this point, it is convenient to introduce some notation. We shall use $\gamma(\mathbf{z}_n)$ to denote the marginal posterior distribution of a latent variable $\mathbf{z}_n$, and $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$ to denote the joint posterior distribution of two successive latent variables, so that

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \tag{13.13}$$

$$\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{z}_{n-1}, \mathbf{z}_n|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}). \tag{13.14}$$

For each value of n, we can store $\gamma(\mathbf{z}_n)$ using a set of K nonnegative numbers that sum to unity, and similarly we can store $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$ using a $K \times K$ matrix of nonnegative numbers that again sum to unity. We shall also use $\gamma(z_{nk})$ to denote the conditional probability of $z_{nk} = 1$, with a similar use of notation for $\xi(z_{n-1,j}, z_{nk})$ and for other probabilistic variables introduced later. Because the expectation of a binary random variable is just the probability that it takes the value 1, we have

$$\gamma(z_{nk}) = \mathbb{E}[z_{nk}] = \sum_{\mathbf{z}_n} \gamma(\mathbf{z}) z_{nk} \tag{13.15}$$

$$\xi(z_{n-1,j}, z_{nk}) = \mathbb{E}[z_{n-1,j} z_{nk}] = \sum_{\mathbf{z}} \gamma(\mathbf{z}) z_{n-1,j} z_{nk}. \tag{13.16}$$

If we substitute the joint distribution $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ given by (13.10) into (13.12),

and make use of the definitions of γ and ξ , we obtain

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{k=1}^{K} \gamma(z_{1k}) \ln \pi_k + \sum_{n=2}^{N} \sum_{j=1}^{K} \sum_{k=1}^{K} \xi(z_{n-1,j}, z_{nk}) \ln A_{jk} + \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) \ln p(\mathbf{x}_n|\boldsymbol{\phi}_k). \tag{13.17}$$

E步中对γ量的计算则由后面章节给出。

The goal of the E step will be to evaluate the quantities $\gamma(\mathbf{z}_n)$ and $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$ efficiently, and we shall discuss this in detail shortly.

In the M step, we maximize $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ with respect to the parameters $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\phi}\}$ in which we treat $\gamma(\mathbf{z}_n)$ and $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$ as constant. Maximization with respect to $\boldsymbol{\pi}$ and $\mathbf{A}$ is easily achieved using appropriate Lagrange multipliers with the results

Exercise 13.5

$$\pi_k = \frac{\gamma(z_{1k})}{\sum_{j=1}^{K} \gamma(z_{1j})} \tag{13.18}$$

$$A_{jk} = \frac{\sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nk})}{\sum_{l=1}^{K} \sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nl})}. \tag{13.19}$$

The EM algorithm must be initialized by choosing starting values for $\boldsymbol{\pi}$ and $\mathbf{A}$, which should of course respect the summation constraints associated with their probabilistic interpretation. Note that any elements of $\boldsymbol{\pi}$ or $\mathbf{A}$ that are set to zero initially will remain zero in subsequent EM updates. A typical initialization procedure would involve selecting random starting values for these parameters subject to the summation and non-negativity constraints. Note that no particular modification to the EM results are required for the case of left-to-right models beyond choosing initial values for the elements A_{jk} in which the appropriate elements are set to zero, because these will remain zero throughout.

Exercise 13.6

这里是说在 left-to-right model 这类模型，可以通过将相应A_{jk}初始化为0，使得转移满足 left-to-right 的限制。

To maximize $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ with respect to $\boldsymbol{\phi}_k$, we notice that only the final term in (13.17) depends on $\boldsymbol{\phi}_k$, and furthermore this term has exactly the same form as the data-dependent term in the corresponding function for a standard mixture distribution for i.i.d. data, as can be seen by comparison with (9.40) for the case of a Gaussian mixture. Here the quantities $\gamma(z_{nk})$ are playing the role of the responsibilities. If the parameters $\boldsymbol{\phi}_k$ are independent for the different components, then this term decouples into a sum of terms one for each value of k, each of which can be maximized independently. We are then simply maximizing the weighted log likelihood function for the emission density $p(\mathbf{x}|\boldsymbol{\phi}_k)$ with weights $\gamma(z_{nk})$. Here we shall suppose that this maximization can be done efficiently. For instance, in the case of

Gaussian emission densities we have $p(\mathbf{x}|\boldsymbol{\phi}_k) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, and maximization of the function $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ then gives

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^{N} \gamma(z_{nk})} \tag{13.20}$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}}{\sum_{n=1}^{N} \gamma(z_{nk})}. \tag{13.21}$$

For the case of discrete multinomial observed variables, the conditional distribution of the observations takes the form

$$p(\mathbf{x}|\mathbf{z}) = \prod_{i=1}^{D} \prod_{k=1}^{K} \mu_{ik}^{x_i z_k} \tag{13.22}$$

Exercise 13.8 and the corresponding M-step equations are given by

$$\mu_{ik} = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) x_{ni}}{\sum_{n=1}^{N} \gamma(z_{nk})}. \tag{13.23}$$

An analogous result holds for Bernoulli observed variables.

The EM algorithm requires initial values for the parameters of the emission distribution. One way to set these is first to treat the data initially as i.i.d. and fit the emission density by maximum likelihood, and then use the resulting values to initialize the parameters for EM.

13.2.2 The forward-backward algorithm

Next we seek an efficient procedure for evaluating the quantities $\gamma(z_{nk})$ and $\xi(z_{n-1,j}, z_{nk})$, corresponding to the E step of the EM algorithm. The graph for the hidden Markov model, shown in Figure 13.5, is a tree, and so we know that the posterior distribution of the latent variables can be obtained efficiently using a two-stage message passing algorithm. In the particular context of the hidden Markov model, this is known as the *forward-backward* algorithm (Rabiner, 1989), or the *Baum-Welch* algorithm (Baum, 1972). There are in fact several variants of the basic algorithm, all of which lead to the exact marginals, according to the precise form of

Section 8.4

the messages that are propagated along the chain (Jordan, 2007). We shall focus on the most widely used of these, known as the alpha-beta algorithm.

这里从头开始指导 forward-backward algorithm，下一小节则可看到其等为 sum-product 算法的特例可很容易得到

As well as being of great practical importance in its own right, the forward-backward algorithm provides us with a nice illustration of many of the concepts introduced in earlier chapters. We shall therefore begin in this section with a 'conventional' derivation of the forward-backward equations, making use of the sum and product rules of probability, and exploiting conditional independence properties which we shall obtain from the corresponding graphical model using d-separation. Then in Section 13.2.3, we shall see how the forward-backward algorithm can be obtained very simply as a specific example of the sum-product algorithm introduced in Section 8.4.4.

It is worth emphasizing that evaluation of the posterior distributions of the latent variables is independent of the form of the emission density $p(\mathbf{x}|\mathbf{z})$ or indeed of whether the observed variables are continuous or discrete. All we require is the values of the quantities $p(\mathbf{x}_n|\mathbf{z}_n)$ for each value of $\mathbf{z}_n$ for every n. Also, in this section and the next we shall omit the explicit dependence on the model parameters $\boldsymbol{\theta}^{\text{old}}$ because these fixed throughout.

We therefore begin by writing down the following conditional independence properties (Jordan, 2007)

$$
\begin{aligned}
p(\mathbf{X}|\mathbf{z}_n) &= p(\mathbf{x}_1,\ldots,\mathbf{x}_n|\mathbf{z}_n) \\
&\quad\; p(\mathbf{x}_{n+1},\ldots,\mathbf{x}_N|\mathbf{z}_n) && (13.24)\\
p(\mathbf{x}_1,\ldots,\mathbf{x}_{n-1}|\mathbf{x}_n,\mathbf{z}_n) &= p(\mathbf{x}_1,\ldots,\mathbf{x}_{n-1}|\mathbf{z}_n) && (13.25)\\
p(\mathbf{x}_1,\ldots,\mathbf{x}_{n-1}|\mathbf{z}_{n-1},\mathbf{z}_n) &= p(\mathbf{x}_1,\ldots,\mathbf{x}_{n-1}|\mathbf{z}_{n-1}) && (13.26)\\
p(\mathbf{x}_{n+1},\ldots,\mathbf{x}_N|\mathbf{z}_n,\mathbf{z}_{n+1}) &= p(\mathbf{x}_{n+1},\ldots,\mathbf{x}_N|\mathbf{z}_{n+1}) && (13.27)\\
p(\mathbf{x}_{n+2},\ldots,\mathbf{x}_N|\mathbf{z}_{n+1},\mathbf{x}_{n+1}) &= p(\mathbf{x}_{n+2},\ldots,\mathbf{x}_N|\mathbf{z}_{n+1}) && (13.28)\\
p(\mathbf{X}|\mathbf{z}_{n-1},\mathbf{z}_n) &= p(\mathbf{x}_1,\ldots,\mathbf{x}_{n-1}|\mathbf{z}_{n-1}) \\
&\quad\; p(\mathbf{x}_n|\mathbf{z}_n)p(\mathbf{x}_{n+1},\ldots,\mathbf{x}_N|\mathbf{z}_n) && (13.29)\\
p(\mathbf{x}_{N+1}|\mathbf{X},\mathbf{z}_{N+1}) &= p(\mathbf{x}_{N+1}|\mathbf{z}_{N+1}) && (13.30)\\
p(\mathbf{z}_{N+1}|\mathbf{z}_N,\mathbf{X}) &= p(\mathbf{z}_{N+1}|\mathbf{z}_N) && (13.31)
\end{aligned}
$$

where $\mathbf{X} = \{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$. These relations are most easily proved using d-separation. For instance in the ~~first~~ second of these results, we note that every path from any one of the nodes $\mathbf{x}_1,\ldots,\mathbf{x}_{n-1}$ to the node $\mathbf{x}_n$ passes through the node $\mathbf{z}_n$, which is observed. Because all such paths are head-to-tail, it follows that the conditional independence property must hold. The reader should take a few moments to verify each of these properties in turn, as an exercise in the application of d-separation. These relations can also be proved directly, though with significantly greater effort, from the joint distribution for the hidden Markov model using the sum and product rules of probability.

Exercise 13.10 求证计算

Let us begin by evaluating $\gamma(z_{nk})$. Recall that for a discrete multinomial random variable the expected value of one of its components is just the probability of that component having the value 1. Thus we are interested in finding the posterior distribution $p(\mathbf{z}_n|\mathbf{x}_1,\ldots,\mathbf{x}_N)$ of $\mathbf{z}_n$ given the observed data set $\mathbf{x}_1,\ldots,\mathbf{x}_N$. This

represents a vector of length K whose entries correspond to the expected values of z_{nk}. Using Bayes' theorem, we have

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n|\mathbf{X}) = \frac{p(\mathbf{X}|\mathbf{z}_n)p(\mathbf{z}_n)}{p(\mathbf{X})}. \tag{13.32}$$

Note that the denominator $p(\mathbf{X})$ is implicitly conditioned on the parameters $\boldsymbol{\theta}^{\text{old}}$ of the HMM and hence represents the likelihood function. Using the conditional independence property (13.24), together with the product rule of probability, we obtain

$$\gamma(\mathbf{z}_n) = \frac{p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{z}_n)p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N|\mathbf{z}_n)}{p(\mathbf{X})} = \frac{\alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})} \tag{13.33}$$

where we have defined

$$\alpha(\mathbf{z}_n) \equiv p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{z}_n) \tag{13.34}$$

$$\beta(\mathbf{z}_n) \equiv p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N|\mathbf{z}_n). \tag{13.35}$$

The quantity $\alpha(\mathbf{z}_n)$ represents the joint probability of observing all of the given data up to time n and the value of $\mathbf{z}_n$, whereas $\beta(\mathbf{z}_n)$ represents the conditional probability of all future data from time $n+1$ up to N given the value of $\mathbf{z}_n$. Again, $\alpha(\mathbf{z}_n)$ and $\beta(\mathbf{z}_n)$ each represent a set of K numbers, one for each of the possible settings of the 1-of-K coded binary vector $\mathbf{z}_n$. We shall use the notation $\alpha(z_{nk})$ to denote the value of $\alpha(\mathbf{z}_n)$ when $z_{nk} = 1$, with an analogous interpretation of $\beta(z_{nk})$.

We now derive recursion relations that allow $\alpha(\mathbf{z}_n)$ and $\beta(\mathbf{z}_n)$ to be evaluated efficiently. Again, we shall make use of conditional independence properties, in particular (13.25) and (13.26), together with the sum and product rules, allowing us to express $\alpha(\mathbf{z}_n)$ in terms of $\alpha(\mathbf{z}_{n-1})$ as follows

$$\begin{aligned}
\alpha(\mathbf{z}_n) &= p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{z}_n) \\
&= p(\mathbf{x}_1, \ldots, \mathbf{x}_n|\mathbf{z}_n)p(\mathbf{z}_n) \\
&= p(\mathbf{x}_n|\mathbf{z}_n)p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}|\mathbf{z}_n)p(\mathbf{z}_n) \\
&= p(\mathbf{x}_n|\mathbf{z}_n)p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}, \mathbf{z}_n) \\
&= p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}, \mathbf{z}_{n-1}, \mathbf{z}_n) \\
&= p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}, \mathbf{z}_n|\mathbf{z}_{n-1})p(\mathbf{z}_{n-1}) \\
&= p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}|\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1})p(\mathbf{z}_{n-1}) \\
&= p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}, \mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1})
\end{aligned}$$

Making use of the definition (13.34) for $\alpha(\mathbf{z}_n)$, we then obtain

$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1}). \tag{13.36}$$

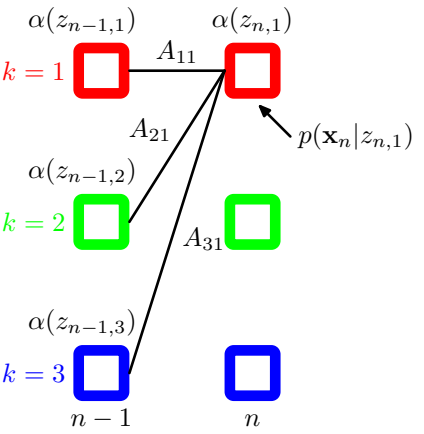


Figure 13.12 Illustration of the forward recursion (13.36) for evaluation of the α variables. In this fragment of the lattice, we see that the quantity $\alpha(z_{n1})$ is obtained by taking the elements $\alpha(z_{n-1,j})$ of $\alpha(\mathbf{z}_{n-1})$ at step $n-1$ and summing them up with weights given by A_{j1}, corresponding to the values of $p(\mathbf{z}_n|\mathbf{z}_{n-1})$, and then multiplying by the data contribution $p(\mathbf{x}_n|z_{n1})$.

It is worth taking a moment to study this recursion relation in some detail. Note that there are K terms in the summation, and the right-hand side has to be evaluated for each of the K values of $\mathbf{z}_n$ so each step of the α recursion has computational cost that scaled like $O(K^2)$. The forward recursion equation for $\alpha(\mathbf{z}_n)$ is illustrated using a lattice diagram in Figure 13.12.

In order to start this recursion, we need an initial condition that is given by

$$\alpha(\mathbf{z}_1) = p(\mathbf{x}_1, \mathbf{z}_1) = p(\mathbf{z}_1)p(\mathbf{x}_1|\mathbf{z}_1) = \prod_{k=1}^{K} \{\pi_k p(\mathbf{x}_1|\boldsymbol{\phi}_k)\}^{z_{1k}} \tag{13.37}$$

which tells us that $\alpha(z_{1k})$, for $k = 1, \ldots, K$, takes the value $\pi_k p(\mathbf{x}_1|\boldsymbol{\phi}_k)$. Starting at the first node of the chain, we can then work along the chain and evaluate $\alpha(\mathbf{z}_n)$ for every latent node. Because each step of the recursion involves multiplying by a $K \times K$ matrix, the overall cost of evaluating these quantities for the whole chain is of $O(K^2N)$.

We can similarly find a recursion relation for the quantities $\beta(\mathbf{z}_n)$ by making use of the conditional independence properties (13.27) and (13.28) giving

$$\begin{aligned}
\beta(\mathbf{z}_n) &= p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N|\mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N, \mathbf{z}_{n+1}|\mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N|\mathbf{z}_n, \mathbf{z}_{n+1})p(\mathbf{z}_{n+1}|\mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N|\mathbf{z}_{n+1})p(\mathbf{z}_{n+1}|\mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+2}, \ldots, \mathbf{x}_N|\mathbf{z}_{n+1})p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1})p(\mathbf{z}_{n+1}|\mathbf{z}_n).
\end{aligned}$$

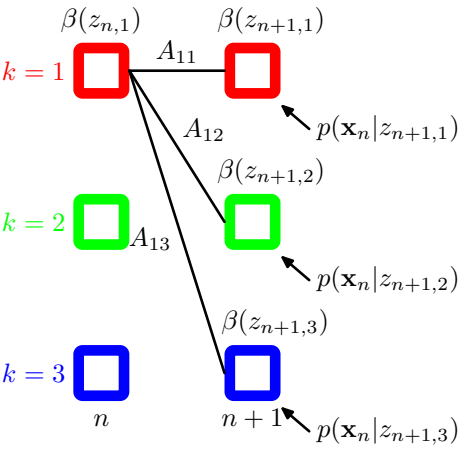


Figure 13.13 Illustration of the backward recursion (13.38) for evaluation of the β variables. In this fragment of the lattice, we see that the quantity $\beta(z_{n,1})$ is obtained by taking the components $\beta(z_{n+1,k})$ of $\beta(\mathbf{z}_{n+1})$ at step $n+1$ and summing them up with weights given by the products of A_{1k}, corresponding to the values of $p(\mathbf{z}_{n+1}|\mathbf{z}_n)$ and the corresponding values of the emission density $p(\mathbf{x}_n|z_{n+1,k})$.

Making use of the definition (13.35) for $\beta(\mathbf{z}_n)$, we then obtain

$$\beta(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} \beta(\mathbf{z}_{n+1}) p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1}) p(\mathbf{z}_{n+1}|\mathbf{z}_n). \tag{13.38}$$

Note that in this case we have a backward message passing algorithm that evaluates $\beta(\mathbf{z}_n)$ in terms of $\beta(\mathbf{z}_{n+1})$. At each step, we absorb the effect of observation $\mathbf{x}_{n+1}$ through the emission probability $p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1})$, multiply by the transition matrix $p(\mathbf{z}_{n+1}|\mathbf{z}_n)$, and then marginalize out $\mathbf{z}_{n+1}$. This is illustrated in Figure 13.13.

Again we need a starting condition for the recursion, namely a value for $\beta(\mathbf{z}_N)$. This can be obtained by setting $n = N$ in (13.33) and replacing $\alpha(\mathbf{z}_N)$ with its definition (13.34) to give

$$p(\mathbf{z}_N|\mathbf{X}) = \frac{p(\mathbf{X}, \mathbf{z}_N)\beta(\mathbf{z}_N)}{p(\mathbf{X})} \tag{13.39}$$

which we see will be correct provided we take $\beta(\mathbf{z}_N) = 1$ for all settings of $\mathbf{z}_N$.

计算P(X)，P(X)作为似然函数是我们优化的终极目标，计算P(X)可用于监控优化的情况

In the M step equations, the quantity $p(\mathbf{X})$ will cancel out, as can be seen, for instance, in the M-step equation for $\boldsymbol{\mu}_k$ given by (13.20), which takes the form

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{n} \gamma(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^{n} \gamma(z_{nk})} = \frac{\sum_{n=1}^{n} \alpha(z_{nk}) \beta(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^{n} \alpha(z_{nk}) \beta(z_{nk})}. \tag{13.40}$$

However, the quantity $p(\mathbf{X})$ represents the likelihood function whose value we typically wish to monitor during the EM optimization, and so it is useful to be able to evaluate it. If we sum both sides of (13.33) over $\mathbf{z}_n$, and use the fact that the left-hand side is a normalized distribution, we obtain

$$p(\mathbf{X}) = \sum_{\mathbf{z}_n} \alpha(\mathbf{z}_n)\beta(\mathbf{z}_n). \tag{13.41}$$

Thus we can evaluate the likelihood function by computing this sum, for any convenient choice of n. For instance, if we only want to evaluate the likelihood function, then we can do this by running the α recursion from the start to the end of the chain, and then use this result for $n = N$, making use of the fact that $\beta(\mathbf{z}_N)$ is a vector of 1s. In this case no β recursion is required, and we simply have

$$p(\mathbf{X}) = \sum_{\mathbf{z}_N} \alpha(\mathbf{z}_N). \tag{13.42}$$

Let us take a moment to interpret this result for $p(\mathbf{X})$. Recall that to compute the likelihood we should take the joint distribution $p(\mathbf{X}, \mathbf{Z})$ and sum over all possible values of $\mathbf{Z}$. Each such value represents a particular choice of hidden state for every time step, in other words every term in the summation is a path through the lattice diagram, and recall that there are exponentially many such paths. By expressing the likelihood function in the form (13.42), we have reduced the computational cost from being exponential in the length of the chain to being linear by swapping the order of the summation and multiplications, so that at each time step n we sum the contributions from all paths passing through each of the states z_{nk} to give the intermediate quantities $\alpha(\mathbf{z}_n)$.

Next we consider the evaluation of the quantities $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$, which correspond to the values of the conditional probabilities $p(\mathbf{z}_{n-1}, \mathbf{z}_n|\mathbf{X})$ for each of the $K \times K$ settings for $(\mathbf{z}_{n-1}, \mathbf{z}_n)$. Using the definition of $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$, and applying Bayes' theorem, we have

$$\begin{aligned}
\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) &= p(\mathbf{z}_{n-1}, \mathbf{z}_n|\mathbf{X}) \\
&= \frac{p(\mathbf{X}|\mathbf{z}_{n-1}, \mathbf{z}_n)p(\mathbf{z}_{n-1}, \mathbf{z}_n)}{p(\mathbf{X})} \\
&= \frac{p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}|\mathbf{z}_{n-1})p(\mathbf{x}_n|\mathbf{z}_n)p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N|\mathbf{z}_n)p(\mathbf{z}_n|\mathbf{z}_{n-1})p(\mathbf{z}_{n-1})}{p(\mathbf{X})} \\
&= \frac{\alpha(\mathbf{z}_{n-1})p(\mathbf{x}_n|\mathbf{z}_n)p(\mathbf{z}_n|\mathbf{z}_{n-1})\beta(\mathbf{z}_n)}{p(\mathbf{X})}
\end{aligned} \tag{13.43}$$

where we have made use of the conditional independence property (13.29) together with the definitions of $\alpha(\mathbf{z}_n)$ and $\beta(\mathbf{z}_n)$ given by (13.34) and (13.35). Thus we can calculate the $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$ directly by using the results of the α and β recursions.

Let us summarize the steps required to train a hidden Markov model using the EM algorithm. We first make an initial selection of the parameters $\boldsymbol{\theta}^{\text{old}}$ where $\boldsymbol{\theta} \equiv (\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\phi})$. The $\mathbf{A}$ and $\boldsymbol{\pi}$ parameters are often initialized either uniformly or randomly from a uniform distribution (respecting their non-negativity and summation constraints). Initialization of the parameters $\boldsymbol{\phi}$ will depend on the form of the distribution. For instance in the case of Gaussians, the parameters $\boldsymbol{\mu}_k$ might be initialized by applying the K-means algorithm to the data, and $\boldsymbol{\Sigma}_k$ might be initialized to the covariance matrix of the corresponding K means cluster. Then we run both the forward α recursion and the backward β recursion and use the results to evaluate $\gamma(\mathbf{z}_n)$ and $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$. At this stage, we can also evaluate the likelihood function.

This completes the E step, and we use the results to find a revised set of parameters $\boldsymbol{\theta}^{\text{new}}$ using the M-step equations from Section 13.2.1. We then continue to alternate between E and M steps until some convergence criterion is satisfied, for instance when the change in the likelihood function is below some threshold.

Note that in these recursion relations the observations enter through conditional distributions of the form $p(\mathbf{x}_n|\mathbf{z}_n)$. The recursions are therefore independent of the type or dimensionality of the observed variables or the form of this conditional distribution, so long as its value can be computed for each of the K possible states of $\mathbf{z}_n$. ~~Since the observed variables $\{\mathbf{x}_n\}$ are fixed, the quantities $p(\mathbf{x}_n|\mathbf{z}_n)$ can be pre-computed as functions of $\mathbf{z}_n$ at the start of the EM algorithm, and remain fixed throughout.~~

We have seen in earlier chapters that the maximum likelihood approach is most effective when the number of data points is large in relation to the number of parameters. Here we note that a hidden Markov model can be trained effectively, using maximum likelihood, provided the training sequence is sufficiently long. Alternatively, we can make use of multiple shorter sequences, which requires a straightforward modification of the hidden Markov model EM algorithm. In the case of left-to-right models, this is particularly important because, in a given observation sequence, a given state transition corresponding to a nondiagonal element of $\mathbf{A}$ will ~~seen~~ occur at most once.

Another quantity of interest is the predictive distribution, in which the observed data is $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and we wish to predict $\mathbf{x}_{N+1}$, which would be important for real-time applications such as financial forecasting. Again we make use of the sum and product rules together with the conditional independence properties (~~13.29~~ 13.30) and (13.31) giving

$$
\begin{aligned}
p(\mathbf{x}_{N+1}|\mathbf{X}) &= \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1}, \mathbf{z}_{N+1}|\mathbf{X}) \\
&= \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1}|\mathbf{z}_{N+1}) p(\mathbf{z}_{N+1}|\mathbf{X}) \\
&= \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1}|\mathbf{z}_{N+1}) \sum_{\mathbf{z}_N} p(\mathbf{z}_{N+1}, \mathbf{z}_N|\mathbf{X}) \\
&= \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1}|\mathbf{z}_{N+1}) \sum_{\mathbf{z}_N} p(\mathbf{z}_{N+1}|\mathbf{z}_N) p(\mathbf{z}_N|\mathbf{X}) \\
&= \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1}|\mathbf{z}_{N+1}) \sum_{\mathbf{z}_N} p(\mathbf{z}_{N+1}|\mathbf{z}_N) \frac{p(\mathbf{z}_N, \mathbf{X})}{p(\mathbf{X})} \\
&= \frac{1}{p(\mathbf{X})} \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1}|\mathbf{z}_{N+1}) \sum_{\mathbf{z}_N} p(\mathbf{z}_{N+1}|\mathbf{z}_N) \alpha(\mathbf{z}_N) \qquad (13.44)
\end{aligned}
$$

which can be evaluated by first running a forward α recursion and then computing the final summations over $\mathbf{z}_N$ and $\mathbf{z}_{N+1}$. The result of the first summation over $\mathbf{z}_N$ can be stored and used once the value of $\mathbf{x}_{N+1}$ is observed in order to run the α recursion forward to the next step in order to predict the subsequent value $\mathbf{x}_{N+2}$.

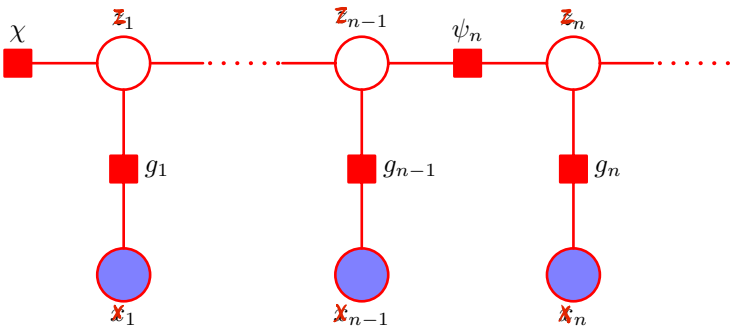


Figure 13.14 A fragment of the factor graph representation for the hidden Markov model.

Note that in (13.44), the influence of all data from $\mathbf{x}_1$ to $\mathbf{x}_N$ is summarized in the K values of $\alpha(\mathbf{z}_N)$. Thus the predictive distribution can be carried forward indefinitely using a fixed amount of storage, as may be required for real-time applications.

MLE→MAP→fully Bayesian CVL)

Here we have discussed the estimation of the parameters of an HMM using maximum likelihood. This framework is easily extended to regularized maximum likelihood by introducing priors over the model parameters $\boldsymbol{\pi}$, $\mathbf{A}$ and $\boldsymbol{\phi}$ whose values are then estimated by maximizing their posterior probability. This can again be done using the EM algorithm in which the E step is the same as discussed above, and the M step involves adding the log of the prior distribution $p(\boldsymbol{\theta})$ to the function $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ before maximization and represents a straightforward application of the techniques developed at various points in this book. Furthermore, we can use variational methods to give a fully Bayesian treatment of the HMM in which we marginalize over the parameter distributions (MacKay, 1997). As with maximum likelihood, this leads to a two-pass forward-backward recursion to compute posterior probabilities.

Section 10.1

13.2.3 The sum-product algorithm for the HMM

The directed graph that represents the hidden Markov model, shown in Figure 13.5, is a tree and so we can solve the problem of finding local marginals for the hidden variables using the sum-product algorithm. Not surprisingly, this turns out to be equivalent to the forward-backward algorithm considered in the previous section, and so the sum-product algorithm therefore provides us with a simple way to derive the alpha-beta recursion formulae.

Section 8.4.4

也sum-product 算法推导 13.2.2节的alpha-beta算法 构造因子图

We begin by transforming the directed graph of Figure 13.5 into a factor graph, of which a representative fragment is shown in Figure 13.14. This form of the factor graph shows all variables, both latent and observed, explicitly. However, for the purpose of solving the inference problem, we shall always be conditioning on the variables $\mathbf{x}_1, \ldots, \mathbf{x}_N$, and so we can simplify the factor graph by absorbing the emission probabilities into the transition probability factors. This leads to the simplified factor graph representation in Figure 13.15, in which the factors are given by

$$h(\mathbf{z}_1) = p(\mathbf{z}_1)p(\mathbf{x}_1|\mathbf{z}_1) \tag{13.45}$$

$$f_n(\mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{z}_n|\mathbf{z}_{n-1})p(\mathbf{x}_n|\mathbf{z}_n). \tag{13.46}$$

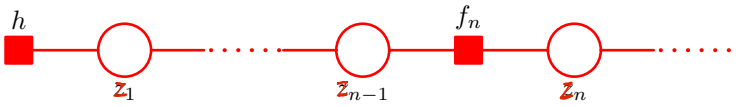


Figure 13.15 A simplified form of factor graph to describe the hidden Markov model.

To derive the alpha-beta algorithm, we denote the final hidden variable $\mathbf{z}_N$ as the root node, and first pass messages from the leaf node h to the root. From the general results (8.66) and (8.69) for message propagation, we see that the messages which are propagated in the hidden Markov model take the form

$$\mu_{\mathbf{z}_{n-1}\to f_n}(\mathbf{z}_{n-1}) = \mu_{f_{n-1}\to\mathbf{z}_{n-1}}(\mathbf{z}_{n-1}) \tag{13.47}$$

$$\mu_{f_n\to\mathbf{z}_n}(\mathbf{z}_n) = \sum_{\mathbf{z}_{n-1}} f_n(\mathbf{z}_{n-1},\mathbf{z}_n)\mu_{\mathbf{z}_{n-1}\to f_n}(\mathbf{z}_{n-1}) \tag{13.48}$$

These equations represent the propagation of messages forward along the chain and are equivalent to the alpha recursions derived in the previous section, as we shall now show. Note that because the variable nodes $\mathbf{z}_n$ have only two neighbours, they perform no computation.

We can eliminate $\mu_{\mathbf{z}_{n-1}\to f_n}(\mathbf{z}_{n-1})$ from (13.48) using (13.47) to give a recursion for the $f \to \mathbf{z}$ messages of the form

$$\mu_{f_n\to\mathbf{z}_n}(\mathbf{z}_n) = \sum_{\mathbf{z}_{n-1}} f_n(\mathbf{z}_{n-1},\mathbf{z}_n)\mu_{f_{n-1}\to\mathbf{z}_{n-1}}(\mathbf{z}_{n-1}). \tag{13.49}$$

If we now recall the definition (13.46), and if we define

$$\alpha(\mathbf{z}_n) = \mu_{f_n\to\mathbf{z}_n}(\mathbf{z}_n) \tag{13.50}$$

then we obtain the alpha recursion given by (13.36). We also need to verify that the quantities $\alpha(\mathbf{z}_n)$ are themselves equivalent to those defined previously. This is easily done by using the initial condition (8.71) and noting that $\alpha(\mathbf{z}_1)$ is given by $h(\mathbf{z}_1) = p(\mathbf{z}_1)p(\mathbf{x}_1|\mathbf{z}_1)$ which is identical to (13.37). Because the initial α is the same, and because they are iteratively computed using the same equation, all subsequent α quantities must be the same.

Next we consider the messages that are propagated from the root node back to the leaf node. These take the form

$$\mu_{f_{n+1}\to\mathbf{z}_n}(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} f_{n+1}(\mathbf{z}_n,\mathbf{z}_{n+1})\mu_{f_{n+2}\to\mathbf{z}_{n+1}}(\mathbf{z}_{n+1}) \tag{13.51}$$

where, as before, we have eliminated the messages of the type $\mathbf{z} \to f$ since the variable nodes perform no computation. Using the definition (13.46) to substitute for $f_{n+1}(\mathbf{z}_n,\mathbf{z}_{n+1})$, and defining

$$\beta(\mathbf{z}_n) = \mu_{f_{n+1}\to\mathbf{z}_n}(\mathbf{z}_n) \tag{13.52}$$

we obtain the beta recursion given by (13.38). Again, we can verify that the beta variables themselves are equivalent by noting that (8.70) implies that the initial message send by the root variable node is $\mu_{\mathbf{z}_N \to f_N}(\mathbf{z}_N) = 1$, which is identical to the initialization of $\beta(\mathbf{z}_N)$ given in Section 13.2.2.

The sum-product algorithm also specifies how to evaluate the marginals once all the messages have been evaluated. In particular, the result (8.63) shows that the local marginal at the node $\mathbf{z}_n$ is given by the product of the incoming messages. Because we have conditioned on the variables $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, we are computing the joint distribution

$$p(\mathbf{z}_n, \mathbf{X}) = \mu_{f_n \to \mathbf{z}_n}(\mathbf{z}_n)\mu_{f_{n+1} \to \mathbf{z}_n}(\mathbf{z}_n) = \alpha(\mathbf{z}_n)\beta(\mathbf{z}_n). \tag{13.53}$$

Dividing both sides by $p(\mathbf{X})$, we then obtain

$$\gamma(\mathbf{z}_n) = \frac{p(\mathbf{z}_n, \mathbf{X})}{p(\mathbf{X})} = \frac{\alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})} \tag{13.54}$$

Exercise 13.11 in agreement with (13.33). The result (13.43) can similarly be derived from (8.72).

13.2.4 Scaling factors

There is an important issue that must be addressed before we can make use of the forward backward algorithm in practice. From the recursion relation (13.36), we note that at each step the new value $\alpha(\mathbf{z}_n)$ is obtained from the previous value $\alpha(\mathbf{z}_{n-1})$ by multiplying by quantities $p(\mathbf{z}_n|\mathbf{z}_{n-1})$ and $p(\mathbf{x}_n|\mathbf{z}_n)$. Because these probabilities are often significantly less than unity, as we work our way forward along the chain, the values of $\alpha(\mathbf{z}_n)$ can go to zero exponentially quickly. For moderate lengths of chain (say 100 or so), the calculation of the $\alpha(\mathbf{z}_n)$ will soon exceed the dynamic range of the computer, even if double precision floating point is used.

In the case of i.i.d. data, we implicitly circumvented this problem with the evaluation of likelihood functions by taking logarithms. Unfortunately, this will not help here because we are forming sums of products of small numbers (we are in fact implicitly summing over all possible paths through the lattice diagram of Figure 13.7). We therefore work with re-scaled versions of $\alpha(\mathbf{z}_n)$ and $\beta(\mathbf{z}_n)$ whose values remain of order unity. As we shall see, the corresponding scaling factors cancel out when we use these re-scaled quantities in the EM algorithm.

In (13.34), we defined $\alpha(\mathbf{z}_n) = p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{z}_n)$ representing the joint distribution of all the observations up to $\mathbf{x}_n$ and the latent variable $\mathbf{z}_n$. Now we define a normalized version of α given by

$$\widehat{\alpha}(\mathbf{z}_n) = p(\mathbf{z}_n|\mathbf{x}_1, \ldots, \mathbf{x}_n) = \frac{\alpha(\mathbf{z}_n)}{p(\mathbf{x}_1, \ldots, \mathbf{x}_n)} \tag{13.55}$$

which we expect to be well behaved numerically because it is a probability distribution over K variables for any value of n. In order to relate the scaled and original alpha variables, we introduce scaling factors defined by conditional distributions over the observed variables

$$c_n = p(\mathbf{x}_n|\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}). \tag{13.56}$$

From the product rule, we then have

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_{m=1}^{n} c_m \tag{13.57}$$

and so

$$\alpha(\mathbf{z}_n) = p(\mathbf{z}_n|\mathbf{x}_1, \ldots, \mathbf{x}_n)p(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \left(\prod_{m=1}^{n} c_m\right) \widehat{\alpha}(\mathbf{z}_n). \tag{13.58}$$

We can then turn the recursion equation (13.36) for α into one for $\widehat{\alpha}$ given by

$$c_n \widehat{\alpha}(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \widehat{\alpha}(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1}). \tag{13.59}$$

Note that at each stage of the forward message passing phase, used to evaluate $\widehat{\alpha}(\mathbf{z}_n)$, we have to evaluate and store c_n, which is easily done because it is the coefficient that normalizes the right-hand side of (13.59) to give $\widehat{\alpha}(\mathbf{z}_n)$.

We can similarly define re-scaled variables $\widehat{\beta}(\mathbf{z}_n)$ using

$$\beta(\mathbf{z}_n) = \left(\prod_{m=n+1}^{N} c_m\right) \widehat{\beta}(\mathbf{z}_n) \tag{13.60}$$

which will again remain within machine precision because, from (13.35), the quantities $\widehat{\beta}(\mathbf{z}_n)$ are simply the ratio of two conditional probabilities

$$\widehat{\beta}(\mathbf{z}_n) = \frac{p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N|\mathbf{z}_n)}{p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N|\mathbf{x}_1, \ldots, \mathbf{x}_n)}. \tag{13.61}$$

The recursion result (13.38) for β then gives the following recursion for the re-scaled variables

$$c_{n+1}\widehat{\beta}(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} \widehat{\beta}(\mathbf{z}_{n+1}) p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1}) p(\mathbf{z}_{n+1}|\mathbf{z}_n). \tag{13.62}$$

In applying this recursion relation, we make use of the scaling factors c_n that were previously computed in the α phase.

From (13.57), we see that the likelihood function can be found using

$$p(\mathbf{X}) = \prod_{n=1}^{N} c_n. \tag{13.63}$$

Exercise 13.15

Similarly, using (13.33) and (13.43), together with (13.63), we see that the required marginals are given by

$$\gamma(\mathbf{z}_n) = \widehat{\alpha}(\mathbf{z}_n)\widehat{\beta}(\mathbf{z}_n) \tag{13.64}$$

$$\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) = c_n \widehat{\alpha}(\mathbf{z}_{n-1}) p(\mathbf{x}_n|\mathbf{z}_n) p(\mathbf{z}_n|\mathbf{z}_{-1}) \widehat{\beta}(\mathbf{z}_n). \tag{13.65}$$

Finally, we note that there is an alternative formulation of the forward-backward algorithm (Jordan, 2007) in which the backward pass is defined by a recursion based the quantities $\gamma(\mathbf{z}_n) = \widehat{\alpha}(\mathbf{z}_n)\widehat{\beta}(\mathbf{z}_n)$ instead of using $\widehat{\beta}(\mathbf{z}_n)$. This α–γ recursion requires that the forward pass be completed first so that all the quantities $\widehat{\alpha}(\mathbf{z}_n)$ are available for the backward pass, whereas the forward and backward passes of the α–β algorithm can be done independently. Although these two algorithms have comparable computational cost, the α–β version is the most commonly encountered one in the case of hidden Markov models, whereas for linear dynamical systems a recursion analogous to the α–γ form is more usual.

Section 13.3

13.2.5 The Viterbi algorithm

In many applications of hidden Markov models, the latent variables have some meaningful interpretation, and so it is often of interest to find the most probable sequence of hidden states for a given observation sequence. For instance in speech recognition, we might wish to find the most probable phoneme sequence for a given series of acoustic observations. Because the graph for the hidden Markov model is a directed tree, this problem can be solved exactly using the max-sum algorithm. We recall from our discussion in Section 8.4.5 that the problem of finding the most probable sequence of latent states is not the same as that of finding the set of states that are individually the most probable. The latter problem can be solved by first running the forward-backward (sum-product) algorithm to find the latent variable marginals $\gamma(\mathbf{z}_n)$ and then maximizing each of these individually (Duda *et al.*, 2001). However, the set of such states will not, in general, correspond to the most probable sequence of states. In fact, this set of states might even represent a sequence having zero probability, if it so happens that two successive states, which in isolation are individually the most probable, are such that the transition matrix element connecting them is zero.

In practice, we are usually interested in finding the most probable *sequence* of states, and this can be solved efficiently using the max-sum algorithm, which in the context of hidden Markov models is known as the *Viterbi* algorithm (Viterbi, 1967). Note that the max-sum algorithm works with log probabilities and so there is no need to use re-scaled variables as was done with the forward-backward algorithm. Figure 13.16 shows a fragment of the hidden Markov model expanded as lattice diagram. As we have already noted, the number of possible paths through the lattice grows exponentially with the length of the chain. The Viterbi algorithm searches this space of paths efficiently to find the most probable path with a computational cost that grows only linearly with the length of the chain.

As with the sum-product algorithm, we first represent the hidden Markov model as a factor graph, as shown in Figure 13.15. Again, we treat the variable node $\mathbf{z}_N$ as the root, and pass messages to the root starting with the leaf nodes. Using the results (8.93) and (8.94), we see that the messages passed in the max-sum algorithm are given by

$$\mu_{\mathbf{z}_n \to f_{n+1}}(\mathbf{z}_n) = \mu_{f_n \to \mathbf{z}_n}(\mathbf{z}_n) \tag{13.66}$$

$$\mu_{f_{n+1} \to \mathbf{z}_{n+1}}(\mathbf{z}_{n+1}) = \max_{\mathbf{z}_n} \left\{ \ln f_{n+1}(\mathbf{z}_n, \mathbf{z}_{n+1}) + \mu_{\mathbf{z}_n \to f_{n+1}}(\mathbf{z}_n) \right\}. \tag{13.67}$$

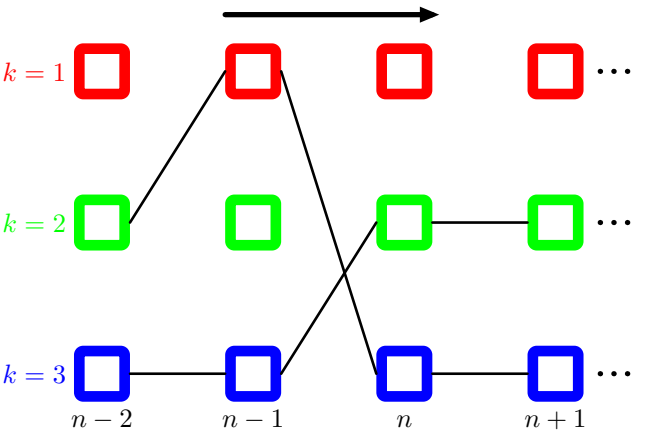


Figure 13.16 A fragment of the HMM lattice showing two possible paths. The Viterbi algorithm efficiently determines the most probable path from amongst the exponentially many possibilities. For any given path, the corresponding probability is given by the product of the elements of the transition matrix A_{jk}, corresponding to the probabilities $p(\mathbf{z}_{n+1}|\mathbf{z}_n)$ for each segment of the path, along with the emission densities $p(\mathbf{x}_n|k)$ associated with each node on the path.

If we eliminate $\mu_{\mathbf{z}_n \to f_{n+1}}(\mathbf{z}_n)$ between these two equations, and make use of (13.46), we obtain a recursion for the $f \to \mathbf{z}$ messages of the form

$$\omega(\mathbf{z}_{n+1}) = \ln p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1}) + \max_{\mathbf{z}_n} \{\ln p(\mathbf{z}_{n+1}|\mathbf{z}_n) + \omega(\mathbf{z}_n)\} \tag{13.68}$$

where we have introduced the notation $\omega(\mathbf{z}_n) \equiv \mu_{f_n \to \mathbf{z}_n}(\mathbf{z}_n)$.

From (8.95) and (8.96), these messages are initialized using

$$\omega(\mathbf{z}_1) = \ln p(\mathbf{z}_1) + \ln p(\mathbf{x}_1|\mathbf{z}_1). \tag{13.69}$$

where we have used (13.45). Note that to keep the notation uncluttered, we omit the dependence on the model parameters $\boldsymbol{\theta}$ that are held fixed when finding the most probable sequence.

从头开始推导Viterbi算法，即证明从第一性原理出发

The Viterbi algorithm can also be derived directly from the definition (13.6) of the joint distribution by taking the logarithm and then exchanging maximizations and summations. It is easily seen that the quantities $\omega(\mathbf{z}_n)$ have the probabilistic interpretation

Exercise 13.16

$$\omega(\mathbf{z}_n) = \max_{\mathbf{z}_1,\ldots,\mathbf{z}_{n-1}} \ln p(\mathbf{x}_1,\ldots,\mathbf{x}_n,\mathbf{z}_1,\ldots,\mathbf{z}_n). \tag{13.70}$$

回溯得到max prob时经过的点的值

Once we have completed the final maximization over $\mathbf{z}_N$, we will obtain the value of the joint distribution $p(\mathbf{X},\mathbf{Z})$ corresponding to the most probable path. We also wish to find the sequence of latent variable values that corresponds to this path. To do this, we simply make use of the back-tracking procedure discussed in Section 8.4.5. Specifically, we note that the maximization over $\mathbf{z}_n$ must be performed for each of the K possible values of $\mathbf{z}_{n+1}$. Suppose we keep a record of the values of $\mathbf{z}_n$ that correspond to the maxima for each value of the K values of $\mathbf{z}_{n+1}$. Let us denote this function by $\psi(k_n)$ where $k \in \{1,\ldots,K\}$. Once we have passed messages to the end of the chain and found the most probable state of $\mathbf{z}_N$, we can then use this function to backtrack along the chain by applying it recursively

$$k_{n-1}^{\max} = \psi(k_{n+1}^{\max}). \tag{13.71}$$

Intuitively, we can understand the Viterbi algorithm as follows. Naively, we could consider explicitly all of the exponentially many paths through the lattice, evaluate the probability for each, and then select the path having the highest probability. However, we notice that we can make a dramatic saving in computational cost as follows. Suppose that for each path we evaluate its probability by summing up products of transition and emission probabilities as we work our way forward along each path through the lattice. Consider a particular time step n and a particular state k at that time step. There will be many possible paths converging on the corresponding node in the lattice diagram. However, we need only retain that particular path that so far has the highest probability. Because there are K states at time step n, we need to keep track of K such paths. At time step $n+1$, there will be K^2 possible paths to consider, comprising K possible paths leading out of each of the K current states, but again we need only retain K of these corresponding to the best path for each state at time $n+1$. When we reach the final time step N we will discover which state corresponds to the overall most probable path. Because there is a unique path coming into that state we can trace the path back to step $N-1$ to see what state it occupied at that time, and so on back through the lattice to the state $n = 1$.

13.2.6 Extensions of the hidden Markov model

The basic hidden Markov model, along with the standard training algorithm based on maximum likelihood, has been extended in numerous ways to meet the requirements of particular applications. Here we discuss a few of the more important examples.

We see from the digits example in Figure 13.11 that hidden Markov models can be quite poor generative models for the data, because many of the synthetic digits look quite unrepresentative of the training data. If the goal is sequence classification, there can be significant benefit in determining the parameters of hidden Markov models using discriminative rather than maximum likelihood techniques. Suppose we have a training set of R observation sequences $\mathbf{X}_r$, where $r = 1, \ldots, R$, each of which is labelled according to its class m, where $m = 1, \ldots, M$. For each class, we have a separate hidden Markov model with its own parameters $\boldsymbol{\theta}_m$, and we treat the problem of determining the parameter values as a standard classification problem in which we optimize the cross-entropy

$$\sum_{r=1}^{R} \ln p(m_r|\mathbf{X}_r). \tag{13.72}$$

Using Bayes' theorem this can be expressed in terms of the sequence probabilities associated with the hidden Markov models

$$\sum_{r=1}^{R} \ln \left\{ \frac{p(\mathbf{X}_r|\boldsymbol{\theta}_r)p(m_r)}{\sum_{l=1}^{M} p(\mathbf{X}_r|\boldsymbol{\theta}_l)p(l_r)} \right\} \tag{13.73}$$

where $p(m)$ is the prior probability of class m. Optimization of this cost function is more complex than for maximum likelihood (Kapadia, 1998), and in particular

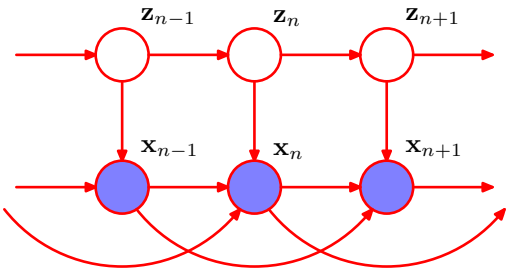


Figure 13.17 Section of an autoregressive hidden Markov model, in which the distribution of the observation $\mathbf{x}_n$ depends on a subset of the previous observations as well as on the hidden state $\mathbf{z}_n$. In this example, the distribution of $\mathbf{x}_n$ depends on the two previous observations $\mathbf{x}_{n-1}$ and $\mathbf{x}_{n-2}$.

requires that every training sequence be evaluated under each of the models in order to compute the denominator in (13.73). Hidden Markov models, coupled with discriminative training methods, are widely used in speech recognition (Kapadia, 1998).

A significant weakness of the hidden Markov model is the way in which it represents the distribution of times for which the system remains in a given state. To see the problem, note that the probability that a sequence sampled from a given hidden Markov model will spend precisely T steps in state k and then make a transition to a different state is given by

$$p(T) = (A_{kk})^T(1 - A_{kk}) \propto \exp\left(T \ln A_{kk}\right) \tag{13.74}$$

and so is an exponentially decaying function of T. For many applications, this will be a very unrealistic model of state duration. The problem can be resolved by modelling state duration directly in which the diagonal coefficients A_{kk} are all set to zero, and each state k is explicitly associated with a probability distribution $p(T|k)$ of possible duration times. From a generative point of view, when a state k is entered, a value T representing the number of time steps that the system will remain in state k is then drawn from $p(T|k)$. The model then emits T values of the observed variable $\mathbf{x}_t$, which are generally assumed to be independent so that the corresponding emission density is simply $\prod_{t=1}^{T} p(\mathbf{x}_t|k)$. This approach requires some straightforward modifications to the EM optimization procedure (Rabiner, 1989).

Another limitation of the standard HMM is that it is poor at capturing long-range correlations between the observed variables (i.e., between variables that are separated by many time steps) because these must be mediated via the first-order Markov chain of hidden states. Longer-range effects could in principle be included by adding extra links to the graphical model of Figure 13.5. One way to address this is to generalize the HMM to give the *autoregressive hidden Markov model* (Ephraim *et al.*, 1989), an example of which is shown in Figure 13.17. For discrete observations, this corresponds to expanded tables of conditional probabilities for the emission distributions. In the case of a Gaussian emission density, we can use the linear-Gaussian framework in which the conditional distribution for $\mathbf{x}_n$ given the values of the previous observations, and the value of $\mathbf{z}_n$, is a Gaussian whose mean is a linear combination of the values of the conditioning variables. Clearly the number of additional links in the graph must be limited to avoid an excessive ~~the~~ number of free parameters. In the example shown in Figure 13.17, each observation depends on

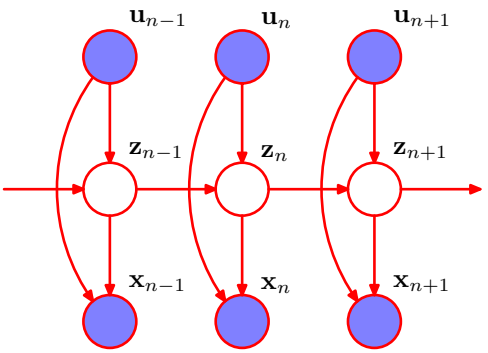


Figure 13.18 Example of an input-output hidden Markov model. In this case, both the emission probabilities and the transition probabilities depend on the values of a sequence of observations $\mathbf{u}_1, \ldots, \mathbf{u}_N$.

the two preceding observed variables as well as on the hidden state. Although this graph looks messy, we can again appeal to d-separation to see that in fact it still has a simple probabilistic structure. In particular, if we imagine conditioning on $\mathbf{z}_n$ we see that, as with the standard HMM, the values of $\mathbf{z}_{n-1}$ and $\mathbf{z}_{n+1}$ are independent, corresponding to the conditional independence property (13.5). This is easily verified by noting that every path from node $\mathbf{z}_{n-1}$ to node $\mathbf{z}_{n+1}$ passes through at least one observed node that is head-to-tail with respect to that path. As a consequence, we can again use a forward-backward recursion in the E step of the EM algorithm to determine the posterior distributions of the latent variables in a computational time that is linear in the length of the chain. Similarly, the M step involves only a minor modification of the standard M-step equations. In the case of Gaussian emission densities this involves estimating the parameters using the standard linear regression equations, discussed in Chapter 3.

We have seen that the autoregressive HMM appears as a natural extension of the standard HMM when viewed as a graphical model. In fact the probabilistic graphical modelling viewpoint motivates a plethora of different graphical structures based on the HMM. Another example is the *input-output* hidden Markov model (Bengio and Frasconi, 1995), in which we have a sequence of observed variables $\mathbf{u}_1, \ldots, \mathbf{u}_N$, in addition to the output variables $\mathbf{x}_1, \ldots, \mathbf{x}_N$, whose values influence either the distribution of latent variables or output variables, or both. An example is shown in Figure 13.18. This extends the HMM framework to the domain of supervised learning for sequential data. It is again easy to show, through the use of the d-separation criterion, that the Markov property (13.5) for the chain of latent variables still holds. To verify this, simply note that there is only one path from node $\mathbf{z}_{n-1}$ to node $\mathbf{z}_{n+1}$ and this is head-to-tail with respect to the observed node $\mathbf{z}_n$. This conditional independence property again allows the formulation of a computationally efficient learning algorithm. In particular, we can determine the parameters $\boldsymbol{\theta}$ of the model by maximizing the likelihood function $L(\boldsymbol{\theta}) = p(\mathbf{X}|\mathbf{U}, \boldsymbol{\theta})$ where $\mathbf{U}$ is a matrix whose rows are given by $\mathbf{u}_n^{\mathrm{T}}$. As a consequence of the conditional independence property (13.5) this likelihood function can be maximized efficiently using an EM algorithm in which the E step involves forward and backward recursions.

Exercise 13.18

Another variant of the HMM worthy of mention is the *factorial hidden Markov model* (Ghahramani and Jordan, 1997), in which there are multiple independent

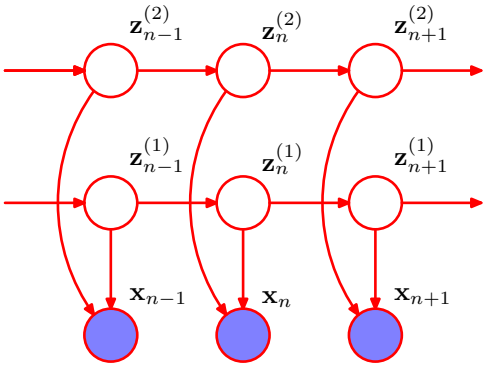


Figure 13.19 A factorial hidden Markov model comprising two Markov chains of latent variables. For continuous observed variables $\mathbf{x}$, one possible choice of emission model is a linear-Gaussian density in which the mean of the Gaussian is a linear combination of the states of the corresponding latent variables.

Markov chains of latent variables, and the distribution of the observed variable at a given time step is conditional on the states of all of the corresponding latent variables at that same time step. Figure 13.19 shows the corresponding graphical model. The motivation for considering factorial HMM can be seen by noting that in order to represent, say, 10 bits of information at a given time step, a standard HMM would need $K = 2^{10} = 1024$ latent states, whereas a factorial HMM could make use of 10 binary latent chains. The primary disadvantage of factorial HMMs, however, lies in the additional complexity of training them. The M step for the factorial HMM model is straightforward. However, observation of the $\mathbf{x}$ variables introduces dependencies between the latent chains, leading to difficulties with the E step. This can be seen by noting that in Figure 13.19, the variables $\mathbf{z}_n^{(1)}$ and $\mathbf{z}_n^{(2)}$ are connected by a path which is head-to-head at node $\mathbf{x}_n$ and hence they are not d-separated. The exact E step for this model does *not* correspond to running forward and backward recursions along the M Markov chains independently. This is confirmed by noting that the key conditional independence property (13.5) is not satisfied for the individual Markov chains in the factorial HMM model, as is shown using d-separation in Figure 13.20. Now suppose that there are M chains of hidden nodes and for simplicity suppose that all latent variables have the same number K of states. Then one approach would be to note that there are K^M combinations of latent variables at a given time step

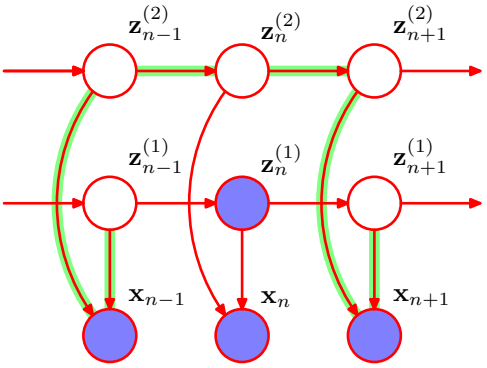


Figure 13.20 Example of a path, highlighted in green, which is head-to-head at the observed nodes $\mathbf{x}_{n-1}$ and $\mathbf{x}_{n+1}$, and head-to-tail at the unobserved nodes $\mathbf{z}_{n-1}^{(2)}$, $\mathbf{z}_n^{(2)}$ and $\mathbf{z}_{n+1}^{(2)}$. Thus the path is not blocked and so the conditional independence property (13.5) does not hold for the individual latent chains of the factorial HMM model. As a consequence, there is no efficient exact E step for this model.

and so we can transform the model into an equivalent standard HMM having a single chain of latent variables each of which has K^M latent states. We can then run the standard forward-backward recursions in the E step. This has computational complexity $O(NK^{2M})$ that is exponential in the number M of latent chains and so will be intractable for anything other than small values of M. One solution would be to use sampling methods (discussed in Chapter 11). As an elegant deterministic alternative, Ghahramani and Jordan (1997) exploited variational inference techniques to obtain a tractable algorithm for approximate inference. This can be done using a simple variational posterior distribution that is fully factorized with respect to the latent variables, or alternatively by using a more powerful approach in which the variational distribution is described by independent Markov chains corresponding to the chains of latent variables in the original model. In the latter case, the variational inference algorithms involves running independent forward and backward recursions along each chain, which is computationally efficient and yet is also able to capture correlations between variables within the same chain.

Section 10.1

Clearly, there are many possible probabilistic structures that can be constructed according to the needs of particular applications. Graphical models provide a general technique for motivating, describing, and analysing such structures, and variational methods provide a powerful framework for performing inference in those models for which exact solution is intractable.

13.3. Linear Dynamical Systems

In order to motivate the concept of linear dynamical systems, let us consider the following simple problem, which often arises in practical settings. Suppose we wish to measure the value of an unknown quantity $\mathbf{z}$ using a noisy sensor that returns a observation $\mathbf{x}$ representing the value of $\mathbf{z}$ plus zero-mean Gaussian noise. Given a single measurement, our best guess for $\mathbf{z}$ is to assume that $\mathbf{z} = \mathbf{x}$. However, we can improve our estimate for $\mathbf{z}$ by taking lots of measurements and averaging them, because the random noise terms will tend to cancel each other. Now let's make the situation more complicated by assuming that we wish to measure a quantity $\mathbf{z}$ that is changing over time. We can take regular measurements of $\mathbf{x}$ so that at some point in time we have obtained $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and we wish to find the corresponding values $\mathbf{z}_1, \ldots, \mathbf{z}_N$. If we simply average the measurements, the error due to random noise will be reduced, but unfortunately we will just obtain a single averaged estimate, in which we have averaged over the changing value of $\mathbf{z}$, thereby introducing a new source of error.

Intuitively, we could imagine doing a bit better as follows. To estimate the value of $\mathbf{z}_N$, we take only the most recent few measurements, say $\mathbf{x}_{N-L}, \ldots, \mathbf{x}_N$ and just average these. If $\mathbf{z}$ is changing slowly, and the random noise level in the sensor is high, it would make sense to choose a relatively long window of observations to average. Conversely, if the signal is changing quickly, and the noise levels are small, we might be better just to use $\mathbf{x}_N$ directly as our estimate of $\mathbf{z}_N$. Perhaps we could do even better if we take a weighted average, in which more recent measurements

make a greater contribution than less recent ones.

Although this sort of intuitive argument seems plausible, it does not tell us how to form a weighted average, and any sort of hand-crafted weighing is hardly likely to be optimal. Fortunately, we can address problems such as this much more systematically by defining a probabilistic model that captures the time evolution and measurement processes and then applying the inference and learning methods developed in earlier chapters. Here we shall focus on a widely used model known as a *linear dynamical system*.

As we have seen, the HMM corresponds to the state space model shown in Figure 13.5 in which the latent variables are discrete but with arbitrary emission probability distributions. This graph of course describes a much broader class of probability distributions, all of which factorize according to (13.6). We now consider extensions to other distributions for the latent variables. In particular, we consider continuous latent variables in which the summations of the sum-product algorithm become integrals. The general form of the inference algorithms will, however, be the same as for the hidden Markov model. It is interesting to note that, historically, hidden Markov models and linear dynamical systems were developed independently. Once they are both expressed as graphical models, however, the deep relationship between them immediately becomes apparent.

One key requirement is that we retain an efficient algorithm for inference which is linear in the length of the chain. This requires that, for instance, when we take a quantity $\widehat{\alpha}(\mathbf{z}_{n-1})$, representing the posterior probability of $\mathbf{z}_{n-1}$ given observations $\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}$, and multiply by the transition probability $p(\mathbf{z}_n|\mathbf{z}_{n-1})$ and the emission probability $p(\mathbf{x}_n|\mathbf{z}_n)$ and then marginalize over $\mathbf{z}_{n-1}$, we obtain a distribution over $\mathbf{z}_n$ that is of the same functional form as that over $\widehat{\alpha}(\mathbf{z}_{n-1})$. That is to say, the distribution must not become more complex at each stage, but must only change in its parameter values. Not surprisingly, the only distributions that have this property of being closed under multiplication are those belonging to the exponential family.

Here we consider the most important example from a practical perspective, which is the Gaussian. In particular, we consider a linear-Gaussian state space model so that the latent variables $\{\mathbf{z}_n\}$, as well as the observed variables $\{\mathbf{x}_n\}$, are multivariate Gaussian distributions whose means are linear functions of the states of their parents in the graph. We have seen that a directed graph of linear-Gaussian units is equivalent to a joint Gaussian distribution over all of the variables. Furthermore, marginals such as $\widehat{\alpha}(\mathbf{z}_n)$ are also Gaussian, so that the functional form of the messages is preserved and we will obtain an efficient inference algorithm. By contrast, suppose that the emission densities $p(\mathbf{x}_n|\mathbf{z}_n)$ comprise a mixture of K Gaussians each of which has a mean that is linear in $\mathbf{z}_n$. Then even if $\widehat{\alpha}(\mathbf{z}_1)$ is Gaussian, the quantity $\widehat{\alpha}(\mathbf{z}_2)$ will be a mixture of K Gaussians, $\widehat{\alpha}(\mathbf{z}_3)$ will be a mixture of K^2 Gaussians, and so on, and exact inference will not be of practical value.

We have seen that the hidden Markov model can be viewed as an extension of the mixture models of Chapter 9 to allow for sequential correlations in the data. In a similar way, we can view the linear dynamical system as a generalization of the continuous latent variable models of Chapter 12 such as probabilistic PCA and factor analysis. Each pair of nodes $\{\mathbf{z}_n, \mathbf{x}_n\}$ represents a linear-Gaussian latent variable

修正 Page 637–643 Equation (13.77)–Equation (13.110): All instances of V_0 should be replaced by P_0 , in equations as well as in the text.

model for that particular observation. However, the latent variables $\{\mathbf{z}_n\}$ are no longer treated as independent but now form a Markov chain.

Because the model is represented by a tree-structured directed graph, inference problems can be solved efficiently using the sum-product algorithm. The forward recursions, analogous to the α messages of the hidden Markov model, are known as the *Kalman filter* equations (Kalman, 1960; Zarchan and Musoff, 2005), and the backward recursions, analogous to the β messages, are known as the *Kalman smoother* equations, or the *Rauch-Tung-Striebel* (RTS) equations (Rauch *et al.*, 1965). The Kalman filter is widely used in many real-time tracking applications.

Because the linear dynamical system is a linear-Gaussian model, the joint distribution over all variables, as well as all marginals and conditionals, will be Gaussian. It follows that the sequence of individually most probable latent variable values is the same as the most probable latent sequence. There is thus no need to consider the analogue of the Viterbi algorithm for the linear dynamical system.

Exercise 13.19

Because the model has linear-Gaussian conditional distributions, we can write the transition and emission distributions in the general form

$$p(\mathbf{z}_n|\mathbf{z}_{n-1}) = \mathcal{N}(\mathbf{z}_n|\mathbf{A}\mathbf{z}_{n-1}, \boldsymbol{\Gamma}) \tag{13.75}$$

$$p(\mathbf{x}_n|\mathbf{z}_n) = \mathcal{N}(\mathbf{x}_n|\mathbf{C}\mathbf{z}_n, \boldsymbol{\Sigma}). \tag{13.76}$$

The initial latent variable also has a Gaussian distribution which we write as

$$p(\mathbf{z}_1) = \mathcal{N}(\mathbf{z}_1|\boldsymbol{\mu}_0, \mathbf{V}_0). \tag{13.77}$$

Note that in order to simplify the notation, we have omitted additive constant terms from the means of the Gaussians. In fact, it is straightforward to include them if desired. Traditionally, these distributions are more commonly expressed in an equivalent form in terms of noisy linear equations given by

Exercise 13.24

$$\mathbf{z}_n = \mathbf{A}\mathbf{z}_{n-1} + \mathbf{w}_n \tag{13.78}$$

$$\mathbf{x}_n = \mathbf{C}\mathbf{z}_n + \mathbf{v}_n \tag{13.79}$$

$$\mathbf{z}_1 = \boldsymbol{\mu}_0 + \mathbf{u} \tag{13.80}$$

where the noise terms have the distributions

$$\mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \boldsymbol{\Gamma}) \tag{13.81}$$

$$\mathbf{v} \sim \mathcal{N}(\mathbf{v}|\mathbf{0}, \boldsymbol{\Sigma}) \tag{13.82}$$

$$\mathbf{u} \sim \mathcal{N}(\mathbf{u}|\mathbf{0}, \mathbf{V}_0). \tag{13.83}$$

The parameters of the model, denoted by $\boldsymbol{\theta} = \{\mathbf{A}, \boldsymbol{\Gamma}, \mathbf{C}, \boldsymbol{\Sigma}, \boldsymbol{\mu}_0, \mathbf{V}_0\}$, can be determined using maximum likelihood through the EM algorithm. In the E step, we need to solve the inference problem of determining the local posterior marginals for the latent variables, which can be solved efficiently using the sum-product algorithm, as we discuss in the next section.

13.3.1 Inference in LDS

We now turn to the problem of finding the marginal distributions for the latent variables conditional on the observation sequence. For given parameter settings, we also wish to make predictions of the next latent state $\mathbf{z}_n$ and of the next observation $\mathbf{x}_n$ conditioned on the observed data $\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}$ for use in real-time applications. These inference problems can be solved efficiently using the sum-product algorithm, which in the context of the linear dynamical system gives rise to the Kalman filter and Kalman smoother equations.

It is worth emphasizing that because the linear dynamical system is a linear-Gaussian model, the joint distribution over all latent and observed variables is simply a Gaussian, and so in principle we could solve inference problems by using the standard results derived in previous chapters for the marginals and conditionals of a multivariate Gaussian. The role of the sum-product algorithm is to provide a more efficient way to perform such computations.

Linear dynamical systems have the identical factorization, given by (13.6), to hidden Markov models, and are again described by the factor graphs in Figures 13.14 and 13.15. Inference algorithms therefore take precisely the same form except that summations over latent variables are replaced by integrations. We begin by considering the forward equations in which we treat $\mathbf{z}_N$ as the root node, and propagate messages from the leaf node $h(\mathbf{z}_1)$ to the root. From (13.77), the initial message will be Gaussian, and because each of the factors is Gaussian, all subsequent messages will also be Gaussian. By convention, we shall propagate messages that are normalized marginal distributions corresponding to $p(\mathbf{z}_n|\mathbf{x}_1, \ldots, \mathbf{x}_n)$, which we denote by

$$\widehat{\alpha}(\mathbf{z}_n) = \mathcal{N}(\mathbf{z}_n|\boldsymbol{\mu}_n, \mathbf{V}_n). \tag{13.84}$$

This is precisely analogous to the propagation of scaled variables $\widehat{\alpha}(\mathbf{z}_n)$ given by (13.59) in the discrete case of the hidden Markov model, and so the recursion equation now takes the form

$$c_n\widehat{\alpha}(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \int \widehat{\alpha}(\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1})\,\mathrm{d}\mathbf{z}_{n-1}. \tag{13.85}$$

Substituting for the conditionals $p(\mathbf{z}_n|\mathbf{z}_{n-1})$ and $p(\mathbf{x}_n|\mathbf{z}_n)$, using (13.75) and (13.76), respectively, and making use of (13.84), we see that (13.85) becomes

$$c_n\mathcal{N}(\mathbf{z}_n|\boldsymbol{\mu}_n, \mathbf{V}_n) = \mathcal{N}(\mathbf{x}_n|\mathbf{C}\mathbf{z}_n, \boldsymbol{\Sigma}) \int \mathcal{N}(\mathbf{z}_n|\mathbf{A}\mathbf{z}_{n-1}, \boldsymbol{\Gamma})\mathcal{N}(\mathbf{z}_{n-1}|\boldsymbol{\mu}_{n-1}, \mathbf{V}_{n-1})\,\mathrm{d}\mathbf{z}_{n-1}. \tag{13.86}$$

Here we are supposing that $\boldsymbol{\mu}_{n-1}$ and $\mathbf{V}_{n-1}$ are known, and by evaluating the integral in (13.86), we wish to determine values for $\boldsymbol{\mu}_n$ and $\mathbf{V}_n$. The integral is easily evaluated by making use of the result (2.115), from which it follows that

$$\int \mathcal{N}(\mathbf{z}_n|\mathbf{A}\mathbf{z}_{n-1}, \boldsymbol{\Gamma})\mathcal{N}(\mathbf{z}_{n-1}|\boldsymbol{\mu}_{n-1}, \mathbf{V}_{n-1})\,\mathrm{d}\mathbf{z}_{n-1} = \mathcal{N}(\mathbf{z}_n|\mathbf{A}\boldsymbol{\mu}_{n-1}, \mathbf{P}_{n-1}) \tag{13.87}$$

where we have defined

$$\mathbf{P}_{n-1} = \mathbf{A}\mathbf{V}_{n-1}\mathbf{A}^{\mathrm{T}} + \boldsymbol{\Gamma}. \tag{13.88}$$

We can now combine this result with the first factor on the right-hand side of (13.86) by making use of (2.115) and (2.116) to give

$$\begin{aligned} \boldsymbol{\mu}_n &= \mathbf{A}\boldsymbol{\mu}_{n-1} + \mathbf{K}_n(\mathbf{x}_n - \mathbf{C}\mathbf{A}\boldsymbol{\mu}_{n-1}) && (13.89)\\ \mathbf{V}_n &= (\mathbf{I} - \mathbf{K}_n\mathbf{C})\mathbf{P}_{n-1} && (13.90)\\ c_n &= \mathcal{N}(\mathbf{x}_n|\mathbf{C}\mathbf{A}\boldsymbol{\mu}_{n-1}, \mathbf{C}\mathbf{P}_{n-1}\mathbf{C}^{\mathrm{T}} + \boldsymbol{\Sigma}). && (13.91) \end{aligned}$$

Here we have made use of the matrix inverse identities (C.5) and (C.7) and also defined the *Kalman gain matrix*

$$\mathbf{K}_n = \mathbf{P}_{n-1}\mathbf{C}^{\mathrm{T}}\left(\mathbf{C}\mathbf{P}_{n-1}\mathbf{C}^{\mathrm{T}} + \boldsymbol{\Sigma}\right)^{-1}. \tag{13.92}$$

Thus, given the values of $\boldsymbol{\mu}_{n-1}$ and $\mathbf{V}_{n-1}$, together with the new observation $\mathbf{x}_n$, we can evaluate the Gaussian marginal for $\mathbf{z}_n$ having mean $\boldsymbol{\mu}_n$ and covariance $\mathbf{V}_n$, as well as the normalization coefficient c_n.

The initial conditions for these recursion equations are obtained from

$$c_1\widehat{\alpha}(\mathbf{z}_1) = p(\mathbf{z}_1)p(\mathbf{x}_1|\mathbf{z}_1). \tag{13.93}$$

Because $p(\mathbf{z}_1)$ is given by (13.77), and $p(\mathbf{x}_1|\mathbf{z}_1)$ is given by (13.76), we can again make use of (2.115) to calculate c_1 and (2.116) to calculate $\boldsymbol{\mu}_1$ and $\mathbf{V}_1$ giving

$$\begin{aligned} \boldsymbol{\mu}_1 &= \boldsymbol{\mu}_0 + \mathbf{K}_1(\mathbf{x}_1 - \mathbf{C}\boldsymbol{\mu}_0) && (13.94)\\ \mathbf{V}_1 &= (\mathbf{I} - \mathbf{K}_1\mathbf{C})\mathbf{V}_0 && (13.95)\\ c_1 &= \mathcal{N}(\mathbf{x}_1|\mathbf{C}\boldsymbol{\mu}_0, \mathbf{C}\mathbf{V}_0\mathbf{C}^{\mathrm{T}} + \boldsymbol{\Sigma}) && (13.96) \end{aligned}$$

where

$$\mathbf{K}_1 = \mathbf{V}_0\mathbf{C}^{\mathrm{T}}\left(\mathbf{C}\mathbf{V}_0\mathbf{C}^{\mathrm{T}} + \boldsymbol{\Sigma}\right)^{-1}. \tag{13.97}$$

Similarly, the likelihood function for the linear dynamical system is given by (13.63) in which the factors c_n are found using the Kalman filtering equations.

对两项的理解(13.89)

We can interpret the steps involved in going from the posterior marginal over $\mathbf{z}_{n-1}$ to the posterior marginal over $\mathbf{z}_n$ as follows. In (13.89), we can view the quantity $\mathbf{A}\boldsymbol{\mu}_{n-1}$ as the prediction of the mean over $\mathbf{z}_n$ obtained by simply taking the mean over $\mathbf{z}_{n-1}$ and projecting it forward one step using the transition probability matrix $\mathbf{A}$. This predicted mean would give a predicted observation for $\mathbf{x}_n$ given by $\mathbf{C}\mathbf{A}\boldsymbol{\mu}_{n-1}$ obtained by applying the emission probability matrix $\mathbf{C}$ to the predicted hidden state mean. We can view the update equation (13.89) for the mean of the hidden variable distribution as taking the predicted mean $\mathbf{A}\boldsymbol{\mu}_{n-1}$ and then adding a correction that is proportional to the error $\mathbf{x}_n - \mathbf{C}\mathbf{A}\boldsymbol{\mu}_{n-1}$ between the predicted observation and the actual observation. The coefficient of this correction is given by the Kalman gain matrix. Thus we can view the Kalman filter as a process of making successive predictions and then correcting these predictions in the light of the new observations. This is illustrated graphically in Figure 13.21.

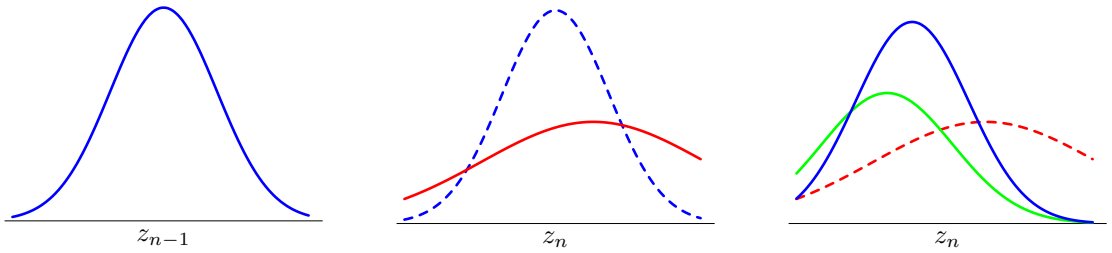


Figure 13.21 The linear dynamical system can be viewed as a sequence of steps in which increasing uncertainty in the state variable due to diffusion is compensated by the arrival of new data. In the left-hand plot, the blue curve shows the distribution $p(\mathbf{z}_{n-1}|\mathbf{x}_1, \ldots, \mathbf{x}_{n-1})$, which incorporates all the data up to step $n-1$. The diffusion arising from the nonzero variance of the transition probability $p(\mathbf{z}_n|\mathbf{z}_{n-1})$ gives the distribution $p(\mathbf{z}_n|\mathbf{x}_1, \ldots, \mathbf{x}_{n-1})$, shown in red in the centre plot. Note that this is broader and shifted relative to the blue curve (which is shown dashed in the centre plot for comparison). The next data observation $\mathbf{x}_n$ contributes through the emission density $p(\mathbf{x}_n|\mathbf{z}_n)$, which is shown as a function of $\mathbf{z}_n$ in green on the right-hand plot. Note that this is not a density with respect to $\mathbf{z}_n$ and so is not normalized to one. Inclusion of this new data point leads to a revised distribution $p(\mathbf{z}_n|\mathbf{x}_1, \ldots, \mathbf{x}_n)$ for the state density shown in blue. We see that observation of the data has shifted and narrowed the distribution compared to $p(\mathbf{z}_n|\mathbf{x}_1, \ldots, \mathbf{x}_{n-1})$ (which is shown in dashed in the right-hand plot for comparison).

2n 随时间的变化程度 vs noise 的大小，对应了13.3节开头引入LDS时介绍的背景与动机

If we consider a situation in which the measurement noise is small compared to the rate at which the latent variable is evolving, then we find that the posterior distribution for $\mathbf{z}_n$ depends only on the current measurement $\mathbf{x}_n$, in accordance with the intuition from our simple example at the start of the section. Similarly, if the latent variable is evolving slowly relative to the observation noise level, we find that the posterior mean for $\mathbf{z}_n$ is obtained by averaging all of the measurements obtained up to that time.

Exercise 13.27 分别给出证明

Exercise 13.28

应用举例 One of the most important applications of the Kalman filter is to tracking, and this is illustrated using a simple example of an object moving in two dimensions in Figure 13.22.

核心是下

计算 γ So far, we have solved the inference problem of finding the posterior marginal for a node $\mathbf{z}_n$ given observations from $\mathbf{x}_1$ up to $\mathbf{x}_n$. Next we turn to the problem of finding the marginal for a node $\mathbf{z}_n$ given all observations $\mathbf{x}_1$ to $\mathbf{x}_N$. For (temporal) 暂时的 时序的 data, this corresponds to the inclusion of future as well as past observations. Although this cannot be used for real-time prediction, it plays a key role in learning the parameters of the model. By analogy with the hidden Markov model, this problem can be solved by propagating messages from node $\mathbf{x}_N$ back to node $\mathbf{x}_1$ and combining this information with that obtained during the forward message passing stage used to compute the $\widehat{\alpha}(\mathbf{z}_n)$.

也就是用前面一段提到的 α-β 算法，流程流程是和HMM用的 α-β 算法计算的是一样的，这里选择计算 γ 而不是 β，本质上是相同的。

In the LDS literature, it is usual to formulate this backward recursion in terms of $\gamma(\mathbf{z}_n) = \widehat{\alpha}(\mathbf{z}_n)\widehat{\beta}(\mathbf{z}_n)$ rather than in terms of $\widehat{\beta}(\mathbf{z}_n)$. Because $\gamma(\mathbf{z}_n)$ must also be Gaussian, we write it in the form

$$\gamma(\mathbf{z}_n) = \widehat{\alpha}(\mathbf{z}_n)\widehat{\beta}(\mathbf{z}_n) = \mathcal{N}(\mathbf{z}_n|\widehat{\boldsymbol{\mu}}_n, \widehat{\mathbf{V}}_n). \tag{13.98}$$

To derive the required recursion, we start from the backward recursion (13.62) for

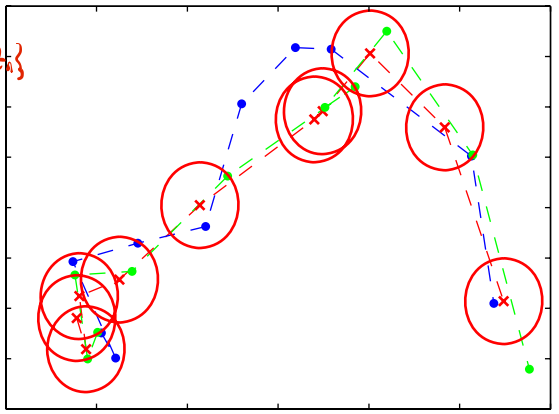


Figure 13.22 An illustration of a linear dynamical system being used to track a moving object. The blue points indicate the true positions of the object in a two-dimensional space at successive time steps, the green points denote noisy measurements of the positions, and the red crosses indicate the means of the inferred posterior distributions of the positions obtained by running the Kalman filtering equations. The covariances of the inferred positions are indicated by the red ellipses, which correspond to contours having one standard deviation.

$\widehat{\beta}(\mathbf{z}_n)$, which, for continuous latent variables, can be written in the form

$$c_{n+1}\widehat{\beta}(\mathbf{z}_n) = \int \widehat{\beta}(\mathbf{z}_{n+1}) p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1}) p(\mathbf{z}_{n+1}|\mathbf{z}_n)\, \mathrm{d}\mathbf{z}_{n+1}. \tag{13.99}$$

We now multiply both sides of (13.99) by $\widehat{\alpha}(\mathbf{z}_n)$ and substitute for $p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1})$ and $p(\mathbf{z}_{n+1}|\mathbf{z}_n)$ using (13.75) and (13.76). Then we make use of (13.89), (13.90) and (13.91), together with (13.98), and after some manipulation we obtain

Exercise 13.29

$$\widehat{\boldsymbol{\mu}}_n = \boldsymbol{\mu}_n + \mathbf{J}_n\left(\widehat{\boldsymbol{\mu}}_{n+1} - \mathbf{A}\boldsymbol{\mu}_n\right) \tag{13.100}$$

$$\widehat{\mathbf{V}}_n = \mathbf{V}_n + \mathbf{J}_n\left(\widehat{\mathbf{V}}_{n+1} - \mathbf{P}_n\right)\mathbf{J}_n^{\mathrm{T}} \tag{13.101}$$

where we have defined

$$\mathbf{J}_n = \mathbf{V}_n\mathbf{A}^{\mathrm{T}}\left(\mathbf{P}_n\right)^{-1} \tag{13.102}$$

and we have made use of $\mathbf{A}\mathbf{V}_n = \mathbf{P}_n\mathbf{J}_n^{\mathrm{T}}$. Note that these recursions require that the forward pass be completed first so that the quantities $\boldsymbol{\mu}_n$ and $\mathbf{V}_n$ will be available for the backward pass.

For the EM algorithm, we also require the pairwise posterior marginals, which can be obtained from (13.65) in the form

$$\begin{aligned}\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) &= (c_n)^{-1}\widehat{\alpha}(\mathbf{z}_{n-1}) p(\mathbf{x}_n|\mathbf{z}_n) p(\mathbf{z}_n|\mathbf{z}_{n-1})\widehat{\beta}(\mathbf{z}_n) \\ &= \frac{\mathcal{N}(\mathbf{z}_{n-1}|\boldsymbol{\mu}_{n-1}, \mathbf{V}_{n-1})\mathcal{N}(\mathbf{z}_n|\mathbf{A}\mathbf{z}_{n-1}, \boldsymbol{\Gamma})\mathcal{N}(\mathbf{x}_n|\mathbf{C}\mathbf{z}_n, \boldsymbol{\Sigma})\mathcal{N}(\mathbf{z}_n|\widehat{\boldsymbol{\mu}}_n, \widehat{\mathbf{V}}_n)}{c_n\widehat{\alpha}(\mathbf{z}_n)}.\end{aligned} \tag{13.103}$$

Substituting for $\widehat{\alpha}(\mathbf{z}_n)$ using (13.84) and rearranging, we see that $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$ is a Gaussian with mean given ~~with components $\gamma(\mathbf{z}_{n-1})$ and $\gamma(\mathbf{z}_n)$,~~ and a covariance between $\mathbf{z}_n$ and $\mathbf{z}_{n-1}$ given by

Exercise 13.31

$$\mathrm{cov}[\mathbf{z}_{n-1}, \mathbf{z}_n] = \mathbf{J}_{n-1}\widehat{\mathbf{V}}_n. \tag{13.104}$$

13.3.2 Learning in LDS

So far, we have considered the inference problem for linear dynamical systems, assuming that the model parameters $\boldsymbol{\theta} = \{\mathbf{A}, \boldsymbol{\Gamma}, \mathbf{C}, \boldsymbol{\Sigma}, \boldsymbol{\mu}_0, \mathbf{V}_0\}$ are known. Next, we consider the determination of these parameters using maximum likelihood (Ghahramani and Hinton, 1996b). Because the model has latent variables, this can be addressed using the EM algorithm, which was discussed in general terms in Chapter 9.

LDS的EM算法 [

We can derive the EM algorithm for the linear dynamical system as follows. Let us denote the estimated parameter values at some particular cycle of the algorithm by $\boldsymbol{\theta}^{\text{old}}$. For these parameter values, we can run the inference algorithm to determine the posterior distribution of the latent variables $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$, or more precisely those local posterior marginals that are required in the M step. In particular, we shall require the following expectations

$$\mathbb{E}[\mathbf{z}_n] = \widehat{\boldsymbol{\mu}}_n \tag{13.105}$$

$$\mathbb{E}\left[\mathbf{z}_n \mathbf{z}_{n-1}^{\mathrm{T}}\right] = \mathbf{J}_{n-1} \widehat{\mathbf{V}}_n + \widehat{\boldsymbol{\mu}}_n \widehat{\boldsymbol{\mu}}_{n-1}^{\mathrm{T}} \tag{13.106}$$

$$\mathbb{E}\left[\mathbf{z}_n \mathbf{z}_n^{\mathrm{T}}\right] = \widehat{\mathbf{V}}_n + \widehat{\boldsymbol{\mu}}_n \widehat{\boldsymbol{\mu}}_n^{\mathrm{T}} \tag{13.107}$$

where we have used (13.104).

Now we consider the complete-data log likelihood function, which is obtained by taking the logarithm of (13.6) and is therefore given by

$$\begin{aligned} \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) = {} & \ln p(\mathbf{z}_1|\boldsymbol{\mu}_0, \mathbf{V}_0) + \sum_{n=2}^{N} \ln p(\mathbf{z}_n|\mathbf{z}_{n-1}, \mathbf{A}, \boldsymbol{\Gamma}) \\ & + \sum_{n=1}^{N} \ln p(\mathbf{x}_n|\mathbf{z}_n, \mathbf{C}, \boldsymbol{\Sigma}) \end{aligned} \tag{13.108}$$

in which we have made the dependence on the parameters explicit. We now take the expectation of the complete-data log likelihood with respect to the posterior distribution $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$ which defines the function

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \mathbb{E}_{\mathbf{Z}|\boldsymbol{\theta}^{\text{old}}}[\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})]. \tag{13.109}$$

上面讨论了E步，下面开始讨论M步，并依次给出 $\boldsymbol{\mu}_0, \mathbf{P}_0, \mathbf{A}, \boldsymbol{\Gamma}, \mathbf{C}, \boldsymbol{\Sigma}$ 的迭代优化公式

In the M step, this function is maximized with respect to the components of $\boldsymbol{\theta}$.

// Consider first the parameters $\boldsymbol{\mu}_0$ and $\mathbf{V}_0$. If we substitute for $p(\mathbf{z}_1|\boldsymbol{\mu}_0, \mathbf{V}_0)$ in (13.108) using (13.77), and then take the expectation with respect to $\mathbf{Z}$, we obtain

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = -\frac{1}{2} \ln |\mathbf{V}_0| - \mathbb{E}_{\mathbf{Z}|\boldsymbol{\theta}^{\text{old}}}\left[\frac{1}{2}(\mathbf{z}_1 - \boldsymbol{\mu}_0)^{\mathrm{T}} \mathbf{V}_0^{-1} (\mathbf{z}_1 - \boldsymbol{\mu}_0)\right] + \text{const}$$

where all terms not dependent on $\boldsymbol{\mu}_0$ or $\mathbf{V}_0$ have been absorbed into the additive constant. Maximization with respect to $\boldsymbol{\mu}_0$ and $\mathbf{V}_0$ is easily performed by making use of the maximum likelihood solution for a Gaussian distribution discussed in Section 2.3.4, giving

Exercise 13.32

$$\boldsymbol{\mu}_0^{\text{new}} = \mathbb{E}[\mathbf{z}_1] \tag{13.110}$$
$$\mathbf{V}_0^{\text{new}} = \mathbb{E}[\mathbf{z}_1\mathbf{z}_1^{\mathrm{T}}] - \mathbb{E}[\mathbf{z}_1]\mathbb{E}[\mathbf{z}_1^{\mathrm{T}}]. \tag{13.111}$$

Similarly, to optimize $\mathbf{A}$ and $\boldsymbol{\Gamma}$, we substitute for $p(\mathbf{z}_n|\mathbf{z}_{n-1}, \mathbf{A}, \boldsymbol{\Gamma})$ in (13.108) using (13.75) giving

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = -\frac{N-1}{2}\ln|\boldsymbol{\Gamma}| - \mathbb{E}_{\mathbf{Z}|\boldsymbol{\theta}^{\text{old}}}\left[\frac{1}{2}\sum_{n=2}^{N}(\mathbf{z}_n - \mathbf{A}\mathbf{z}_{n-1})^{\mathrm{T}}\boldsymbol{\Gamma}^{-1}(\mathbf{z}_n - \mathbf{A}\mathbf{z}_{n-1})\right] + \text{const} \tag{13.112}$$

in which the constant comprises terms that are independent of $\mathbf{A}$ and $\boldsymbol{\Gamma}$. Maximizing with respect to these parameters then gives

Exercise 13.33

$$\mathbf{A}^{\text{new}} = \left(\sum_{n=2}^{N}\mathbb{E}\left[\mathbf{z}_n\mathbf{z}_{n-1}^{\mathrm{T}}\right]\right)\left(\sum_{n=2}^{N}\mathbb{E}\left[\mathbf{z}_{n-1}\mathbf{z}_{n-1}^{\mathrm{T}}\right]\right)^{-1} \tag{13.113}$$

$$\boldsymbol{\Gamma}^{\text{new}} = \frac{1}{N-1}\sum_{n=2}^{N}\left\{\mathbb{E}\left[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}\right] - \mathbf{A}^{\text{new}}\mathbb{E}\left[\mathbf{z}_{n-1}\mathbf{z}_n^{\mathrm{T}}\right] - \mathbb{E}\left[\mathbf{z}_n\mathbf{z}_{n-1}^{\mathrm{T}}\right](\mathbf{A}^{\text{new}})^{\mathrm{T}} + \mathbf{A}^{\text{new}}\mathbb{E}\left[\mathbf{z}_{n-1}\mathbf{z}_{n-1}^{\mathrm{T}}\right](\mathbf{A}^{\text{new}})^{\mathrm{T}}\right\}. \tag{13.114}$$

Note that $\mathbf{A}^{\text{new}}$ must be evaluated first, and the result can then be used to determine $\boldsymbol{\Gamma}^{\text{new}}$.

Finally, in order to determine the new values of $\mathbf{C}$ and $\boldsymbol{\Sigma}$, we substitute for $p(\mathbf{x}_n|\mathbf{z}_n, \mathbf{C}, \boldsymbol{\Sigma})$ in (13.108) using (13.76) giving

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = -\frac{N}{2}\ln|\boldsymbol{\Sigma}| - \mathbb{E}_{\mathbf{Z}|\boldsymbol{\theta}^{\text{old}}}\left[\frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n - \mathbf{C}\mathbf{z}_n)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \mathbf{C}\mathbf{z}_n)\right] + \text{const}.$$

Exercise 13.34

Maximizing with respect to $\mathbf{C}$ and $\boldsymbol{\Sigma}$ then gives

$$\mathbf{C}^{\text{new}} = \left(\sum_{n=1}^{N}\mathbf{x}_n\mathbb{E}\left[\mathbf{z}_n^{\mathrm{T}}\right]\right)\left(\sum_{n=1}^{N}\mathbb{E}\left[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}\right]\right)^{-1} \tag{13.115}$$

$$\boldsymbol{\Sigma}^{\text{new}} = \frac{1}{N}\sum_{n=1}^{N}\left\{\mathbf{x}_n\mathbf{x}_n^{\mathrm{T}} - \mathbf{C}^{\text{new}}\mathbb{E}\left[\mathbf{z}_n\right]\mathbf{x}_n^{\mathrm{T}} - \mathbf{x}_n\mathbb{E}\left[\mathbf{z}_n^{\mathrm{T}}\right](\mathbf{C}^{\text{new}})^{\mathrm{T}} + \mathbf{C}^{\text{new}}\mathbb{E}\left[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}\right](\mathbf{C}^{\text{new}})^{\mathrm{T}}\right\}. \tag{13.116}$$

We have approached parameter learning in the linear dynamical system using maximum likelihood. Inclusion of priors to give a MAP estimate is straightforward, and a fully Bayesian treatment can be found by applying the analytical approximation techniques discussed in Chapter 10, though a detailed treatment is precluded here due to lack of space.

13.3.3 Extensions of LDS

As with the hidden Markov model, there is considerable interest in extending the basic linear dynamical system in order to increase its capabilities. Although the assumption of a linear-Gaussian model leads to efficient algorithms for inference and learning, it also implies that the marginal distribution of the observed variables is simply a Gaussian, which represents a significant limitation. One simple extension of the linear dynamical system is to use a Gaussian mixture as the initial distribution for $\mathbf{z}_1$. If this mixture has K components, then the forward recursion equations (13.85) will lead to a mixture of K Gaussians over each hidden variable $\mathbf{z}_n$, and so the model is again tractable.

For many applications, the Gaussian emission density is a poor approximation. If instead we try to use a mixture of K Gaussians as the emission density, then the posterior $\widehat{\alpha}(\mathbf{z}_1)$ will also be a mixture of K Gaussians. However, from (13.85) the posterior $\widehat{\alpha}(\mathbf{z}_2)$ will comprise a mixture of K^2 Gaussians, and so on, with $\widehat{\alpha}(\mathbf{z}_n)$ being given by a mixture of K^n Gaussians. Thus the number of components grows exponentially with the length of the chain, and so this model is impractical.

More generally, introducing transition or emission models that depart from the linear-Gaussian (or other exponential family) model leads to an intractable inference problem. We can make deterministic approximations such as assumed density filtering or expectation propagation, or we can make use of sampling methods, as discussed in Section 13.3.4. One widely used approach is to make a Gaussian approximation by linearizing around the mean of the predicted distribution, which gives rise to the *extended Kalman filter* (Zarchan and Musoff, 2005).

As with hidden Markov models, we can develop interesting extensions of the basic linear dynamical system by expanding its graphical representation. For example, the *switching state space model* (Ghahramani and Hinton, 1998) can be viewed as a combination of the hidden Markov model with a set of linear dynamical systems. The model has multiple Markov chains of continuous linear-Gaussian latent variables, each of which is analogous to the latent chain of the linear dynamical system discussed earlier, together with a Markov chain of discrete variables of the form used in a hidden Markov model. The output at each time step is determined by stochastically choosing one of the continuous latent chains, using the state of the discrete latent variable as a switch, and then emitting an observation from the corresponding conditional output distribution. Exact inference in this model is intractable, but variational methods lead to an efficient inference scheme involving forward-backward recursions along each of the continuous and discrete Markov chains independently. Note that, if we consider multiple chains of discrete latent variables, and use one as the switch to select from the remainder, we obtain an analogous model having only discrete latent variables known as the *switching hidden Markov model*.

Chapter 10

13.3.4 Particle filters

粒子滤波的背景

Chapter 11

非高斯，基于样本进行推断

For dynamical systems which ~~do not have a~~ are not linear-Gaussian, for example, if they use a non-Gaussian emission density, we can turn to sampling methods in order to find a tractable inference algorithm. In particular, we can apply the sampling-importance-resampling formalism of Section 11.1.5 to obtain a sequential Monte Carlo algorithm known as the particle filter.

Consider the class of distributions represented by the graphical model in Figure 13.5, and suppose we are given the observed values $\mathbf{X}_n = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and we wish to draw L samples from the posterior distribution $p(\mathbf{z}_n|\mathbf{X}_n)$. Using Bayes' theorem, we have

$$\begin{aligned}
\mathbb{E}[f(\mathbf{z}_n)] &= \int f(\mathbf{z}_n) p(\mathbf{z}_n|\mathbf{X}_n)\, \mathrm{d}\mathbf{z}_n \\
&= \int f(\mathbf{z}_n) p(\mathbf{z}_n|\mathbf{x}_n, \mathbf{X}_{n-1})\, \mathrm{d}\mathbf{z}_n \qquad p(x_n|z_n, X_{n-1})\, p(z_n|X_{n-1}) \\
&= \frac{\int f(\mathbf{z}_n) p(\mathbf{x}_n|\mathbf{z}_n) p(\mathbf{z}_n|\mathbf{X}_{n-1})\, \mathrm{d}\mathbf{z}_n}{\int p(\mathbf{x}_n|\mathbf{z}_n) p(\mathbf{z}_n|\mathbf{X}_{n-1})\, \mathrm{d}\mathbf{z}_n} \qquad p(x_n, z_n|X_{n-1}) \qquad p(x_n|X_{n-1}) \\
&\simeq \sum_{l=1}^{L} w_n^{(l)} f(\mathbf{z}_n^{(l)}) \qquad (13.117)
\end{aligned}$$

where $\{\mathbf{z}_n^{(l)}\}$ is a set of samples drawn from $p(\mathbf{z}_n|\mathbf{X}_{n-1})$ and we have made use of the conditional independence property $p(\mathbf{x}_n|\mathbf{z}_n, \mathbf{X}_{n-1}) = p(\mathbf{x}_n|\mathbf{z}_n)$, which follows from the graph in Figure 13.5. The sampling weights $\{w_n^{(l)}\}$ are defined by

$$w_n^{(l)} = \frac{p(\mathbf{x}_n|\mathbf{z}_n^{(l)})}{\sum_{m=1}^{L} p(\mathbf{x}_n|\mathbf{z}_n^{(m)})} \qquad (13.118)$$

where the same samples are used in the numerator as in the denominator. Thus the posterior distribution $p(\mathbf{z}_n|\mathbf{X}_n)$ is represented by the set of samples $\{\mathbf{z}_n^{(l)}\}$ together with the corresponding weights $\{w_n^{(l)}\}$. Note that these weights satisfy $0 \leqslant w_n^{(l)} \leqslant 1$ and $\sum_l w_n^{(l)} = 1$.

承上启下

Because we wish to find a sequential sampling scheme, we shall suppose that a set of samples and weights have been obtained at time step n, and that we have subsequently observed the value of $\mathbf{x}_{n+1}$, and we wish to find the weights and samples at time step $n+1$. We first sample from the distribution $p(\mathbf{z}_{n+1}|\mathbf{X}_n)$. This is

straightforward since, again using Bayes' theorem

$$\begin{aligned} p(\mathbf{z}_{n+1}|\mathbf{X}_n) &= \int p(\mathbf{z}_{n+1}|\mathbf{z}_n, \mathbf{X}_n) p(\mathbf{z}_n|\mathbf{X}_n)\, \mathrm{d}\mathbf{z}_n \\ &= \int p(\mathbf{z}_{n+1}|\mathbf{z}_n) p(\mathbf{z}_n|\mathbf{X}_n)\, \mathrm{d}\mathbf{z}_n \\ &= \int p(\mathbf{z}_{n+1}|\mathbf{z}_n) p(\mathbf{z}_n|\mathbf{x}_n, \mathbf{X}_{n-1})\, \mathrm{d}\mathbf{z}_n \\ &= \frac{\int p(\mathbf{z}_{n+1}|\mathbf{z}_n) p(\mathbf{x}_n|\mathbf{z}_n) p(\mathbf{z}_n|\mathbf{X}_{n-1})\, \mathrm{d}\mathbf{z}_n}{\int p(\mathbf{x}_n|\mathbf{z}_n) p(\mathbf{z}_n|\mathbf{X}_{n-1})\, \mathrm{d}\mathbf{z}_n} \\ &\simeq \sum_l w_n^{(l)} p(\mathbf{z}_{n+1}|\mathbf{z}_n^{(l)}) \end{aligned} \tag{13.119}$$

where we have made use of the conditional independence properties

$$p(\mathbf{z}_{n+1}|\mathbf{z}_n, \mathbf{X}_n) = p(\mathbf{z}_{n+1}|\mathbf{z}_n) \tag{13.120}$$

$$p(\mathbf{x}_n|\mathbf{z}_n, \mathbf{X}_{n-1}) = p(\mathbf{x}_n|\mathbf{z}_n) \tag{13.121}$$

which follow from the application of the d-separation criterion to the graph in Figure 13.5. The distribution given by (13.119) is a mixture distribution, and samples can be drawn by choosing a component l with probability given by the mixing coefficients $w^{(l)}$ and then drawing a sample from the corresponding component.

In summary, we can view each step of the particle filter algorithm as comprising two stages. At time step n, we have a sample representation of the posterior distribution $p(\mathbf{z}_n|\mathbf{X}_n)$ expressed as samples $\{\mathbf{z}_n^{(l)}\}$ with corresponding weights $\{w_n^{(l)}\}$. This can be viewed as a mixture representation of the form (13.119). To obtain the corresponding representation for the next time step, we first draw L samples from the mixture distribution (13.119), and then for each sample we use the new observation $\mathbf{x}_{n+1}$ to evaluate the corresponding weights $w_{n+1}^{(l)} \propto p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1}^{(l)})$. This is illustrated, for the case of a single variable z, in Figure 13.23.

The particle filtering, or sequential Monte Carlo, approach has appeared in the literature under various names including the *bootstrap filter* (Gordon *et al.*, 1993), *survival of the fittest* (Kanazawa *et al.*, 1995), and the *condensation* algorithm (Isard and Blake, 1998).

Exercises

13.1 (⋆) WWW Use the technique of d-separation, discussed in Section 8.2, to verify that the Markov model shown in Figure 13.3 having N nodes in total satisfies the conditional independence properties (13.3) for $n = 2, \ldots, N$. Similarly, show that a model described by the graph in Figure 13.4 in which there are N nodes in total

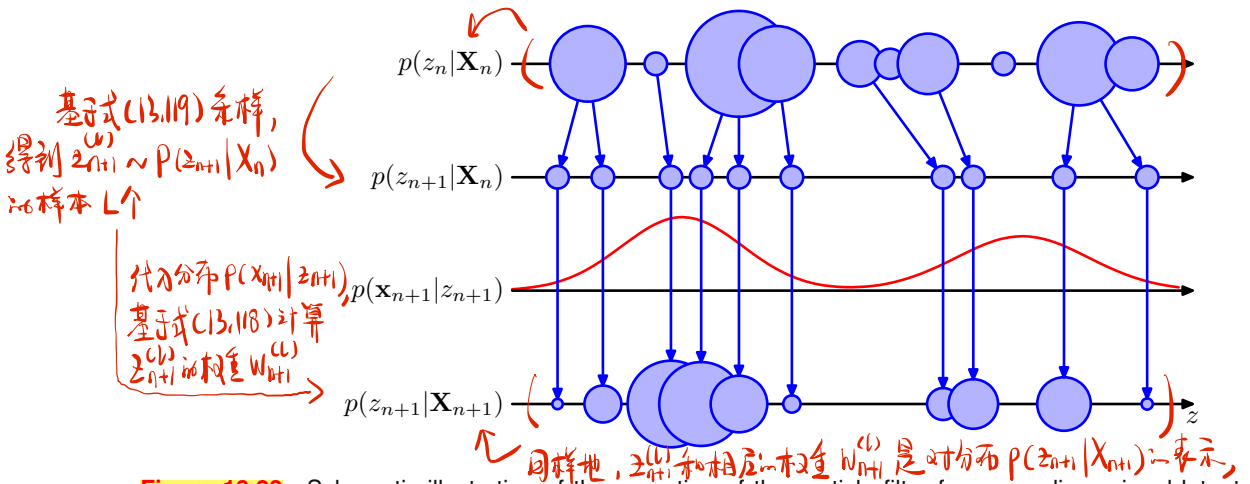


Figure 13.23 Schematic illustration of the operation of the particle filter for a one-dimensional latent space. At time step n, the posterior $p(z_n|\mathbf{x}_n)$ is represented as a mixture distribution, shown schematically as circles whose sizes are proportional to the weights $w_n^{(l)}$. A set of L samples is then drawn from this distribution and the new weights $w_{n+1}^{(l)}$ evaluated using $p(\mathbf{x}_{n+1}|z_{n+1}^{(l)})$.

satisfies the conditional independence properties

$$p(\mathbf{x}_n|\mathbf{x}_1,\ldots,\mathbf{x}_{n-1}) = p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{x}_{n-2}) \tag{13.122}$$

for $n = 3,\ldots,N$.

13.2 ($\star\star$) Consider the joint probability distribution (13.2) corresponding to the directed graph of Figure 13.3. Using the sum and product rules of probability, verify that this joint distribution satisfies the conditional independence property (13.3) for $n = 2,\ldots,N$. Similarly, show that the second-order Markov model described by the joint distribution (13.4) satisfies the conditional independence property

$$p(\mathbf{x}_n|\mathbf{x}_1,\ldots,\mathbf{x}_{n-1}) = p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{x}_{n-2}) \tag{13.123}$$

for $n = 3,\ldots,N$.

13.3 ($\star$) By using d-separation, show that the distribution $p(\mathbf{x}_1,\ldots,\mathbf{x}_N)$ of the observed data for the state space model represented by the directed graph in Figure 13.5 does not satisfy any conditional independence properties and hence does not exhibit the Markov property at any finite order.

13.4 ($\star\star$) **WWW** Consider a hidden Markov model in which the emission densities are represented by a parametric model $p(\mathbf{x}|\mathbf{z},\mathbf{w})$, such as a linear regression model or a neural network, in which $\mathbf{w}$ is a vector of adaptive parameters. Describe how the parameters $\mathbf{w}$ can be learned from data using maximum likelihood.

13.5 ($\star\star$) Verify the M-step equations (13.18) and (13.19) for the initial state probabilities and transition probability parameters of the hidden Markov model by maximization of the expected complete-data log likelihood function (13.17), using appropriate Lagrange multipliers to enforce the summation constraints on the components of $\boldsymbol{\pi}$ and $\mathbf{A}$.

13.6 ($\star$) Show that if any elements of the parameters $\boldsymbol{\pi}$ or $\mathbf{A}$ for a hidden Markov model are initially set to zero, then those elements will remain zero in all subsequent updates of the EM algorithm.

13.7 ($\star$) Consider a hidden Markov model with Gaussian emission densities. Show that maximization of the function $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ with respect to the mean and covariance parameters of the Gaussians gives rise to the M-step equations (13.20) and (13.21).

13.8 ($\star\star$) **www** For a hidden Markov model having discrete observations governed by a multinomial distribution, show that the conditional distribution of the observations given the hidden variables is given by (13.22) and the corresponding M step equations are given by (13.23). Write down the analogous equations for the conditional distribution and the M step equations for the case of a hidden Markov with multiple binary output variables each of which is governed by a Bernoulli conditional distribution. Hint: refer to Sections 2.1 and 2.2 for a discussion of the corresponding maximum likelihood solutions for i.i.d. data if required.

13.9 ($\star\star$) **www** Use the d-separation criterion to verify that the conditional independence properties (13.24)–(13.31) are satisfied by the joint distribution for the hidden Markov model defined by (13.6).

13.10 ($\star\star\star$) By applying the sum and product rules of probability, verify that the conditional independence properties (13.24)–(13.31) are satisfied by the joint distribution for the hidden Markov model defined by (13.6).

13.11 ($\star\star$) Starting from the expression (8.72) for the marginal distribution over the variables of a factor in a factor graph, together with the results for the messages in the sum-product algorithm obtained in Section 13.2.3, derive the result (13.43) for the joint posterior distribution over two successive latent variables in a hidden Markov model.

13.12 ($\star\star$) Suppose we wish to train a hidden Markov model by maximum likelihood using data that comprises R independent sequences of observations, which we denote by $\mathbf{X}^{(r)}$ where $r = 1, \ldots, R$. Show that in the E step of the EM algorithm, we simply evaluate posterior probabilities for the latent variables by running the α and β recursions independently for each of the sequences. Also show that in the M step, the initial probability and transition probability parameters are re-estimated

using modified forms of (13.18) and (13.19) given by

$$\pi_k = \frac{\sum_{r=1}^{R} \gamma(z_{1k}^{(r)})}{\sum_{r=1}^{R}\sum_{j=1}^{K} \gamma(z_{1j}^{(r)})} \tag{13.124}$$

$$A_{jk} = \frac{\sum_{r=1}^{R}\sum_{n=2}^{N} \xi(z_{n-1,j}^{(r)}, z_{n,k}^{(r)})}{\sum_{r=1}^{R}\sum_{l=1}^{K}\sum_{n=2}^{N} \xi(z_{n-1,j}^{(r)}, z_{n,l}^{(r)})} \tag{13.125}$$

where, for notational convenience, we have assumed that the sequences are of the same length (the generalization to sequences of different lengths is straightforward). Similarly, show that the M-step equation for re-estimation of the means of Gaussian emission models is given by

$$\boldsymbol{\mu}_k = \frac{\sum_{r=1}^{R}\sum_{n=1}^{N} \gamma(z_{nk}^{(r)})\mathbf{x}_n^{(r)}}{\sum_{r=1}^{R}\sum_{n=1}^{N} \gamma(z_{nk}^{(r)})}. \tag{13.126}$$

Note that the M-step equations for other emission model parameters and distributions take an analogous form.

13.13 (⋆⋆) **www** Use the definition (8.64) of the messages passed from a factor node to a variable node in a factor graph, together with the expression (13.6) for the joint distribution in a hidden Markov model, to show that the definition (13.50) of the alpha message is the same as the definition (13.34).

13.14 (⋆⋆) Use the definition (8.67) of the messages passed from a factor node to a variable node in a factor graph, together with the expression (13.6) for the joint distribution in a hidden Markov model, to show that the definition (13.52) of the beta message is the same as the definition (13.35).

13.15 (⋆⋆) Use the expressions (13.33) and (13.43) for the marginals in a hidden Markov model to derive the corresponding results (13.64) and (13.65) expressed in terms of re-scaled variables.

13.16 (⋆⋆⋆) In this exercise, we derive the forward message passing equation for the Viterbi algorithm directly from the expression (13.6) for the joint distribution. This involves maximizing over all of the hidden variables $\mathbf{z}_1, \ldots, \mathbf{z}_N$. By taking the logarithm and then exchanging maximizations and summations, derive the recursion

(13.68) where the quantities $\omega(\mathbf{z}_n)$ are defined by (13.70). Show that the initial condition for this recursion is given by (13.69).

13.17 ($\star$) **www** Show that the directed graph for the input-output hidden Markov model, given in Figure 13.18, can be expressed as a tree-structured factor graph of the form shown in Figure 13.15 and write down expressions for the initial factor $h(\mathbf{z}_1)$ and for the general factor $f_n(\mathbf{z}_{n-1}, \mathbf{z}_n)$ where $2 \leqslant n \leqslant N$.

13.18 ($\star\star\star$) Using the result of Exercise 13.17, derive the recursion equations, including the initial conditions, for the forward-backward algorithm for the input-output hidden Markov model shown in Figure 13.18.

13.19 ($\star$) **www** The Kalman filter and smoother equations allow the posterior distributions over individual latent variables, conditioned on all of the observed variables, to be found efficiently for linear dynamical systems. Show that the sequence of latent variable values obtained by maximizing each of these posterior distributions individually is the same as the most probable sequence of latent values. To do this, simply note that the joint distribution of all latent and observed variables in a linear dynamical system is Gaussian, and hence all conditionals and marginals will also be Gaussian, and then make use of the result (2.98).

13.20 ($\star\star$) **www** Use the result (2.115) to prove (13.87).

13.21 ($\star\star$) Use the results (2.115) and (2.116), together with the matrix identities (C.5) and (C.7), to derive the results (13.89), (13.90), and (13.91), where the Kalman gain matrix $\mathbf{K}_n$ is defined by (13.92).

13.22 ($\star\star$) **www** Using (13.93), together with the definitions (13.76) and (13.77) and the result (2.115), derive (13.96).

13.23 ($\star\star$) Using (13.93), together with the definitions (13.76) and (13.77) and the result (2.116), derive (13.94), (13.95) and (13.97).

13.24 ($\star\star$) **www** Consider a generalization of (13.75) and (13.76) in which we include constant terms $\mathbf{a}$ and $\mathbf{c}$ in the Gaussian means, so that

$$p(\mathbf{z}_n|\mathbf{z}_{n-1}) = \mathcal{N}(\mathbf{z}_n|\mathbf{A}\mathbf{z}_{n-1} + \mathbf{a}, \boldsymbol{\Gamma}) \tag{13.127}$$

$$p(\mathbf{x}_n|\mathbf{z}_n) = \mathcal{N}(\mathbf{x}_n|\mathbf{C}\mathbf{z}_n + \mathbf{c}, \boldsymbol{\Sigma}). \tag{13.128}$$

Show that this extension can be re-cast in the framework discussed in this chapter by defining a state vector $\mathbf{z}$ with an additional component fixed at unity, and then augmenting the matrices $\mathbf{A}$ and $\mathbf{C}$ using extra columns corresponding to the parameters $\mathbf{a}$ and $\mathbf{c}$.

13.25 ($\star\star$) In this exercise, we show that when the Kalman filter equations are applied to independent observations, they reduce to the results given in Section 2.3 for the maximum likelihood solution for a single Gaussian distribution. Consider the problem of finding the mean μ of a single Gaussian random variable x, in which we are given a set of independent observations $\{x_1, \ldots, x_N\}$. To model this we can use

a linear dynamical system governed by (13.75) and (13.76), with latent variables $\{z_1, \ldots, z_N\}$ in which ~~$\mathbf{C}$ becomes the identity matrix and where the transition probability $\mathbf{A} = \mathbf{0}$ because the observations are independent~~. Let the parameters $\boldsymbol{\mu}_0$ and $\mathbf{V}_0$ of the initial state be denoted by μ_0 and σ_0^2, respectively, and suppose that $\boldsymbol{\Sigma}$ becomes σ^2. Write down the corresponding Kalman filter equations starting from the general results (13.89) and (13.90), together with (13.94) and (13.95). Show that these are equivalent to the results (2.141) and (2.142) obtained directly by considering independent data.

13.26 ($\star\star\star$) Consider a special case of the linear dynamical system of Section 13.3 that is equivalent to probabilistic PCA, so that the transition matrix $\mathbf{A} = \mathbf{0}$, the covariance $\boldsymbol{\Gamma} = \mathbf{I}$, and the noise covariance $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$. By making use of the matrix inversion identity (C.7) show that, if the emission density matrix $\mathbf{C}$ is denoted $\mathbf{W}$, then the posterior distribution over the hidden states defined by (13.89) and (13.90) reduces to the result (12.42) for probabilistic PCA.

13.27 ($\star$) **www** Consider a linear dynamical system of the form discussed in Section 13.3 in which the amplitude of the observation noise goes to zero, so that $\boldsymbol{\Sigma} = \mathbf{0}$. Show that the posterior distribution for $\mathbf{z}_n$ has mean $\mathbf{x}_n$ and zero variance. This accords with our intuition that if there is no noise, we should just use the current observation $\mathbf{x}_n$ to estimate the state variable $\mathbf{z}_n$ and ignore all previous observations.

13.28 ($\star\star\star$) Consider a special case of the linear dynamical system of Section 13.3 in which the state variable $\mathbf{z}_n$ is constrained to be equal to the previous state variable, which corresponds to $\mathbf{A} = \mathbf{I}$ and $\boldsymbol{\Gamma} = \mathbf{0}$. For simplicity, assume also that $\mathbf{V}_0 \to \infty$ so that the initial conditions for $\mathbf{z}$ are unimportant, and the predictions are determined purely by the data. Use proof by induction to show that the posterior mean for state $\mathbf{z}_n$ is determined by the average of $\mathbf{x}_1, \ldots, \mathbf{x}_n$. This corresponds to the intuitive result that if the state variable is constant, our best estimate is obtained by averaging the observations.

13.29 ($\star\star\star$) Starting from the backwards recursion equation (13.99), derive the RTS smoothing equations (13.100) and (13.101) for the Gaussian linear dynamical system.

13.30 ($\star\star$) Starting from the result (13.65) for the pairwise posterior marginal in a state space model, derive the specific form (13.103) for the case of the Gaussian linear dynamical system.

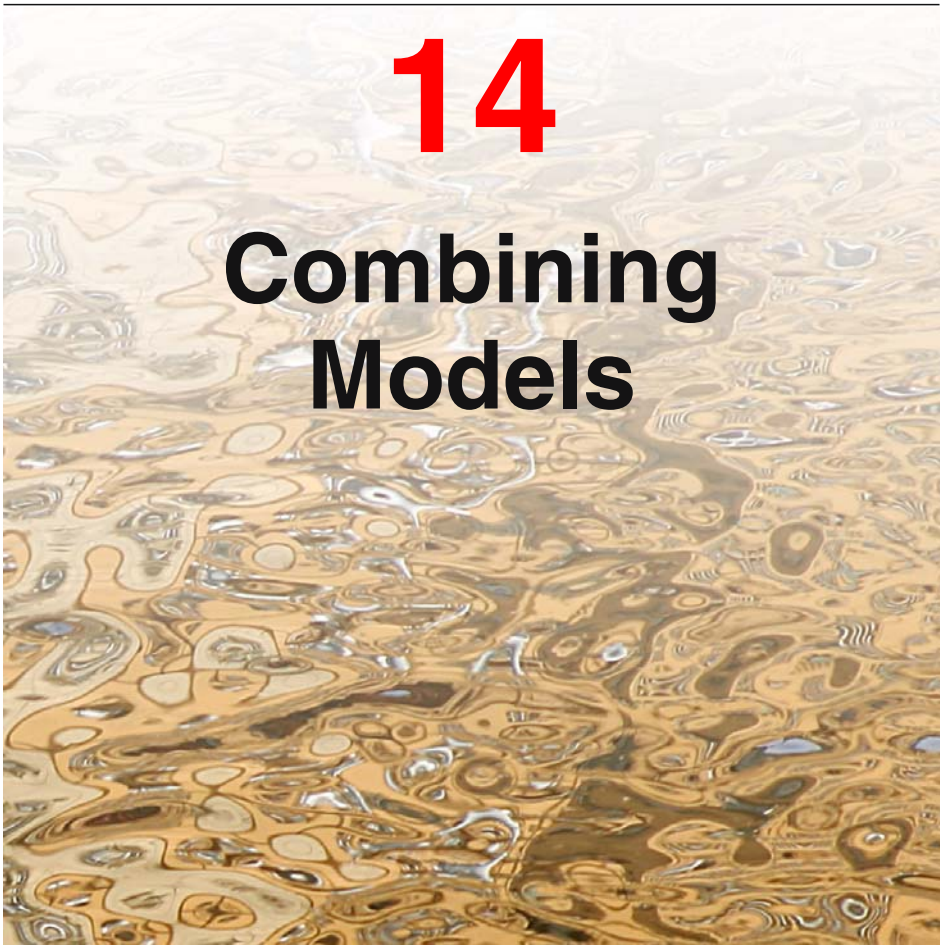
13.31 ($\star\star$) Starting from the result (13.103) and by substituting for $\widehat{\alpha}(\mathbf{z}_n)$ using (13.84), verify the result (13.104) for the covariance between $\mathbf{z}_n$ and $\mathbf{z}_{n-1}$.

13.32 ($\star\star$) **www** Verify the results (13.110) and (13.111) for the M-step equations for $\boldsymbol{\mu}_0$ and $\mathbf{V}_0$ in the linear dynamical system.

13.33 ($\star\star$) Verify the results (13.113) and (13.114) for the M-step equations for $\mathbf{A}$ and $\boldsymbol{\Gamma}$ in the linear dynamical system.

13.34 ($\star\star$) Verify the results (13.115) and (13.116) for the M-step equations for $\mathbf{C}$ and $\boldsymbol{\Sigma}$ in the linear dynamical system.

14 Combining Models



In earlier chapters, we have explored a range of different models for solving classification and regression problems. It is often found that improved performance can be obtained by combining multiple models together in some way, instead of just using a single model in isolation. For instance, we might train L different models and then make predictions using the average of the predictions made by each model. Such combinations of models are sometimes called *committees*. In Section 14.2, we discuss ways to apply the committee concept in practice, and we also give some insight into why it can sometimes be an effective procedure.

One important variant of the committee method, known as *boosting*, involves training multiple models in sequence in which the error function used to train a particular model depends on the performance of the previous models. This can produce substantial improvements in performance compared to the use of a single model and is discussed in Section 14.3.

Instead of averaging the predictions of a set of models, an alternative form of

model combination is to select one of the models to make the prediction, in which the choice of model is a function of the input variables. Thus different models become responsible for making predictions in different regions of input space. One widely used framework of this kind is known as a *decision tree* in which the selection process can be described as a sequence of binary selections corresponding to the traversal of a tree structure and is discussed in Section 14.4. In this case, the individual models are generally chosen to be very simple, and the overall flexibility of the model arises from the input-dependent selection process. Decision trees can be applied to both classification and regression problems.

One limitation of decision trees is that the division of input space is based on hard splits in which only one model is responsible for making predictions for any given value of the input variables. The decision process can be softened by moving to a probabilistic framework for combining models, as discussed in Section 14.5. For example, if we have a set of K models for a conditional distribution $p(t|\mathbf{x}, k)$ where $\mathbf{x}$ is the input variable, t is the target variable, and $k = 1, \ldots, K$ indexes the model, then we can form a probabilistic mixture of the form

$$p(t|\mathbf{x}) = \sum_{k=1}^{K} \pi_k(\mathbf{x}) p(t|\mathbf{x}, k) \tag{14.1}$$

in which $\pi_k(\mathbf{x}) = p(k|\mathbf{x})$ represent the input-dependent mixing coefficients. Such models can be viewed as mixture distributions in which the component densities, as well as the mixing coefficients, are conditioned on the input variables and are known as *mixtures of experts*. They are closely related to the mixture density network model discussed in Section 5.6.

14.1. Bayesian Model Averaging

It is important to distinguish between model combination methods and Bayesian model averaging, as the two are often confused. To understand the difference, consider the example of density estimation using a mixture of Gaussians in which several Gaussian components are combined probabilistically. The model contains a binary latent variable $\mathbf{z}$ that indicates which component of the mixture is responsible for generating the corresponding data point. Thus the model is specified in terms of a joint distribution

Section 9.2

$$p(\mathbf{x}, \mathbf{z}) \tag{14.2}$$

and the corresponding density over the observed variable $\mathbf{x}$ is obtained by marginalizing over the latent variable

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}). \tag{14.3}$$

In the case of our Gaussian mixture example, this leads to a distribution of the form

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{14.4}$$

with the usual interpretation of the symbols. This is an example of model combination. For independent, identically distributed data, we can use (14.3) to write the marginal probability of a data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ in the form

$$p(\mathbf{X}) = \prod_{n=1}^{N} p(\mathbf{x}_n) = \prod_{n=1}^{N} \left[\sum_{\mathbf{z}_n} p(\mathbf{x}_n, \mathbf{z}_n) \right]. \tag{14.5}$$

Thus we see that each observed data point $\mathbf{x}_n$ has a corresponding latent variable $\mathbf{z}_n$.

上面讲的混合模型，下面讲的贝叶斯模型平均

// Now suppose we have several different models indexed by $h = 1, \ldots, H$ with prior probabilities $p(h)$. For instance one model might be a mixture of Gaussians and another model might be a mixture of Cauchy distributions. The marginal distribution over the data set is given by

$$p(\mathbf{X}) = \sum_{h=1}^{H} p(\mathbf{X}|h)p(h). \tag{14.6}$$

This is an example of Bayesian model averaging. The interpretation of this summation over h is that just one model is responsible for generating the whole data set, and the probability distribution over h simply reflects our uncertainty as to which model that is. As the size of the data set increases, this uncertainty reduces, and the posterior probabilities $p(h|\mathbf{X})$ become increasingly focussed on just one of the models.

区别

// This highlights the key difference between Bayesian model averaging and model combination, because in Bayesian model averaging the whole data set is generated by a single model. By contrast, when we combine multiple models, as in (14.5), we see that different data points within the data set can potentially be generated from different values of the latent variable $\mathbf{z}$ and hence by different components.]

Although we have considered the marginal probability $p(\mathbf{X})$, the same considerations apply for the predictive density $p(\mathbf{x}|\mathbf{X})$ or for conditional distributions such as $p(\mathbf{t}|\mathbf{x}, \mathbf{X}, \mathbf{T})$.

Exercise 14.1

给出混合模型与贝叶斯模型平均的区别

14.2. Committees

The simplest way to construct a committee is to average the predictions of a set of individual models. Such a procedure can be motivated from a frequentist perspective by considering the trade-off between bias and variance, which decomposes the error due to a model into the bias component that arises from differences between the model and the true function to be predicted, and the variance component that represents the sensitivity of the model to the individual data points. Recall from Figure 3.5

Section 3.2

that when we trained multiple polynomials using the sinusoidal data, and then averaged the resulting functions, the contribution arising from the variance term tended to cancel, leading to improved predictions. When we averaged a set of low-bias models (corresponding to higher order polynomials), we obtained accurate predictions for the underlying sinusoidal function from which the data were generated.

In practice, of course, we have only a single data set, and so we have to find a way to introduce variability between the different models within the committee. One approach is to use *bootstrap* data sets, discussed in Section 1.2.3. Consider a regression problem in which we are trying to predict the value of a single continuous variable, and suppose we generate M bootstrap data sets and then use each to train a separate copy $y_m(\mathbf{x})$ of a predictive model where $m = 1, \ldots, M$. The committee prediction is given by

$$y_{\mathrm{COM}}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x}). \tag{14.7}$$

This procedure is known as bootstrap aggregation or *bagging* (Breiman, 1996).

通过证明式(14.14)给出了Bagging 的有效性，但因其建立在假设式(14.12)和(14.13)上，实际情况下很难满足，所以一种更有价值的Boosting 可视为一种合适的Bagging 和Boosting 都属于Committees 方法。

Suppose the true regression function that we are trying to predict is given by $h(\mathbf{x})$, so that the output of each of the models can be written as the true value plus an error in the form

$$y_m(\mathbf{x}) = h(\mathbf{x}) + \epsilon_m(\mathbf{x}). \tag{14.8}$$

The average sum-of-squares error then takes the form

$$\mathbb{E}_{\mathbf{x}}\left[\{y_m(\mathbf{x}) - h(\mathbf{x})\}^2\right] = \mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})^2\right] \tag{14.9}$$

where $\mathbb{E}_{\mathbf{x}}[\cdot]$ denotes a frequentist expectation with respect to the distribution of the input vector $\mathbf{x}$. The average error made by the models acting individually is therefore

$$E_{\mathrm{AV}} = \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})^2\right]. \tag{14.10}$$

Similarly, the expected error from the committee (14.7) is given by

$$\begin{aligned} E_{\mathrm{COM}} &= \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x}) - h(\mathbf{x})\right\}^2\right] \\ &= \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} \epsilon_m(\mathbf{x})\right\}^2\right] \end{aligned} \tag{14.11}$$

If we assume that the errors have zero mean and are uncorrelated, so that

$$\mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})\right] = 0 \tag{14.12}$$

$$\mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})\epsilon_l(\mathbf{x})\right] = 0, \qquad m \neq l \tag{14.13}$$

Exercise 14.2

then we obtain

$$E_{\mathrm{COM}} = \frac{1}{M} E_{\mathrm{AV}}. \tag{14.14}$$

This apparently dramatic result suggests that the average error of a model can be reduced by a factor of M simply by averaging M versions of the model. Unfortunately, it depends on the key assumption that the errors due to the individual models are uncorrelated. In practice, the errors are typically highly correlated, and the reduction in overall error is generally small. It can, however, be shown that the expected committee error will not exceed the expected error of the constituent models, so that $E_{\mathrm{COM}} \leqslant E_{\mathrm{AV}}$. In order to achieve more significant improvements, we turn to a more sophisticated technique for building committees, known as boosting.

Exercise 14.3

14.3. Boosting

Boosting is a powerful technique for combining multiple 'base' classifiers to produce a form of committee whose performance can be significantly better than that of any of the base classifiers. Here we describe the most widely used form of boosting algorithm called *AdaBoost*, short for 'adaptive boosting', developed by Freund and Schapire (1996). Boosting can give good results even if the base classifiers have a performance that is only slightly better than random, and hence sometimes the base classifiers are known as *weak learners*. Originally designed for solving classification problems, boosting can also be extended to regression (Friedman, 2001).

The principal difference between boosting and the committee methods such as bagging discussed above, is that the base classifiers are trained in sequence, and each base classifier is trained using a weighted form of the data set in which the weighting coefficient associated with each data point depends on the performance of the previous classifiers. In particular, points that are misclassified by one of the base classifiers are given greater weight when used to train the next classifier in the sequence. Once all the classifiers have been trained, their predictions are then combined through a weighted majority voting scheme, as illustrated schematically in Figure 14.1.

Consider a two-class classification problem, in which the training data comprises input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$ along with corresponding binary target variables $t_1, \ldots, t_N$ where $t_n \in \{-1, 1\}$. Each data point is given an associated weighting parameter w_n, which is initially set $1/N$ for all data points. We shall suppose that we have a procedure available for training a base classifier using weighted data to give a function $y(\mathbf{x}) \in \{-1, 1\}$. At each stage of the algorithm, AdaBoost trains a new classifier using a data set in which the weighting coefficients are adjusted according to the performance of the previously trained classifier so as to give greater weight to the misclassified data points. Finally, when the desired number of base classifiers have been trained, they are combined to form a committee using coefficients that give different weight to different base classifiers. The precise form of the AdaBoost algorithm is given below.

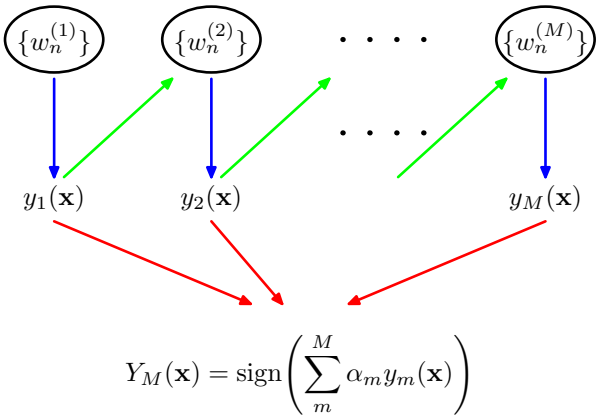


Figure 14.1 Schematic illustration of the boosting framework. Each base classifier $y_m(\mathbf{x})$ is trained on a weighted form of the training set (blue arrows) in which the weights $w_n^{(m)}$ depend on the performance of the previous base classifier $y_{m-1}(\mathbf{x})$ (green arrows). Once all base classifiers have been trained, they are combined to give the final classifier $Y_M(\mathbf{x})$ (red arrows).

AdaBoost

1. Initialize the data weighting coefficients $\{w_n\}$ by setting $w_n^{(1)} = 1/N$ for $n = 1, \ldots, N$.
2. For $m = 1, \ldots, M$:
   (a) Fit a classifier $y_m(\mathbf{x})$ to the training data by minimizing the weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) \tag{14.15}$$

   where $I(y_m(\mathbf{x}_n) \neq t_n)$ is the indicator function and equals 1 when $y_m(\mathbf{x}_n) \neq t_n$ and 0 otherwise.

   (b) Evaluate the quantities

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}} \tag{14.16}$$

   and then use these to evaluate

$$\alpha_m = \ln\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\}. \tag{14.17}$$

   (c) Update the data weighting coefficients

$$w_n^{(m+1)} = w_n^{(m)} \exp\left\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\right\} \tag{14.18}$$

3. Make predictions using the final model, which is given by

$$Y_M(\mathbf{x}) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x})\right). \tag{14.19}$$

We see that the first base classifier $y_1(\mathbf{x})$ is trained using weighting coefficients $w_n^{(1)}$ that are all equal, which therefore corresponds to the usual procedure for training a single classifier. From (14.18), we see that in subsequent iterations the weighting coefficients $w_n^{(m)}$ are increased for data points that are misclassified and ~~decreased~~ unchanged for data points that are correctly classified. Successive classifiers are therefore forced to place greater emphasis on points that have been misclassified by previous classifiers, and data points that continue to be misclassified by successive classifiers receive ever greater weight. The quantities ϵ_m represent weighted measures of the error rates of each of the base classifiers on the data set. We therefore see that the weighting coefficients α_m defined by (14.17) give greater weight to the more accurate classifiers when computing the overall output given by (14.19).

The AdaBoost algorithm is illustrated in Figure 14.2, using a subset of 30 data points taken from the toy classification data set shown in Figure A.7. Here each base learners consists of a threshold on one of the input variables. This simple classifier corresponds to a form of decision tree known as a 'decision stumps', i.e., a decision tree with a single node. Thus each base learner classifies an input according to whether one of the input features exceeds some threshold and therefore simply partitions the space into two regions separated by a linear decision surface that is ~~parallel~~ perpendicular to one of the axes.

Section 14.4 AdaBoost 算法对基学习器没有要求，Figure 14.2 的例子用的是只含一个结点的决策树桩，即决策桩

14.3.1 Minimizing exponential error

Boosting was originally motivated using statistical learning theory, leading to upper bounds on the generalization error. However, these bounds turn out to be too loose to have practical value, and the actual performance of boosting is much better than the bounds alone would suggest. Friedman *et al.* (2000) gave a different and very simple interpretation of boosting in terms of the sequential minimization of an exponential error function.

统计学习理论 指统计学习理论中得出的各种 bounds

本节就是从这种角度来推导 AdaBoost

Consider the exponential error function defined by

$$E = \sum_{n=1}^{N} \exp\{-t_n f_m(\mathbf{x}_n)\} \tag{14.20}$$

where $f_m(\mathbf{x})$ is a classifier defined in terms of a linear combination of base classifiers $y_l(\mathbf{x})$ of the form

$$f_m(\mathbf{x}) = \frac{1}{2}\sum_{l=1}^{m} \alpha_l y_l(\mathbf{x}) \tag{14.21}$$

and $t_n \in \{-1, 1\}$ are the training set target values. Our goal is to minimize E with respect to both the weighting coefficients α_l and the parameters of the base classifiers $y_l(\mathbf{x})$.

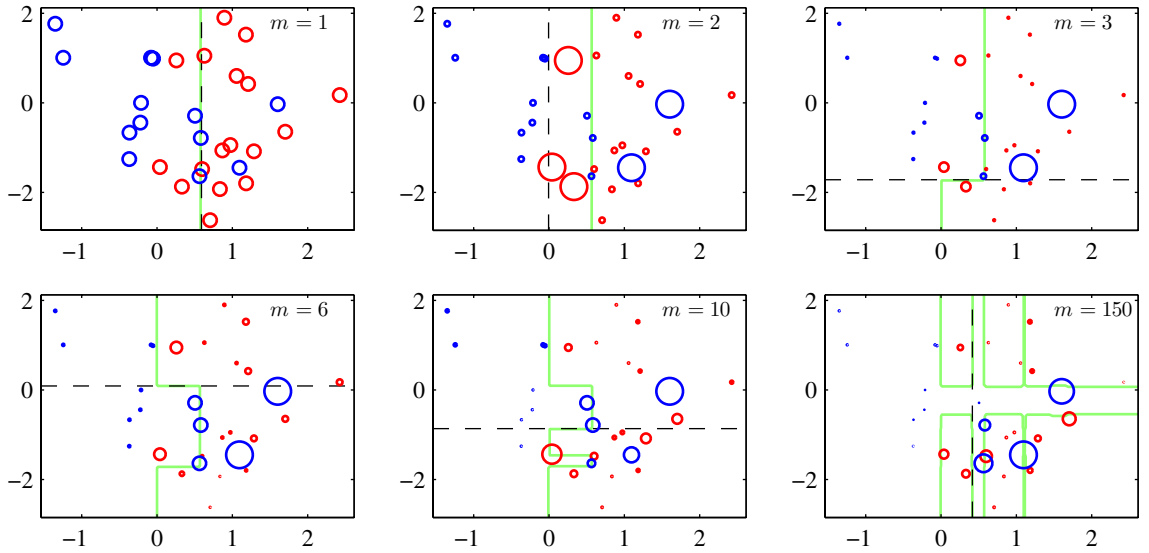


Figure 14.2 Illustration of boosting in which the base learners consist of simple thresholds applied to one or other of the axes. Each figure shows the number m of base learners trained so far, along with the decision boundary of the most recent base learner (dashed black line) and the combined decision boundary of the ensemble (solid green line). Each data point is depicted by a circle whose radius indicates the weight assigned to that data point when training the most recently added base learner. Thus, for instance, we see that points that are misclassified by the $m = 1$ base learner are given greater weight when training the $m = 2$ base learner.

Instead of doing a global error function minimization, however, we shall suppose that the base classifiers $y_1(\mathbf{x}), \ldots, y_{m-1}(\mathbf{x})$ are fixed, as are their coefficients $\alpha_1, \ldots, \alpha_{m-1}$, and so we are minimizing only with respect to α_m and $y_m(\mathbf{x})$. Separating off the contribution from base classifier $y_m(\mathbf{x})$, we can then write the error function in the form

$$\begin{aligned} E &= \sum_{n=1}^{N} \exp\left\{-t_n f_{m-1}(\mathbf{x}_n) - \frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right\} \\ &= \sum_{n=1}^{N} w_n^{(m)} \exp\left\{-\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right\} \end{aligned} \tag{14.22}$$

where the coefficients $w_n^{(m)} = \exp\{-t_n f_{m-1}(\mathbf{x}_n)\}$ can be viewed as constants because we are optimizing only α_m and $y_m(\mathbf{x})$. If we denote by $\mathcal{T}_m$ the set of data points that are correctly classified by $y_m(\mathbf{x})$, and if we denote the remaining misclassified points by $\mathcal{M}_m$, then we can in turn rewrite the error function in the

form

$$\begin{aligned} E &= e^{-\alpha_m/2} \sum_{n\in\mathcal{T}_m} w_n^{(m)} + e^{\alpha_m/2} \sum_{n\in\mathcal{M}_m} w_n^{(m)} \\ &= (e^{\alpha_m/2} - e^{-\alpha_m/2}) \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m/2} \sum_{n=1}^{N} w_n^{(m)}. \end{aligned} \tag{14.23}$$

When we minimize this with respect to $y_m(\mathbf{x})$, we see that the second term is constant, and so this is equivalent to minimizing (14.15) because the overall multiplicative factor in front of the summation does not affect the location of the minimum. Similarly, minimizing with respect to α_m, we obtain (14.17) in which ϵ_m is defined by (14.16).

Exercise 14.6

From (14.22) we see that, having found α_m and $y_m(\mathbf{x})$, the weights on the data points are updated using

$$w_n^{(m+1)} = w_n^{(m)} \exp\left\{-\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right\}. \tag{14.24}$$

Making use of the fact that

$$t_n y_m(\mathbf{x}_n) = 1 - 2I(y_m(\mathbf{x}_n) \neq t_n) \tag{14.25}$$

we see that the weights $w_n^{(m)}$ are updated at the next iteration using

$$w_n^{(m+1)} = w_n^{(m)} \exp(-\alpha_m/2) \exp\left\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\right\}. \tag{14.26}$$

Because the term $\exp(-\alpha_m/2)$ is independent of n, we see that it weights all data points by the same factor and so can be discarded. Thus we obtain (14.18).

Finally, once all the base classifiers are trained, new data points are classified by evaluating the sign of the combined function defined according to (14.21). Because the factor of $1/2$ does not affect the sign it can be omitted, giving (14.19).

14.3.2 Error functions for boosting

The exponential error function that is minimized by the AdaBoost algorithm differs from those considered in previous chapters. To gain some insight into the nature of the exponential error function, we first consider the expected error given by

$$\mathbb{E}_{\mathbf{x},t}\left[\exp\{-ty(\mathbf{x})\}\right] = \sum_t \int \exp\{-ty(\mathbf{x})\} p(t|\mathbf{x}) p(\mathbf{x})\, \mathrm{d}\mathbf{x}. \tag{14.27}$$

If we perform a variational minimization with respect to all possible functions $y(\mathbf{x})$, we obtain

Exercise 14.7

$$y(\mathbf{x}) = \frac{1}{2} \ln\left\{\frac{p(t=1|\mathbf{x})}{p(t=-1|\mathbf{x})}\right\} \tag{14.28}$$

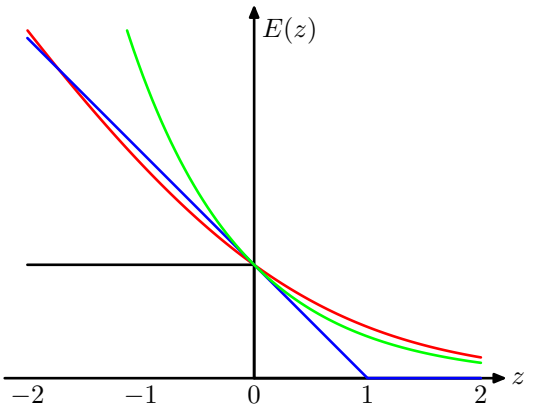


Figure 14.3 Plot of the exponential (green) and rescaled cross-entropy (red) error functions along with the hinge error (blue) used in support vector machines, and the misclassification error (black). Note that for large negative values of $z = ty(\mathbf{x})$, the cross-entropy gives a linearly increasing penalty, whereas the exponential loss gives an exponentially increasing penalty.

which is half the log-odds. Thus the AdaBoost algorithm is seeking the best approximation to the log odds ratio, within the space of functions represented by the linear combination of base classifiers, subject to the constrained minimization resulting from the sequential optimization strategy. This result motivates the use of the sign function in (14.19) to arrive at the final classification decision.

We have already seen that the minimizer $y(\mathbf{x})$ of the cross-entropy error (4.90) for two-class classification is given by the posterior class probability. In the case of a target variable $t \in \{-1, 1\}$, we have seen that the error function is given by $\ln(1 + \exp(-yt))$. This is compared with the exponential error function in Figure 14.3, where we have divided the cross-entropy error by a constant factor $\ln(2)$ so that it passes through the point $(0, 1)$ for ease of comparison. We see that both can be seen as continuous approximations to the ideal misclassification error function. An advantage of the exponential error is that its sequential minimization leads to the simple AdaBoost scheme. One drawback, however, is that it penalizes large negative values of $ty(\mathbf{x})$ much more strongly than cross-entropy. In particular, we see that for large negative values of ty, the cross-entropy grows linearly with $|ty|$, whereas the exponential error function grows exponentially with $|ty|$. Thus the exponential error function will be much less robust to outliers or misclassified data points. Another important difference between cross-entropy and the exponential error function is that the latter cannot be interpreted as the log likelihood function of any well-defined probabilistic model. Furthermore, the exponential error does not generalize to classification problems having $K > 2$ classes, again in contrast to the cross-entropy for a probabilistic model, which is easily generalized to give (4.108).

Section 7.1.2

Exercise 14.8

Section 4.3.4

The interpretation of boosting as the sequential optimization of an additive model under an exponential error (Friedman *et al.*, 2000) opens the door to a wide range of boosting-like algorithms, including multiclass extensions, by altering the choice of error function. It also motivates the extension to regression problems (Friedman, 2001). If we consider a sum-of-squares error function for regression, then sequential minimization of an additive model of the form (14.21) simply involves fitting each new base classifier to the residual errors $t_n - f_{m-1}(\mathbf{x}_n)$ from the previous model. As we have noted, however, the sum-of-squares error is not robust to outliers, and this

Exercise 14.9

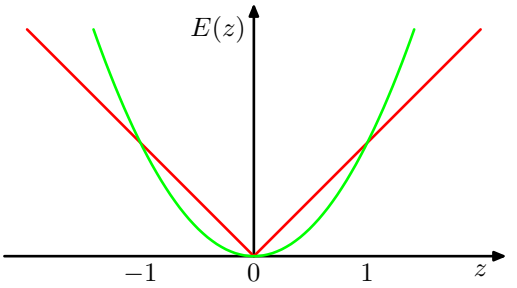


Figure 14.4 Comparison of the squared error (green) with the absolute error (red) showing how the latter places much less emphasis on large errors and hence is more robust to outliers and mislabelled data points.

can be addressed by basing the boosting algorithm on the absolute deviation $|y - t|$ instead. These two error functions are compared in Figure 14.4.

14.4. Tree-based Models

从这段介绍中可以看到单颗决策树也可视为一种集成学习方法。本书很介绍诸如GBDT, XGBoost等以决策树为基学习器的集成学习算法，而是介绍如何构建单颗决策树，因为单颗决策树本身也可视为一种集成学习(更准确地说是混合模型)

There are various simple, but widely used, models that work by partitioning the input space into cuboid regions, whose edges are aligned with the axes, and then assigning a simple model (for example, a constant) to each region. They can be viewed as a model combination method in which only one model is responsible for making predictions at any given point in input space. The process of selecting a specific model, given a new input $\mathbf{x}$, can be described by a sequential decision making process corresponding to the traversal of a binary tree (one that splits into two branches at each node). Here we focus on a particular tree-based framework called *classification and regression trees*, or *CART* (Breiman *et al.*, 1984), although there are many other variants going by such names as ID3 and C4.5 (Quinlan, 1986; Quinlan, 1993). and 14.6 举例 Figure 14.5 shows an illustration of a recursive binary partitioning of the input space, along with the corresponding tree structure. In this example, the first step

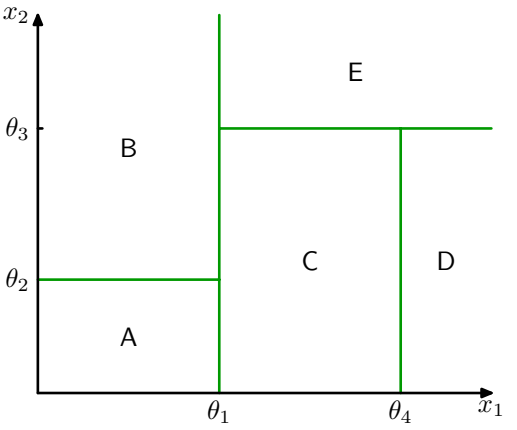


Figure 14.5 Illustration of a two-dimensional input space that has been partitioned into five regions using axis-aligned boundaries.

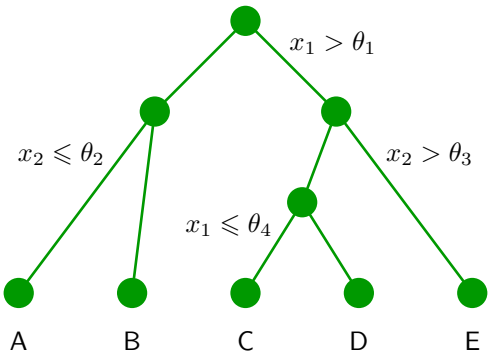


Figure 14.6 Binary tree corresponding to the partitioning of input space shown in Figure 14.5.

divides the whole of the input space into two regions according to whether $x_1 \leqslant \theta_1$ or $x_1 > \theta_1$ where θ_1 is a parameter of the model. This creates two subregions, each of which can then be subdivided independently. For instance, the region $x_1 \leqslant \theta_1$ is further subdivided according to whether $x_2 \leqslant \theta_2$ or $x_2 > \theta_2$, giving rise to the regions denoted A and B. The recursive subdivision can be described by the traversal of the binary tree shown in Figure 14.6. For any new input $\mathbf{x}$, we determine which region it falls into by starting at the top of the tree at the root node and following a path down to a specific leaf node according to the decision criteria at each node. Note that such decision trees are not probabilistic graphical models.

决策树模型的解释性较好。

Within each region, there is a separate model to predict the target variable. For instance, in regression we might simply predict a constant over each region, or in classification we might assign each region to a specific class. A key property of tree-based models, which makes them popular in fields such as medical diagnosis, for example, is that they are readily interpretable by humans because they correspond to a sequence of binary decisions applied to the individual input variables. For instance, to predict a patient's disease, we might first ask "is their temperature greater than some threshold?". If the answer is yes, then we might next ask "is their blood pressure less than some threshold?". Each leaf of the tree is then associated with a specific diagnosis.

决策树的学习

In order to learn such a model from a training set, we have to determine the structure of the tree, including which input variable is chosen at each node to form the split criterion as well as the value of the threshold parameter θ_i for the split. We also have to determine the values of the predictive variable within each region.

Consider first a regression problem in which the goal is to predict a single target variable t from a D-dimensional vector $\mathbf{x} = (x_1, \ldots, x_D)^{\mathrm{T}}$ of input variables. The training data consists of input vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ along with the corresponding continuous labels $\{t_1, \ldots, t_N\}$. If the partitioning of the input space is given, and we minimize the sum-of-squares error function, then the optimal value of the predictive variable within any given region is just given by the average of the values of t_n for those data points that fall in that region.

Exercise 14.10

Now consider how to determine the structure of the decision tree. Even for a fixed number of nodes in the tree, the problem of determining the optimal structure (including choice of input variable for each split as well as the corresponding thresh-

olds) to minimize the sum-of-squares error is usually computationally infeasible due to the combinatorially large number of possible solutions. Instead, a greedy optimization is generally done by starting with a single root node, corresponding to the whole input space, and then growing the tree by adding nodes one at a time. At each step there will be some number of candidate regions in input space that can be split, corresponding to the addition of a pair of leaf nodes to the existing tree. For each of these, there is a choice of which of the D input variables to split, as well as the value of the threshold. The joint optimization of the choice of region to split, and the choice of input variable and threshold, can be done efficiently by exhaustive search noting that, for a given choice of split variable and threshold, the optimal choice of predictive variable is given by the local average of the data, as noted earlier. This is repeated for all possible choices of variable to be split, and the one that gives the smallest residual sum-of-squares error is retained.

Given a greedy strategy for growing the tree, there remains the issue of when to stop adding nodes. A simple approach would be to stop when the reduction in residual error falls below some threshold. However, it is found empirically that often none of the available splits produces a significant reduction in error, and yet after several more splits a substantial error reduction is found. For this reason, it is common practice to grow a large tree, using a stopping criterion based on the number of data points associated with the leaf nodes, and then prune back the resulting tree. The pruning is based on a criterion that balances residual error against a measure of model complexity. If we denote the starting tree for pruning by T_0, then we define $T \subset T_0$ to be a subtree of T_0 if it can be obtained by pruning nodes from T_0 (in other words, by collapsing internal nodes by combining the corresponding regions). Suppose the leaf nodes are indexed by $\tau = 1, \ldots, |T|$, with leaf node τ representing a region $\mathcal{R}_\tau$ of input space having N_τ data points, and $|T|$ denoting the total number of leaf nodes. The optimal prediction for region $\mathcal{R}_\tau$ is then given by

$$y_\tau = \frac{1}{N_\tau} \sum_{\mathbf{x}_n \in \mathcal{R}_\tau} t_n \tag{14.29}$$

and the corresponding contribution to the residual sum-of-squares is then

$$Q_\tau(T) = \sum_{\mathbf{x}_n \in \mathcal{R}_\tau} \{t_n - y_\tau\}^2 . \tag{14.30}$$

The pruning criterion is then given by

$$C(T) = \sum_{\tau=1}^{|T|} Q_\tau(T) + \lambda |T| \tag{14.31}$$

The regularization parameter λ determines the trade-off between the overall residual sum-of-squares error and the complexity of the model as measured by the number $|T|$ of leaf nodes, and its value is chosen by cross-validation.

For classification problems, the process of growing and pruning the tree is similar, except that the sum-of-squares error is replaced by a more appropriate measure

of performance. If we define $p_{\tau k}$ to be the proportion of data points in region $\mathcal{R}_\tau$ assigned to class k, where $k = 1, \ldots, K$, then two commonly used choices are the cross-entropy

$$Q_\tau(T) = -\sum_{k=1}^{K} p_{\tau k} \ln p_{\tau k} \tag{14.32}$$

and the *Gini index*

$$Q_\tau(T) = \sum_{k=1}^{K} p_{\tau k} (1 - p_{\tau k}) . \tag{14.33}$$

if $p_{\tau k}=1$ for any one $k=1,\ldots,K$ (in which case $p_{\tau j}=0$ for all $j \neq k$) and have their maxima at $p_{\tau k}=1/K$ for all $k=1,\ldots,K$

These both vanish ~~for $p_{\tau k} = 0$ and $p_{\tau k} = 1$ and have a maximum at $p_{\tau k} = 0.5$~~. They encourage the formation of regions in which a high proportion of the data points are assigned to one class. The cross entropy and the Gini index are better measures than the misclassification rate for growing the tree because they are more sensitive to the node probabilities. Also, unlike misclassification rate, they are differentiable and hence better suited to gradient based optimization methods. For subsequent pruning of the tree, the misclassification rate is generally used.

优化 cross entropy 或 Gini index 比直接优化分类错误率要好

Exercise 14.11

决策树的优缺点

The human interpretability of a tree model such as CART is often seen as its major strength. However, in practice it is found that the particular tree structure that is learned is very sensitive to the details of the data set, so that a small change to the training data can result in a very different set of splits (Hastie *et al.*, 2001).

There are other problems with tree-based methods of the kind considered in this section. One is that the splits are aligned with the axes of the feature space, which may be very suboptimal. For instance, to separate two classes whose optimal decision boundary runs at 45 degrees to the axes would need a large number of axis-parallel splits of the input space as compared to a single non-axis-aligned split. Furthermore, the splits in a decision tree are hard, so that each region of input space is associated with one, and only one, leaf node model. The last issue is particularly problematic in regression where we are typically aiming to model smooth functions, and yet the tree model produces piecewise-constant predictions with discontinuities at the split boundaries.

14.5. Conditional Mixture Models

We have seen that standard decision trees are restricted by hard, axis-aligned splits of the input space. These constraints can be relaxed, at the expense of interpretability, by allowing soft, probabilistic splits that can be functions of all of the input variables, not just one of them at a time. If we also give the leaf models a probabilistic interpretation, we arrive at a fully probabilistic tree-based model called the *hierarchical mixture of experts*, which we consider in Section 14.5.3.

An alternative way to motivate the hierarchical mixture of experts model is to start with a standard probabilistic mixtures of unconditional density models such as Gaussians and replace the component densities with conditional distributions. Here we consider mixtures of linear regression models (Section 14.5.1) and mixtures of

Chapter 9

梳理一下：①混合模型中被混合的分布 condition on input × 就得到了 conditional mixture model 条件混合模型；
①→②→③ 而②条件混合模型中的混合系数 condition on input × 就得到了 mixture of experts model，即式(14.53)；
③ 进一步对③混合系数模型进行套娃就得到了 hierarchical mixture of experts model (HME model)。

HME model 可更比决策树模型，最底层的被混合的模型即experts 对应叶子结点，而有个 Gating function 对x 则对应一个中间结点；二者的不同之处在于，不论叶子结点还是中间结点，HME 均以一定概率进行软(soft)处理，而非像标准决策树那样为硬(hard)处理(包括区域的划分和最终结果的判定)。

logistic regression models (Section 14.5.2). In the simplest case, the mixing coefficients are independent of the input variables. If we make a further generalization to allow the mixing coefficients also to depend on the inputs then we obtain a *mixture of experts* model. Finally, if we allow each component in the mixture model to be itself a mixture of experts model, then we obtain a hierarchical mixture of experts.

14.5.1 Mixtures of linear regression models

One of the many advantages of giving a probabilistic interpretation to the linear regression model is that it can then be used as a component in more complex probabilistic models. This can be done, for instance, by viewing the conditional distribution representing the linear regression model as a node in a directed probabilistic graph. Here we consider a simple example corresponding to a mixture of linear regression models, which represents a straightforward extension of the Gaussian mixture model discussed in Section 9.2 to the case of conditional Gaussian distributions.

模型的介绍可以看到，与Section 9.2 GMM的区别在于混合的高斯分布 condition on x，即 π_k 中，π不再是与x无关的参数 π_k。也就是说，这是混合线性回归模型对应 Conditional Gaussian，而GMM对应 unconditional Gaussian.

We therefore consider K linear regression models, each governed by its own weight parameter $\mathbf{w}_k$. In many applications, it will be appropriate to use a common noise variance, governed by a precision parameter β, for all K components, and this is the case we consider here. We will once again restrict attention to a single target variable t, though the extension to multiple outputs is straightforward. If we denote the mixing coefficients by π_k, then the mixture distribution can be written

Exercise 14.12

$$p(t|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(t|\mathbf{w}_k^{\mathrm{T}}\boldsymbol{\phi}, \beta^{-1}) \tag{14.34}$$

where $\boldsymbol{\theta}$ denotes the set of all adaptive parameters in the model, namely $\mathbf{W} = \{\mathbf{w}_k\}$, $\boldsymbol{\pi} = \{\pi_k\}$, and β. The log likelihood function for this model, given a data set of observations $\{\boldsymbol{\phi}_n, t_n\}$, then takes the form

$$\ln p(\mathbf{t}|\boldsymbol{\theta}) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(t_n|\mathbf{w}_k^{\mathrm{T}}\boldsymbol{\phi}_n, \beta^{-1}) \right) \tag{14.35}$$

where $\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$ denotes the vector of target variables.

模型的学习 EM算法

In order to maximize this likelihood function, we can once again appeal to the EM algorithm, which will turn out to be a simple extension of the EM algorithm for unconditional Gaussian mixtures of Section 9.2. We can therefore build on our experience with the unconditional mixture and introduce a set $\mathbf{Z} = \{\mathbf{z}_n\}$ of binary latent variables where $z_{nk} \in \{0, 1\}$ in which, for each data point n, all of the elements $k = 1, \ldots, K$ are zero except for a single value of 1 indicating which component of the mixture was responsible for generating that data point. The joint distribution over latent and observed variables can be represented by the graphical model shown in Figure 14.7.

Exercise 14.13

The complete-data log likelihood function then takes the form

$$\ln p(\mathbf{t}, \mathbf{Z}|\boldsymbol{\theta}) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \ln \left\{ \pi_k \mathcal{N}(t_n|\mathbf{w}_k^{\mathrm{T}}\boldsymbol{\phi}_n, \beta^{-1}) \right\}. \tag{14.36}$$

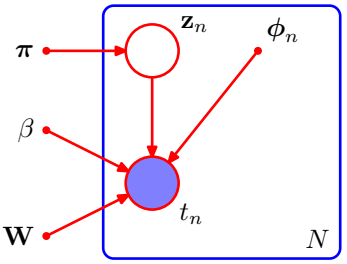


Figure 14.7 Probabilistic directed graph representing a mixture of linear regression models, defined by (14.35).

The EM algorithm begins by first choosing an initial value $\boldsymbol{\theta}^{\text{old}}$ for the model parameters. In the E step, these parameter values are then used to evaluate the posterior probabilities, or responsibilities, of each component k for every data point n given by

$$\gamma_{nk} = \mathbb{E}[z_{nk}] = p(k|\boldsymbol{\phi}_n, \boldsymbol{\theta}^{\text{old}}) = \frac{\pi_k \mathcal{N}(t_n|\mathbf{w}_k^{\text{T}}\boldsymbol{\phi}_n, \beta^{-1})}{\sum_j \pi_j \mathcal{N}(t_n|\mathbf{w}_j^{\text{T}}\boldsymbol{\phi}_n, \beta^{-1})}. \tag{14.37}$$

The responsibilities are then used to determine the expectation, with respect to the posterior distribution $p(\mathbf{Z}|\mathbf{t}, \boldsymbol{\theta}^{\text{old}})$, of the complete-data log likelihood, which takes the form

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \mathbb{E}_{\mathbf{Z}}\left[\ln p(\mathbf{t}, \mathbf{Z}|\boldsymbol{\theta})\right] = \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma_{nk}\left\{\ln \pi_k + \ln \mathcal{N}(t_n|\mathbf{w}_k^{\text{T}}\boldsymbol{\phi}_n, \beta^{-1})\right\}.$$

上面给出E步，下面开始讨论M步。

In the M step, we maximize the function $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ with respect to $\boldsymbol{\theta}$, keeping the γ_{nk} fixed. For the optimization with respect to the mixing coefficients π_k we need to take account of the constraint $\sum_k \pi_k = 1$, which can be done with the aid of a Lagrange multiplier, leading to an M-step re-estimation equation for π_k in the form

Exercise 14.14

$$\pi_k = \frac{1}{N}\sum_{n=1}^{N} \gamma_{nk}. \tag{14.38}$$

Note that this has exactly the same form as the corresponding result for a simple mixture of unconditional Gaussians given by (9.22).

Next consider the maximization with respect to the parameter vector $\mathbf{w}_k$ of the k^{th} linear regression model. Substituting for the Gaussian distribution, we see that the function $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$, as a function of the parameter vector $\mathbf{w}_k$, takes the form

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{n=1}^{N} \gamma_{nk}\left\{-\frac{\beta}{2}\left(t_n - \mathbf{w}_k^{\text{T}}\boldsymbol{\phi}_n\right)^2\right\} + \text{const} \tag{14.39}$$

where the constant term includes the contributions from other weight vectors $\mathbf{w}_j$ for $j \neq k$. Note that the quantity we are maximizing is similar to the (negative of the) standard sum-of-squares error (3.12) for a single linear regression model, but with the inclusion of the responsibilities γ_{nk}. This represents a *weighted least squares*

problem, in which the term corresponding to the n^{th} data point carries a weighting coefficient given by $\beta\gamma_{nk}$, which could be interpreted as an effective precision for each data point. We see that each component linear regression model in the mixture, governed by its own parameter vector $\mathbf{w}_k$, is fitted separately to the whole data set in the M step, but with each data point n weighted by the responsibility γ_{nk} that model k takes for that data point. Setting the derivative of (14.39) with respect to $\mathbf{w}_k$ equal to zero gives

$$0 = \sum_{n=1}^{N} \gamma_{nk} \left(t_n - \mathbf{w}_k^{\mathrm{T}} \boldsymbol{\phi}_n\right) \boldsymbol{\phi}_n \tag{14.40}$$

which we can write in matrix notation as

$$0 = \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{R}_k (\mathbf{t} - \boldsymbol{\Phi} \mathbf{w}_k) \tag{14.41}$$

where $\mathbf{R}_k = \mathrm{diag}(\gamma_{nk})$ is a diagonal matrix of size $N \times N$. Solving for $\mathbf{w}_k$, we obtain

$$\mathbf{w}_k = \left(\boldsymbol{\Phi}^{\mathrm{T}} \mathbf{R}_k \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{R}_k \mathbf{t}. \tag{14.42}$$

This represents a set of modified normal equations corresponding to the weighted least squares problem, of the same form as (4.99) found in the context of logistic regression. Note that after each E step, the matrix $\mathbf{R}_k$ will change and so we will have to solve the normal equations afresh in the subsequent M step.

Finally, we maximize $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}})$ with respect to β. Keeping only terms that depend on β, the function $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}})$ can be written

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} \left\{ \frac{1}{2} \ln \beta - \frac{\beta}{2} \left(t_n - \mathbf{w}_k^{\mathrm{T}} \boldsymbol{\phi}_n\right)^2 \right\}. \tag{14.43}$$

Setting the derivative with respect to β equal to zero, and rearranging, we obtain the M-step equation for β in the form

$$\frac{1}{\beta} = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} \left(t_n - \mathbf{w}_k^{\mathrm{T}} \boldsymbol{\phi}_n\right)^2. \tag{14.44}$$

举例

In Figure 14.8, we illustrate this EM algorithm using the simple example of fitting a mixture of two straight lines to a data set having one input variable x and one target variable t. The predictive density (14.34) is plotted in Figure 14.9 using the converged parameter values obtained from the EM algorithm, corresponding to the right-hand plot in Figure 14.8. Also shown in this figure is the result of fitting a single linear regression model, which gives a unimodal predictive density. We see that the mixture model gives a much better representation of the data distribution, and this is reflected in the higher likelihood value. However, the mixture model also assigns significant probability mass to regions where there is no data because its predictive distribution is bimodal for all values of x. This problem can be resolved by extending the model to allow the mixture coefficients themselves to be functions of x, leading to models such as the mixture density networks discussed in Section 5.6, and hierarchical mixture of experts discussed in Section 14.5.3.

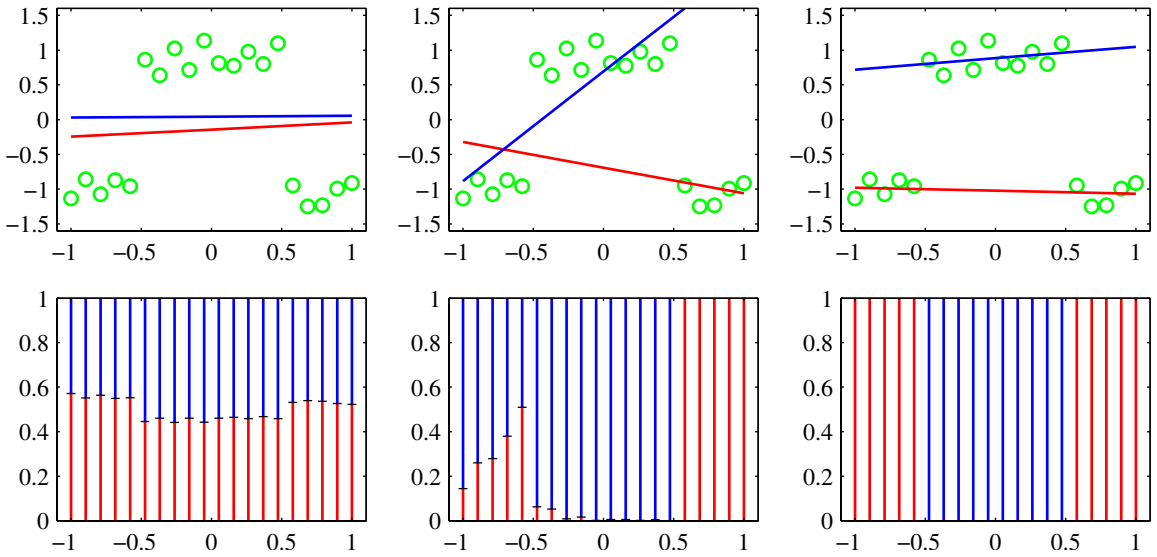


Figure 14.8 Example of a synthetic data set, shown by the green points, having one input variable x and one target variable t, together with a mixture of two linear regression models whose mean functions $y(x, \mathbf{w}_k)$, where $k \in \{1, 2\}$, are shown by the blue and red lines. The upper three plots show the initial configuration (left), the result of running 30 iterations of EM (centre), and the result after 50 iterations of EM (right). Here β was initialized to the reciprocal of the true variance of the set of target values. The lower three plots show the corresponding responsibilities plotted as a vertical line for each data point in which the length of the blue segment gives the posterior probability of the blue line for that data point (and similarly for the red segment).

14.5.1 小节针对回归问题介绍了混合线性回归，本小节则针对分类问题介绍了混合逻辑回归模型。

和上一小节的情况不同，logistic regression本身就condition on input x

14.5.2 Mixtures of logistic models

模型的介绍

Because the logistic regression model defines a conditional distribution for the target variable, given the input vector, it is straightforward to use it as the component distribution in a mixture model, thereby giving rise to a richer family of conditional distributions compared to a single logistic regression model. This example involves a straightforward combination of ideas encountered in earlier sections of the book and will help consolidate these for the reader.

The conditional distribution of the target variable, for a probabilistic mixture of K logistic regression models, is given by

$$p(t|\boldsymbol{\phi}, \boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k y_k^t \left[1 - y_k\right]^{1-t} \tag{14.45}$$

where $\boldsymbol{\phi}$ is the feature vector, $y_k = \sigma\left(\mathbf{w}_k^{\mathrm{T}} \boldsymbol{\phi}\right)$ is the output of component k, and $\boldsymbol{\theta}$ denotes the adjustable parameters namely $\{\pi_k\}$ and $\{\mathbf{w}_k\}$.

Now suppose we are given a data set $\{\boldsymbol{\phi}_n, t_n\}$. The corresponding likelihood

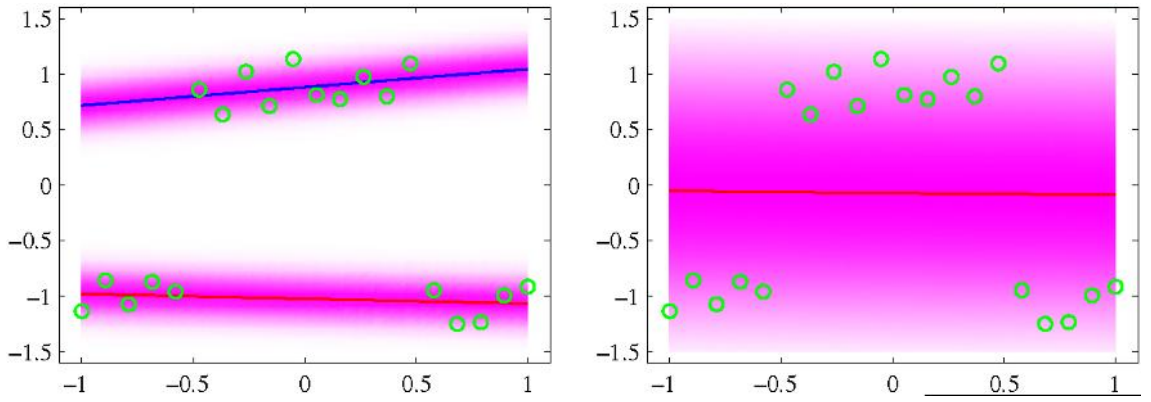


Figure 14.9 The left plot shows the predictive conditional density corresponding to the converged solution in Figure 14.8. This gives a log likelihood value of -3.0. A vertical slice through one of these plots at a particular value of x represents the corresponding conditional distribution $p(t|x)$, which we see is bimodal. The plot on the right shows the predictive density for a single linear regression model fitted to the same data set using maximum likelihood. This model has a smaller log likelihood of -27.6.

function is then given by

$$p(\mathbf{t}|\boldsymbol{\theta}) = \prod_{n=1}^{N} \left(\sum_{k=1}^{K} \pi_k y_{nk}^{t_n} \left[1 - y_{nk}\right]^{1-t_n} \right) \tag{14.46}$$

模型混合：EM算法

where $y_{nk} = \sigma(\mathbf{w}_k^{\mathrm{T}} \boldsymbol{\phi}_n)$ and $\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$. We can maximize this likelihood function iteratively by making use of the EM algorithm. This involves introducing latent variables z_{nk} that correspond to a 1-of-K coded binary indicator variable for each data point n. The complete-data likelihood function is then given by

$$p(\mathbf{t}, \mathbf{Z}|\boldsymbol{\theta}) = \prod_{n=1}^{N} \prod_{k=1}^{K} \left\{ \pi_k y_{nk}^{t_n} \left[1 - y_{nk}\right]^{1-t_n} \right\}^{z_{nk}} \tag{14.47}$$

where $\mathbf{Z}$ is the matrix of latent variables with elements z_{nk}. We initialize the EM algorithm by choosing an initial value $\boldsymbol{\theta}^{\mathrm{old}}$ for the model parameters. In the E step, we then use these parameter values to evaluate the posterior probabilities of the components k for each data point n, which are given by

$$\gamma_{nk} = \mathbb{E}[z_{nk}] = p(k|\boldsymbol{\phi}_n, \boldsymbol{\theta}^{\mathrm{old}}) = \frac{\pi_k y_{nk}^{t_n} \left[1 - y_{nk}\right]^{1-t_n}}{\sum_j \pi_j y_{nj}^{t_n} \left[1 - y_{nj}\right]^{1-t_n}}. \tag{14.48}$$

These responsibilities are then used to find the expected complete-data log likelihood as a function of $\boldsymbol{\theta}$, given by

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}}) &= \mathbb{E}_{\mathbf{Z}} \left[\ln p(\mathbf{t}, \mathbf{Z}|\boldsymbol{\theta})\right] \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} \left\{ \ln \pi_k + t_n \ln y_{nk} + (1 - t_n) \ln \left(1 - y_{nk}\right) \right\}. \end{aligned} \tag{14.49}$$

上面给出M步下面开始 // 讲M步

The M step involves maximization of this function with respect to $\boldsymbol{\theta}$, keeping $\boldsymbol{\theta}^{\text{old}}$, and hence γ_{nk}, fixed. Maximization with respect to π_k can be done in the usual way, with a Lagrange multiplier to enforce the summation constraint $\sum_k \pi_k = 1$, giving the familiar result

$$\pi_k = \frac{1}{N}\sum_{n=1}^{N}\gamma_{nk}. \tag{14.50}$$

To determine the $\{\mathbf{w}_k\}$, we note that the $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ function comprises a sum over terms indexed by k each of which depends only on one of the vectors $\mathbf{w}_k$, so that the different vectors are decoupled in the M step of the EM algorithm. In other words, the different components interact only via the responsibilities, which are fixed during the M step. Note that the M step does not have a closed-form solution and must be solved iteratively using, for instance, the iterative reweighted least squares (IRLS) algorithm. The gradient and the Hessian for the vector $\mathbf{w}_k$ are given by

M步没有解析解需采用迭代优化算法

Section 4.3.3

$$\nabla_k Q = \sum_{n=1}^{N}\gamma_{nk}(t_n - y_{nk})\boldsymbol{\phi}_n \tag{14.51}$$

$$\mathbf{H}_k = -\nabla_k \nabla_k Q = \sum_{n=1}^{N}\gamma_{nk} y_{nk}(1 - y_{nk})\boldsymbol{\phi}_n \boldsymbol{\phi}_n^{\mathrm{T}} \tag{14.52}$$

where ∇_k denotes the gradient with respect to $\mathbf{w}_k$. For fixed γ_{nk}, these are independent of $\{\mathbf{w}_j\}$ for $j \neq k$ and so we can solve for each $\mathbf{w}_k$ separately using the IRLS algorithm. Thus the M-step equations for component k correspond simply to fitting a single logistic regression model to a weighted data set in which data point n carries a weight γ_{nk}. Figure 14.10 shows an example of the mixture of logistic regression models applied to a simple classification problem. The extension of this model to a mixture of softmax models for more than two classes is straightforward.

Section 4.3.3

Exercise 14.16

14.5.3 Mixtures of experts

承上启下，引出 mixture of experts model，即式(14.53)。而对式(14.53)进行变换就得到了后文所介绍的 hierarchical mixture of experts model。

In Section 14.5.1, we considered a mixture of linear regression models, and in Section 14.5.2 we discussed the analogous mixture of linear classifiers. Although these simple mixtures extend the flexibility of linear models to include more complex (e.g., multimodal) predictive distributions, they are still very limited. We can further increase the capability of such models by allowing the mixing coefficients themselves to be functions of the input variable, so that

$$p(\mathbf{t}|\mathbf{x}) = \sum_{k=1}^{K}\pi_k(\mathbf{x})p_k(\mathbf{t}|\mathbf{x}). \tag{14.53}$$

This is known as a *mixture of experts* model (Jacobs *et al.*, 1991) in which the mixing coefficients $\pi_k(\mathbf{x})$ are known as *gating* functions and the individual component densities $p_k(\mathbf{t}|\mathbf{x})$ are called *experts*. The notion behind the terminology is that different components can model the distribution in different regions of input space (they

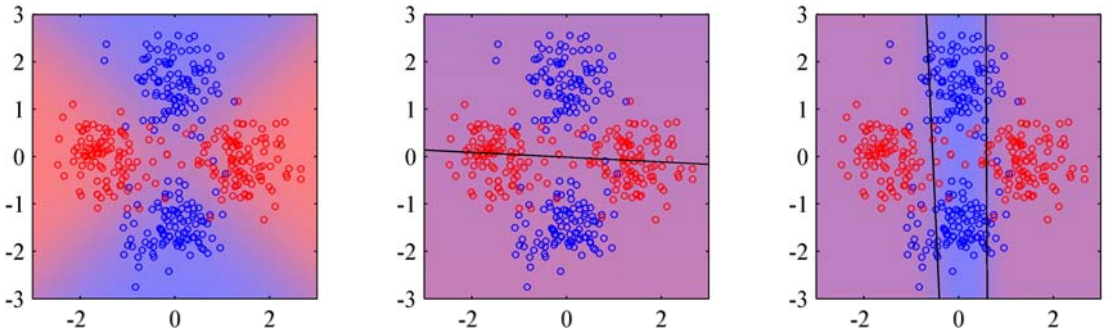


Figure 14.10 Illustration of a mixture of logistic regression models. The left plot shows data points drawn from two classes denoted red and blue, in which the background colour (which varies from pure red to pure blue) denotes the true probability of the class label. The centre plot shows the result of fitting a single logistic regression model using maximum likelihood, in which the background colour denotes the corresponding probability of the class label. Because the colour is a near-uniform purple, we see that the model assigns a probability of around 0.5 to each of the classes over most of input space. The right plot shows the result of fitting a mixture of two logistic regression models, which now gives much higher probability to the correct labels for many of the points in the blue class.

are 'experts' at making predictions in their own regions), and the gating functions determine which components are dominant in which region.

The gating functions $\pi_k(\mathbf{x})$ must satisfy the usual constraints for mixing coefficients, namely $0 \leqslant \pi_k(\mathbf{x}) \leqslant 1$ and $\sum_k \pi_k(\mathbf{x}) = 1$. They can therefore be represented, for example, by linear softmax models of the form (4.104) and (4.105). If the experts are also linear (regression or classification) models, then the whole model can be fitted efficiently using the EM algorithm, with iterative reweighted least squares being employed in the M step (Jordan and Jacobs, 1994).

Such a model still has significant limitations due to the use of linear models for the gating and expert functions. A much more flexible model is obtained by using a multilevel gating function to give the *hierarchical mixture of experts*, or *HME* model (Jordan and Jacobs, 1994). To understand the structure of this model, imagine a mixture distribution in which each component in the mixture is itself a mixture distribution. For simple unconditional mixtures, this hierarchical mixture is trivially equivalent to a single flat mixture distribution. However, when the mixing coefficients are input dependent, this hierarchical model becomes nontrivial. The HME model can also be viewed as a probabilistic version of *decision trees* discussed in Section 14.4 and can again be trained efficiently by maximum likelihood using an EM algorithm with IRLS in the M step. A Bayesian treatment of the HME has been given by Bishop and Svensén (2003) based on variational inference.

Exercise 14.17

Section 4.3.3

We shall not discuss the HME in detail here. However, it is worth pointing out the close connection with the *mixture density network* discussed in Section 5.6. The principal advantage of the mixtures of experts model is that it can be optimized by EM in which the M step for each mixture component and gating model involves a convex optimization (although the overall optimization is nonconvex). By contrast, the advantage of the mixture density network approach is that the component

densities and the mixing coefficients share the hidden units of the neural network. Furthermore, in the mixture density network, the splits of the input space are further relaxed compared to the hierarchical mixture of experts in that they are not only soft, and not constrained to be axis aligned, but they can also be nonlinear.

2023年7月14日04:26 完结撒花！

Exercises

14.1 (⋆⋆) **www** Consider a set models of the form $p(\mathbf{t}|\mathbf{x}, \mathbf{z}_h, \boldsymbol{\theta}_h, h)$ in which $\mathbf{x}$ is the input vector, $\mathbf{t}$ is the target vector, h indexes the different models, $\mathbf{z}_h$ is a latent variable for model h, and $\boldsymbol{\theta}_h$ is the set of parameters for model h. Suppose the models have prior probabilities $p(h)$ and that we are given a training set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and $\mathbf{T} = \{\mathbf{t}_1, \ldots, \mathbf{t}_N\}$. Write down the formulae needed to evaluate the predictive distribution $p(\mathbf{t}|\mathbf{x}, \mathbf{X}, \mathbf{T})$ in which the latent variables and the model index are marginalized out. Use these formulae to highlight the difference between Bayesian averaging of different models and the use of latent variables within a single model.

14.2 (⋆) The expected sum-of-squares error E_{AV} for a simple committee model can be defined by (14.10), and the expected error of the committee itself is given by (14.11). Assuming that the individual errors satisfy (14.12) and (14.13), derive the result (14.14).

14.3 (⋆) **www** By making use of Jensen's inequality (1.115), for the special case of the convex function $f(x) = x^2$, show that the average expected sum-of-squares error E_{AV} of the members of a simple committee model, given by (14.10), and the expected error E_{COM} of the committee itself, given by (14.11), satisfy

$$E_{\mathrm{COM}} \leqslant E_{\mathrm{AV}}. \tag{14.54}$$

14.4 (⋆⋆) By making use of Jensen's in equality (1.115), show that the result (14.54) derived in the previous exercise ~~holds~~ holds for any error function $E(y)$, not just sum-of-squares, provided it is a convex function of y.

14.5 (⋆⋆) **www** Consider a committee in which we allow unequal weighting of the constituent models, so that

$$y_{\mathrm{COM}}(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m y_m(\mathbf{x}). \tag{14.55}$$

In order to ensure that the predictions $y_{\mathrm{COM}}(\mathbf{x})$ remain within sensible limits, suppose that we require that they be bounded at each value of $\mathbf{x}$ by the minimum and maximum values given by any of the members of the committee, so that

$$y_{\min}(\mathbf{x}) \leqslant y_{\mathrm{COM}}(\mathbf{x}) \leqslant y_{\max}(\mathbf{x}). \tag{14.56}$$

Show that a necessary and sufficient condition for this constraint is that the coefficients α_m satisfy

$$\alpha_m \geqslant 0, \qquad \sum_{m=1}^{M} \alpha_m = 1. \tag{14.57}$$

14.6 (⋆) **www** By differentiating the error function (14.23) with respect to α_m, show that the parameters α_m in the AdaBoost algorithm are updated using (14.17) in which ϵ_m is defined by (14.16).

14.7 (⋆) By making a variational minimization of the expected exponential error function given by (14.27) with respect to all possible functions $y(\mathbf{x})$, show that the minimizing function is given by (14.28).

14.8 (⋆) Show that the exponential error function (14.20), which is minimized by the AdaBoost algorithm, does not correspond to the log likelihood of any well-behaved probabilistic model. This can be done by showing that the corresponding conditional distribution $p(t|\mathbf{x})$ cannot be correctly normalized.

14.9 (⋆) **www** Show that the sequential minimization of the sum-of-squares error function for an additive model of the form (14.21) in the style of boosting simply involves fitting each new base classifier to the residual errors $t_n - f_{m-1}(\mathbf{x}_n)$ from the previous model.

14.10 (⋆) Verify that if we minimize the sum-of-squares error between a set of training values $\{t_n\}$ and a single predictive value t, then the optimal solution for t is given by the mean of the $\{t_n\}$.

14.11 **Page 675** Exercise 14.11: The text of this exercise should be changed to "Consider a data set comprising 400 data points from class $\mathcal{C}_1$ and 400 data points from class $\mathcal{C}_2$. Suppose that a tree model A splits these into $(300, 100)$ assigned to the first leaf node (predicting $\mathcal{C}_1$) and $(100, 300)$ assigned to the second leaf node (predicting $\mathcal{C}_2$), where (n, m) denotes that n points come from class $\mathcal{C}_1$ and m points come from class $\mathcal{C}_2$. Similarly, suppose that a second tree model B splits them into $(200, 400)$ and $(200, 0)$, respectively. Evaluate the misclassification rates for the two trees and hence show that they are equal. Similarly, evaluate the pruning criterion (14.31) for the cross-entropy case (14.32) and the Gini index case (14.33) for the two trees and show that they are both lower for tree B than for tree A.".

14.12 (⋆⋆) Extend the results of Section 14.5.1 for a mixture of linear regression models to the case of multiple target values described by a vector $\mathbf{t}$. To do this, make use of the results of Section 3.1.5.

14.13 (⋆) **www** Verify that the complete-data log likelihood function for the mixture of linear regression models is given by (14.36).

14.14 (⋆) Use the technique of Lagrange multipliers (Appendix E) to show that the M-step re-estimation equation for the mixing coefficients in the mixture of linear regression models trained by maximum likelihood EM is given by (14.38).

14.15 (⋆) **www** We have already noted that if we use a squared loss function in a regression problem, the corresponding optimal prediction of the target variable for a new input vector is given by the conditional mean of the predictive distribution. Show that the conditional mean for the mixture of linear regression models discussed in Section 14.5.1 is given by a linear combination of the means of each component distribution. Note that if the conditional distribution of the target data is multimodal, the conditional mean can give poor predictions.

14.16 ($\star\star\star$) Extend the logistic regression mixture model of Section 14.5.2 to a mixture of softmax classifiers representing $C \geqslant 2$ classes. Write down the EM algorithm for determining the parameters of this model through maximum likelihood.

14.17 ($\star\star$) **WWW** Consider a mixture model for a conditional distribution $p(t|\mathbf{x})$ of the form

$$p(t|\mathbf{x}) = \sum_{k=1}^{K} \pi_k \psi_k(t|\mathbf{x}) \tag{14.58}$$

in which each mixture component $\psi_k(t|\mathbf{x})$ is itself a mixture model. Show that this two-level hierarchical mixture is equivalent to a conventional single-level mixture model. Now suppose that the mixing coefficients in both levels of such a hierarchical model are arbitrary functions of $\mathbf{x}$. Again, show that this hierarchical model is again equivalent to a single-level model with $\mathbf{x}$-dependent mixing coefficients. Finally, consider the case in which the mixing coefficients at both levels of the hierarchical mixture are constrained to be linear classification (logistic or softmax) models. Show that the hierarchical mixture cannot in general be represented by a single-level mixture having linear classification models for the mixing coefficients. Hint: to do this it is sufficient to construct a single counter-example, so consider a mixture of two components in which one of those components is itself a mixture of two components, with mixing coefficients given by linear-logistic models. Show that this cannot be represented by a single-level mixture of 3 components having mixing coefficients determined by a linear-softmax model.

Appendix A. Data Sets

In this appendix, we give a brief introduction to the data sets used to illustrate some of the algorithms described in this book. Detailed information on file formats for these data sets, as well as the data files themselves, can be obtained from the book web site:

http://research.microsoft.com/∼cmbishop/PRML

Handwritten Digits

The digits data used in this book is taken from the MNIST data set (LeCun *et al.*, 1998), which itself was constructed by modifying a subset of the much larger data set produced by NIST (the National Institute of Standards and Technology). It comprises a training set of $60,000$ examples and a test set of $10,000$ examples. Some of the data was collected from Census Bureau employees and the rest was collected from high-school children, and care was taken to ensure that the test examples were written by different individuals to the training examples.

The original NIST data had binary (black or white) pixels. To create MNIST, these images were size normalized to fit in a 20×20 pixel box while preserving their aspect ratio. As a consequence of the anti-aliasing used to change the resolution of the images, the resulting MNIST digits are grey scale. These images were then centred in a 28×28 box. Examples of the MNIST digits are shown in Figure A.1.

Error rates for classifying the digits range from 12% for a simple linear classifier, through 0.56% for a carefully designed support vector machine, to 0.4% for a convolutional neural network (LeCun *et al.*, 1998).

Figure A.1 One hundred examples of the MNIST digits chosen at random from the training set.

Oil Flow

This is a synthetic data set that arose out of a project aimed at measuring noninvasively the proportions of oil, water, and gas in North Sea oil transfer pipelines (Bishop and James, 1993). It is based on the principle of *dual-energy gamma densitometry*. The ideas is that if a narrow beam of gamma rays is passed through the pipe, the attenuation in the intensity of the beam provides information about the density of material along its path. Thus, for instance, the beam will be attenuated more strongly by oil than by gas.

A single attenuation measurement alone is not sufficient because there are two degrees of freedom corresponding to the fraction of oil and the fraction of water (the fraction of gas is redundant because the three fractions must add to one). To address this, two gamma beams of different energies (in other words different frequencies or wavelengths) are passed through the pipe along the same path, and the attenuation of each is measured. Because the absorbtion properties of different materials vary differently as a function of energy, measurement of the attenuations at the two energies provides two independent pieces of information. Given the known absorbtion properties of oil, water, and gas at the two energies, it is then a simple matter to calculate the average fractions of oil and water (and hence of gas) measured *along the path* of the gamma beams.

There is a further complication, however, associated with the motion of the materials along the pipe. If the flow velocity is small, then the oil floats on top of the water with the gas sitting above the oil. This is known as a *laminar* or *stratified*

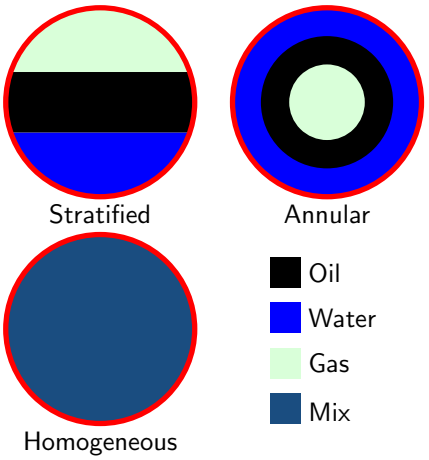


Figure A.2 The three geometrical configurations of the oil, water, and gas phases used to generate the oil-flow data set. For each configuration, the proportions of the three phases can vary.

flow configuration and is illustrated in Figure A.2. As the flow velocity is increased, more complex geometrical configurations of the oil, water, and gas can arise. For the purposes of this data set, two specific idealizations are considered. In the *annular* configuration the oil, water, and gas form concentric cylinders with the water around the outside and the gas in the centre, whereas in the *homogeneous* configuration the oil, water and gas are assumed to be intimately mixed as might occur at high flow velocities under turbulent conditions. These configurations are also illustrated in Figure A.2.

We have seen that a single dual-energy beam gives the oil and water fractions measured along the path length, whereas we are interested in the volume fractions of oil and water. This can be addressed by using multiple dual-energy gamma densitometers whose beams pass through different regions of the pipe. For this particular data set, there are six such beams, and their spatial arrangement is shown in Figure A.3. A single observation is therefore represented by a 12-dimensional vector comprising the fractions of oil and water measured along the paths of each of the beams. We are, however, interested in obtaining the overall volume fractions of the three phases in the pipe. This is much like the classical problem of tomographic reconstruction, used in medical imaging for example, in which a two-dimensional dis-

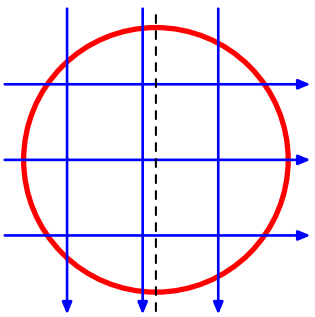


Figure A.3 Cross section of the pipe showing the arrangement of the six beam lines, each of which comprises a single dual-energy gamma densitometer. Note that the vertical beams are asymmetrically arranged relative to the central axis (shown by the dotted line).

tribution is to be reconstructed from an number of one-dimensional averages. Here there are far fewer line measurements than in a typical tomography application. On the other hand the range of geometrical configurations is much more limited, and so the configuration, as well as the phase fractions, can be predicted with reasonable accuracy from the densitometer data.

For safety reasons, the intensity of the gamma beams is kept relatively weak and so to obtain an accurate measurement of the attenuation, the measured beam intensity is integrated over a specific time interval. For a finite integration time, there are random fluctuations in the measured intensity due to the fact that the gamma beams comprise discrete packets of energy called photons. In practice, the integration time is chosen as a compromise between reducing the noise level (which requires a long integration time) and detecting temporal variations in the flow (which requires a short integration time). The oil flow data set is generated using realistic known values for the absorption properties of oil, water, and gas at the two gamma energies used, and with a specific choice of integration time (10 seconds) chosen as characteristic of a typical practical setup.

Each point in the data set is generated independently using the following steps:

1. Choose one of the three phase configurations at random with equal probability.

2. Choose three random numbers f_1, f_2 and f_3 from the uniform distribution over $(0, 1)$ and define

$$f_{\text{oil}} = \frac{f_1}{f_1 + f_2 + f_3}, \qquad f_{\text{water}} = \frac{f_2}{f_1 + f_2 + f_3}. \tag{A.1}$$

   This treats the three phases on an equal footing and ensures that the volume fractions add to one.

3. For each of the six beam lines, calculate the effective path lengths through oil and water for the given phase configuration.

4. Perturb the path lengths using the Poisson distribution based on the known beam intensities and integration time to allow for the effect of photon statistics.

Each point in the data set comprises the 12 path length measurements, together with the fractions of oil and water and a binary label describing the phase configuration. The data set is divided into training, validation, and test sets, each of which comprises $1,000$ independent data points. Details of the data format are available from the book web site.

In Bishop and James (1993), statistical machine learning techniques were used to predict the volume fractions and also the geometrical configuration of the phases shown in Figure A.2, from the 12-dimensional vector of measurements. The 12-dimensional observation vectors can also be used to test data visualization algorithms.

This data set has a rich and interesting structure, as follows. For any given configuration there are two degrees of freedom corresponding to the fractions of

oil and water, and so for infinite integration time the data will locally live on a two-dimensional manifold. For a finite integration time, the individual data points will be perturbed away from the manifold by the photon noise. In the homogeneous phase configuration, the path lengths in oil and water are linearly related to the fractions of oil and water, and so the data points lie close to a linear manifold. For the annular configuration, the relationship between phase fraction and path length is nonlinear and so the manifold will be nonlinear. In the case of the laminar configuration the situation is even more complex because small variations in the phase fractions can cause one of the horizontal phase boundaries to move across one of the horizontal beam lines leading to a discontinuous jump in the 12-dimensional observation space. In this way, the two-dimensional nonlinear manifold for the laminar configuration is broken into ~~six~~ distinct segments. Note also that some of the manifolds for different phase configurations meet at specific points, for example if the pipe is filled entirely with oil, it corresponds to specific instances of the laminar, annular, and homogeneous configurations.

Old Faithful

Old Faithful, shown in Figure A.4, is a hydrothermal geyser in Yellowstone National Park in the state of Wyoming, U.S.A., and is a popular tourist attraction. Its name stems from the supposed regularity of its eruptions.

The data set comprises 272 observations, each of which represents a single eruption and contains two variables corresponding to the duration in minutes of the eruption, and the time until the next eruption, also in minutes. Figure A.5 shows a plot of the time to the next eruption versus the duration of the eruptions. It can be seen that the time to the next eruption varies considerably, although knowledge of the duration of the current eruption allows it to be predicted more accurately. Note that there exist several other data sets relating to the eruptions of Old Faithful.

Figure A.4 The Old Faithful geyser in Yellowstone National Park. ©Bruce T. Gourley www.brucegourley.com.

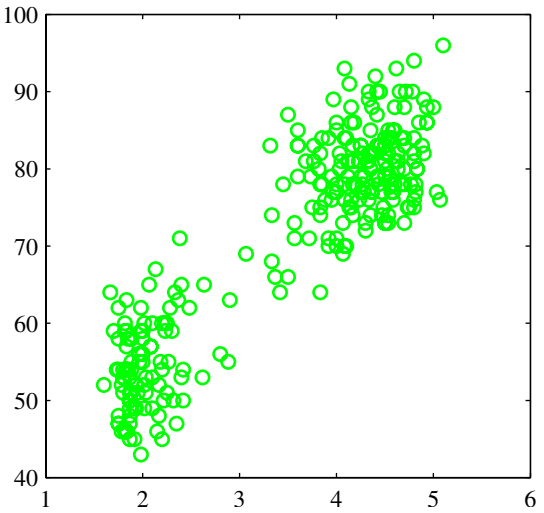


Figure A.5 Plot of the time to the next eruption in minutes (vertical axis) versus the duration of the eruption in minutes (horizontal axis) for the Old Faithful data set.

Synthetic Data

Throughout the book, we use two simple synthetic data sets to illustrate many of the algorithms. The first of these is a regression problem, based on the sinusoidal function, shown in Figure A.6. The input values $\{x_n\}$ are generated uniformly in range $(0, 1)$, and the corresponding target values $\{t_n\}$ are obtained by first computing the corresponding values of the function $\sin(2\pi x)$, and then adding random noise with a Gaussian distribution having standard deviation 0.3. Various forms of this data set, having different numbers of data points, are used in the book.

The second data set is a classification problem having two classes, with equal prior probabilities, and is shown in Figure A.7. The blue class is generated from a single Gaussian while the red class comes from a mixture of two Gaussians. Because we know the class priors and the class-conditional densities, it is straightforward to evaluate and plot the true posterior probabilities as well as the minimum misclassification-rate decision boundary, as shown in Figure A.7.

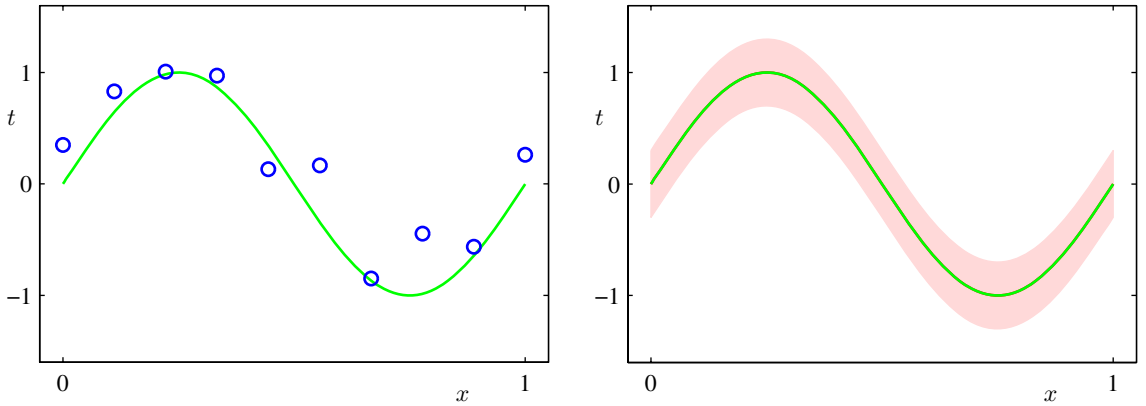


Figure A.6 The left-hand plot shows the synthetic regression data set along with the underlying sinusoidal function from which the data points were generated. The right-hand plot shows the true conditional distribution $p(t|x)$ from which the labels are generated, in which the green curve denotes the mean, and the shaded region spans one standard deviation on each side of the mean.

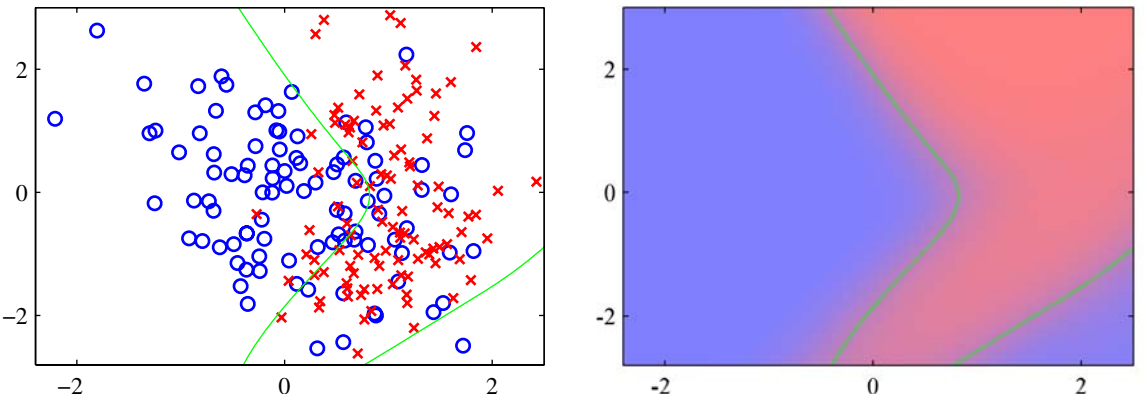


Figure A.7 The left plot shows the synthetic classification data set with data from the two classes shown in red and blue. On the right is a plot of the true posterior probabilities, shown on a colour scale going from pure red denoting probability of the red class is 1 to pure blue denoting probability of the red class is 0. Because these probabilities are known, the optimal decision boundary for minimizing the misclassification rate (which corresponds to the contour along which the posterior probabilities for each class equal 0.5) can be evaluated and is shown by the green curve. This decision boundary is also plotted on the left-hand figure.

Appendix B. Probability Distributions

The maximum of a distribution is known as its mode

In this appendix, we summarize the main properties of some of the most widely used probability distributions, and for each distribution we list some key statistics such as the expectation $\mathbb{E}[\mathbf{x}]$, the variance (or covariance), the mode, and the entropy $\mathrm{H}[\mathbf{x}]$. All of these distributions are members of the exponential family and are widely used as building blocks for more sophisticated probabilistic models.

Bernoulli

This is the distribution for a single binary variable $x \in \{0, 1\}$ representing, for example, the result of flipping a coin. It is governed by a single continuous parameter $\mu \in [0, 1]$ that represents the probability of $x = 1$.

$$\mathrm{Bern}(x|\mu) = \mu^x(1-\mu)^{1-x} \tag{B.1}$$

$$\mathbb{E}[x] = \mu \tag{B.2}$$

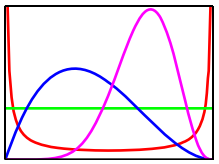
$$\mathrm{var}[x] = \mu(1-\mu) \tag{B.3}$$

$$\mathrm{mode}[x] = \begin{cases} 1 & \text{if } \mu \geqslant 0.5, \\ 0 & \text{otherwise} \end{cases} \tag{B.4}$$

$$\mathrm{H}[x] = -\mu \ln \mu - (1-\mu)\ln(1-\mu). \tag{B.5}$$

The Bernoulli is a special case of the binomial distribution for the case of a single observation. Its conjugate prior for μ is the beta distribution.

Beta



This is a distribution over a continuous variable $\mu \in [0, 1]$, which is often used to represent the probability for some binary event. It is governed by two parameters a and b that are constrained by $a > 0$ and $b > 0$ to ensure that the distribution can be normalized.

$$\text{Beta}(\mu|a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1-\mu)^{b-1} \tag{B.6}$$

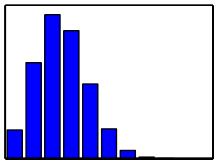
$$\mathbb{E}[\mu] = \frac{a}{a+b} \tag{B.7}$$

$$\text{var}[\mu] = \frac{ab}{(a+b)^2(a+b+1)} \tag{B.8}$$

$$\text{mode}[\mu] = \frac{a-1}{a+b-2}. \tag{B.9}$$

The beta is the conjugate prior for the Bernoulli distribution, for which a and b can be interpreted as the effective prior number of observations of $x = 1$ and $x = 0$, respectively. Its density is finite if $a \geqslant 1$ and $b \geqslant 1$, otherwise there is a singularity at $\mu = 0$ and/or $\mu = 1$. For $a = b = 1$, it reduces to a uniform distribution. The beta distribution is a special case of the K-state Dirichlet distribution for $K = 2$.

Binomial



The binomial distribution gives the probability of observing m occurrences of $x = 1$ in a set of N samples from a Bernoulli distribution, where the probability of observing $x = 1$ is $\mu \in [0, 1]$.

$$\text{Bin}(m|N, \mu) = \binom{N}{m}\mu^m(1-\mu)^{N-m} \tag{B.10}$$

$$\mathbb{E}[m] = N\mu \tag{B.11}$$

$$\text{var}[m] = N\mu(1-\mu) \tag{B.12}$$

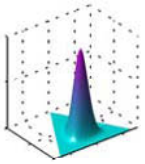
$$\text{mode}[m] = \lfloor (N+1)\mu \rfloor \tag{B.13}$$

where $\lfloor (N+1)\mu \rfloor$ denotes the largest integer that is less than or equal to $(N+1)\mu$, and the quantity

$$\binom{N}{m} = \frac{N!}{m!(N-m)!} \tag{B.14}$$

denotes the number of ways of choosing m objects out of a total of N identical objects. Here $m!$, pronounced 'factorial m', denotes the product $m \times (m-1) \times \ldots, \times 2 \times 1$. The particular case of the binomial distribution for $N = 1$ is known as the Bernoulli distribution, and for large N the binomial distribution is approximately Gaussian. The conjugate prior for μ is the beta distribution.

Dirichlet



The Dirichlet is a multivariate distribution over K random variables $0 \leqslant \mu_k \leqslant 1$, where $k = 1, \ldots, K$, subject to the constraints

$$0 \leqslant \mu_k \leqslant 1, \qquad \sum_{k=1}^{K} \mu_k = 1. \tag{B.15}$$

Denoting $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)^{\mathrm{T}}$ and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)^{\mathrm{T}}$, we have

$$\begin{aligned}
\mathrm{Dir}(\boldsymbol{\mu}|\boldsymbol{\alpha}) &= C(\boldsymbol{\alpha}) \prod_{k=1}^{K} \mu_k^{\alpha_k - 1} && \text{(B.16)}\\
\mathbb{E}[\mu_k] &= \frac{\alpha_k}{\widehat{\alpha}} && \text{(B.17)}\\
\mathrm{var}[\mu_k] &= \frac{\alpha_k(\widehat{\alpha} - \alpha_k)}{\widehat{\alpha}^2(\widehat{\alpha} + 1)} && \text{(B.18)}\\
\mathrm{cov}[\mu_j \mu_k] &= -\frac{\alpha_j \alpha_k}{\widehat{\alpha}^2(\widehat{\alpha} + 1)} && \text{(B.19)}\\
\mathrm{mode}[\mu_k] &= \frac{\alpha_k - 1}{\widehat{\alpha} - K} && \text{(B.20)}\\
\mathbb{E}[\ln \mu_k] &= \psi(\alpha_k) - \psi(\widehat{\alpha}) && \text{(B.21)}\\
\mathrm{H}[\boldsymbol{\mu}] &= -\sum_{k=1}^{K} (\alpha_k - 1)\{\psi(\alpha_k) - \psi(\widehat{\alpha})\} - \ln C(\boldsymbol{\alpha}) && \text{(B.22)}
\end{aligned}$$

where

$$C(\boldsymbol{\alpha}) = \frac{\Gamma(\widehat{\alpha})}{\Gamma(\alpha_1) \cdots \Gamma(\alpha_K)} \tag{B.23}$$

and

$$\widehat{\alpha} = \sum_{k=1}^{K} \alpha_k. \tag{B.24}$$

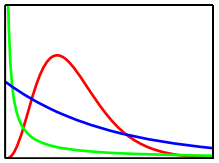
Here

$$\psi(a) \equiv \frac{d}{da} \ln \Gamma(a) \tag{B.25}$$

is known as the *digamma* function (Abramowitz and Stegun, 1965). The parameters α_k are subject to the constraint $\alpha_k > 0$ in order to ensure that the distribution can be normalized.

The Dirichlet forms the conjugate prior for the multinomial distribution and represents a generalization of the beta distribution. In this case, the parameters α_k can be interpreted as effective numbers of observations of the corresponding values of the K-dimensional binary observation vector $\mathbf{x}$. As with the beta distribution, the Dirichlet has finite density everywhere provided $\alpha_k \geqslant 1$ for all k.

Gamma



The Gamma is a probability distribution over a positive random variable $\tau > 0$ governed by parameters a and b that are subject to the constraints $a > 0$ and $b > 0$ to ensure that the distribution can be normalized.

$$\mathrm{Gam}(\tau|a,b) = \frac{1}{\Gamma(a)} b^a \tau^{a-1} e^{-b\tau} \tag{B.26}$$

$$\mathbb{E}[\tau] = \frac{a}{b} \tag{B.27}$$

$$\mathrm{var}[\tau] = \frac{a}{b^2} \tag{B.28}$$

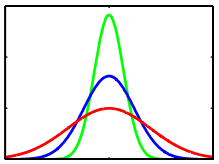
$$\mathrm{mode}[\tau] = \frac{a-1}{b} \quad \text{for } a \geqslant 1 \tag{B.29}$$

$$\mathbb{E}[\ln \tau] = \psi(a) - \ln b \tag{B.30}$$

$$\mathrm{H}[\tau] = \ln \Gamma(a) - (a-1)\psi(a) - \ln b + a \tag{B.31}$$

where $\psi(\cdot)$ is the digamma function defined by (B.25). The gamma distribution is the conjugate prior for the precision (inverse variance) of a univariate Gaussian. For $a \geqslant 1$ the density is everywhere finite, and the special case of $a = 1$ is known as the *exponential* distribution.

Gaussian



The Gaussian is the most widely used distribution for continuous variables. It is also known as the *normal* distribution. In the case of a single variable $x \in (-\infty, \infty)$ it is governed by two parameters, the mean $\mu \in (-\infty, \infty)$ and the variance $\sigma^2 > 0$.

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} \tag{B.32}$$

$$\mathbb{E}[x] = \mu \tag{B.33}$$

$$\mathrm{var}[x] = \sigma^2 \tag{B.34}$$

$$\mathrm{mode}[x] = \mu \tag{B.35}$$

$$\mathrm{H}[x] = \frac{1}{2}\ln\sigma^2 + \frac{1}{2}\left(1 + \ln(2\pi)\right). \tag{B.36}$$

The inverse of the variance $\tau = 1/\sigma^2$ is called the precision, and the square root of the variance σ is called the standard deviation. The conjugate prior for μ is the Gaussian, and the conjugate prior for τ is the gamma distribution. If both μ and τ are unknown, their joint conjugate prior is the Gaussian-gamma distribution.

For a D-dimensional vector $\mathbf{x}$, the Gaussian is governed by a D-dimensional mean vector $\boldsymbol{\mu}$ and a $D \times D$ covariance matrix $\boldsymbol{\Sigma}$ that must be symmetric and

positive-definite.

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\} \quad \text{(B.37)}$$

$$\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu} \quad \text{(B.38)}$$

$$\mathrm{cov}[\mathbf{x}] = \boldsymbol{\Sigma} \quad \text{(B.39)}$$

$$\mathrm{mode}[\mathbf{x}] = \boldsymbol{\mu} \quad \text{(B.40)}$$

$$\mathrm{H}[\mathbf{x}] = \frac{1}{2}\ln|\boldsymbol{\Sigma}| + \frac{D}{2}(1+\ln(2\pi)). \quad \text{(B.41)}$$

The inverse of the covariance matrix $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ is the precision matrix, which is also symmetric and positive definite. Averages of random variables tend to a Gaussian, by the central limit theorem, and the sum of two Gaussian variables is again Gaussian. The Gaussian is the distribution that maximizes the entropy for a given variance (or covariance). Any linear transformation of a Gaussian random variable is again Gaussian. The marginal distribution of a multivariate Gaussian with respect to a subset of the variables is itself Gaussian, and similarly the conditional distribution is also Gaussian. The conjugate prior for $\boldsymbol{\mu}$ is the Gaussian, the conjugate prior for $\boldsymbol{\Lambda}$ is the Wishart, and the conjugate prior for $(\boldsymbol{\mu},\boldsymbol{\Lambda})$ is the Gaussian-Wishart.

If we have a marginal Gaussian distribution for $\mathbf{x}$ and a conditional Gaussian distribution for $\mathbf{y}$ given $\mathbf{x}$ in the form

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Lambda}^{-1}) \quad \text{(B.42)}$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x}+\mathbf{b},\mathbf{L}^{-1}) \quad \text{(B.43)}$$

then the marginal distribution of $\mathbf{y}$, and the conditional distribution of $\mathbf{x}$ given $\mathbf{y}$, are given by

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu}+\mathbf{b},\mathbf{L}^{-1}+\mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}) \quad \text{(B.44)}$$

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y}-\mathbf{b})+\boldsymbol{\Lambda}\boldsymbol{\mu}\},\boldsymbol{\Sigma}) \quad \text{(B.45)}$$

where

$$\boldsymbol{\Sigma} = (\boldsymbol{\Lambda}+\mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1}. \quad \text{(B.46)}$$

If we have a joint Gaussian distribution $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma})$ with $\boldsymbol{\Lambda} \equiv \boldsymbol{\Sigma}^{-1}$ and we define the following partitions

$$\mathbf{x} = \begin{pmatrix}\mathbf{x}_a\\ \mathbf{x}_b\end{pmatrix}, \qquad \boldsymbol{\mu} = \begin{pmatrix}\boldsymbol{\mu}_a\\ \boldsymbol{\mu}_b\end{pmatrix} \quad \text{(B.47)}$$

$$\boldsymbol{\Sigma} = \begin{pmatrix}\boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab}\\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb}\end{pmatrix}, \qquad \boldsymbol{\Lambda} = \begin{pmatrix}\boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab}\\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb}\end{pmatrix} \quad \text{(B.48)}$$

then the conditional distribution $p(\mathbf{x}_a|\mathbf{x}_b)$ is given by

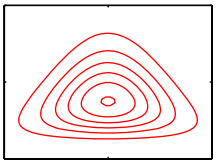
$$p(\mathbf{x}_a|\mathbf{x}_b) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{a|b},\boldsymbol{\Lambda}_{aa}^{-1}) \quad \text{(B.49)}$$

$$\boldsymbol{\mu}_{a|b} = \boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{aa}^{-1}\boldsymbol{\Lambda}_{ab}(\mathbf{x}_b-\boldsymbol{\mu}_b) \quad \text{(B.50)}$$

and the marginal distribution $p(\mathbf{x}_a)$ is given by

$$p(\mathbf{x}_a) = \mathcal{N}(\mathbf{x}_a|\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_{aa}). \tag{B.51}$$

Gaussian-Gamma



This is the conjugate prior distribution for a univariate Gaussian $\mathcal{N}(x|\mu, \lambda^{-1})$ in which the mean μ and the precision λ are both unknown and is also called the *normal-gamma* distribution. It comprises the product of a Gaussian distribution for μ, whose precision is proportional to λ, and a gamma distribution over λ.

$$p(\mu, \lambda|\mu_0, \beta, a, b) = \mathcal{N}\left(\mu|\mu_o, (\beta\lambda)^{-1}\right) \mathrm{Gam}(\lambda|a, b). \tag{B.52}$$

Gaussian-Wishart

This is the conjugate prior distribution for a multivariate Gaussian $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda})$ in which both the mean $\boldsymbol{\mu}$ and the precision $\boldsymbol{\Lambda}$ are unknown, and is also called the normal-Wishart distribution. It comprises the product of a Gaussian distribution for $\boldsymbol{\mu}$, whose precision is proportional to $\boldsymbol{\Lambda}$, and a Wishart distribution over $\boldsymbol{\Lambda}$.

$$p(\boldsymbol{\mu}, \boldsymbol{\Lambda}|\boldsymbol{\mu}_0, \beta, \mathbf{W}, \nu) = \mathcal{N}\left(\boldsymbol{\mu}|\boldsymbol{\mu}_0, (\beta\boldsymbol{\Lambda})^{-1}\right) \mathcal{W}(\boldsymbol{\Lambda}|\mathbf{W}, \nu). \tag{B.53}$$

For the particular case of a scalar x, this is equivalent to the Gaussian-gamma distribution.

Multinomial

If we generalize the Bernoulli distribution to an K-dimensional binary variable $\mathbf{x}$ with components $x_k \in \{0, 1\}$ such that $\sum_k x_k = 1$, then we obtain the following discrete distribution

$$p(\mathbf{x}) = \prod_{k=1}^{K} \mu_k^{x_k} \tag{B.54}$$

$$\mathbb{E}[x_k] = \mu_k \tag{B.55}$$

$$\mathrm{var}[x_k] = \mu_k(1 - \mu_k) \tag{B.56}$$

$$\mathrm{cov}[x_j x_k] = -\mu_j \mu_k, \; j \neq k \tag{B.57}$$

$$\mathrm{H}[\mathbf{x}] = -\sum_{k=1}^{K} \mu_k \ln \mu_k \tag{B.58}$$

where I_{jk} is the j, k element of the identity matrix. Because $p(x_k = 1) = \mu_k$, the parameters must satisfy $0 \leqslant \mu_k \leqslant 1$ and $\sum_k \mu_k = 1$.

The multinomial distribution is a multivariate generalization of the binomial and gives the distribution over counts m_k for a K-state discrete variable to be in state k given a total number of observations N.

$$\text{Mult}(m_1, m_2, \ldots, m_K|\boldsymbol{\mu}, N) = \binom{N}{m_1 m_2 \ldots m_K} \prod_{k=1}^{K} \mu_k^{m_k} \tag{B.59}$$

$$\mathbb{E}[m_k] = N\mu_k \tag{B.60}$$

$$\text{var}[m_k] = N\mu_k(1 - \mu_k) \tag{B.61}$$

$$\text{cov}[m_j m_k] = -N\mu_j\mu_k, \quad j \neq k \tag{B.62}$$

where $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)^{\text{T}}$, and the quantity

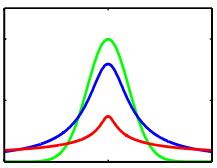
$$\binom{N}{m_1 m_2 \ldots m_K} = \frac{N!}{m_1! \ldots m_K!} \tag{B.63}$$

gives the number of ways of taking N identical objects and assigning m_k of them to bin k for $k = 1, \ldots, K$. The value of μ_k gives the probability of the random variable taking state k, and so these parameters are subject to the constraints $0 \leqslant \mu_k \leqslant 1$ and $\sum_k \mu_k = 1$. The conjugate prior distribution for the parameters $\{\mu_k\}$ is the Dirichlet.

Normal

The normal distribution is simply another name for the Gaussian. In this book, we use the term Gaussian throughout, although we retain the conventional use of the symbol $\mathcal{N}$ to denote this distribution. For consistency, we shall refer to the normal-gamma distribution as the Gaussian-gamma distribution, and similarly the normal-Wishart is called the Gaussian-Wishart.

Student's t



This distribution was published by William Gosset in 1908, but his employer, Guiness Breweries, required him to publish under a pseudonym, so he chose 'Student'. In the univariate form, Student's t-distribution is obtained by placing a conjugate gamma prior over the precision of a univariate Gaussian distribution and then integrating out the precision variable. It can therefore be viewed as an infinite mixture

of Gaussians having the same mean but different variances.

$$\mathrm{St}(x|\mu,\lambda,\nu) = \frac{\Gamma(\nu/2+1/2)}{\Gamma(\nu/2)}\left(\frac{\lambda}{\pi\nu}\right)^{1/2}\left[1+\frac{\lambda(x-\mu)^2}{\nu}\right]^{-\nu/2-1/2} \tag{B.64}$$

$$\mathbb{E}[x] = \mu \quad \text{for } \nu > 1 \tag{B.65}$$

$$\mathrm{var}[x] = \frac{1}{\lambda}\frac{\nu}{\nu-2} \quad \text{for } \nu > 2 \tag{B.66}$$

$$\mathrm{mode}[x] = \mu. \tag{B.67}$$

Here $\nu > 0$ is called the number of degrees of freedom of the distribution. The particular case of $\nu = 1$ is called the *Cauchy* distribution.

For a D-dimensional variable $\mathbf{x}$, Student's t-distribution corresponds to marginalizing the precision matrix of a multivariate Gaussian with respect to a conjugate Wishart prior and takes the form

$$\mathrm{St}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Lambda},\nu) = \frac{\Gamma(\nu/2+D/2)}{\Gamma(\nu/2)}\frac{|\boldsymbol{\Lambda}|^{1/2}}{(\nu\pi)^{D/2}}\left[1+\frac{\Delta^2}{\nu}\right]^{-\nu/2-D/2} \tag{B.68}$$

$$\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu} \quad \text{for } \nu > 1 \tag{B.69}$$

$$\mathrm{cov}[\mathbf{x}] = \frac{\nu}{\nu-2}\boldsymbol{\Lambda}^{-1} \quad \text{for } \nu > 2 \tag{B.70}$$

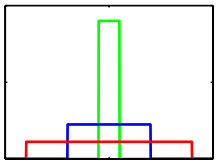
$$\mathrm{mode}[\mathbf{x}] = \boldsymbol{\mu} \tag{B.71}$$

where Δ^2 is the squared Mahalanobis distance defined by

$$\Delta^2 = (\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Lambda}(\mathbf{x}-\boldsymbol{\mu}). \tag{B.72}$$

In the limit $\nu \to \infty$, the t-distribution reduces to a Gaussian with mean μ and precision $\boldsymbol{\Lambda}$. Student's t-distribution provides a generalization of the Gaussian whose maximum likelihood parameter values are robust to outliers.

Uniform



This is a simple distribution for a continuous variable x defined over a finite interval $x \in [a,b]$ where $b > a$.

$$\mathrm{U}(x|a,b) = \frac{1}{b-a} \tag{B.73}$$

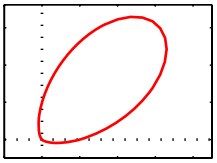
$$\mathbb{E}[x] = \frac{(b+a)}{2} \tag{B.74}$$

$$\mathrm{var}[x] = \frac{(b-a)^2}{12} \tag{B.75}$$

$$\mathrm{H}[x] = \ln(b-a). \tag{B.76}$$

If x has distribution $\mathrm{U}(x|0,1)$, then $a+(b-a)x$ will have distribution $\mathrm{U}(x|a,b)$.

Von Mises



The von Mises distribution, also known as the circular normal or the circular Gaussian, is a univariate Gaussian-like periodic distribution for a variable $\theta \in [0, 2\pi)$.

$$p(\theta|\theta_0, m) = \frac{1}{2\pi I_0(m)} \exp\{m\cos(\theta - \theta_0)\} \tag{B.77}$$

where $I_0(m)$ is the zeroth-order Bessel function of the first kind. The distribution has period 2π so that $p(\theta + 2\pi) = p(\theta)$ for all θ. Care must be taken in interpreting this distribution because simple expectations will be dependent on the (arbitrary) choice of origin for the variable θ. The parameter θ_0 is analogous to the mean of a univariate Gaussian, and the parameter $m > 0$, known as the *concentration* parameter, is analogous to the precision (inverse variance). For large m, the von Mises distribution is approximately a Gaussian centred on θ_0.

Wishart

The Wishart distribution is the conjugate prior for the precision matrix of a multivariate Gaussian.

$$\mathcal{W}(\mathbf{\Lambda}|\mathbf{W}, \nu) = B(\mathbf{W}, \nu)|\mathbf{\Lambda}|^{(\nu-D-1)/2} \exp\left(-\frac{1}{2}\mathrm{Tr}(\mathbf{W}^{-1}\mathbf{\Lambda})\right) \tag{B.78}$$

where

$$B(\mathbf{W}, \nu) \equiv |\mathbf{W}|^{-\nu/2} \left(2^{\nu D/2}\, \pi^{D(D-1)/4} \prod_{i=1}^{D} \Gamma\left(\frac{\nu + 1 - i}{2}\right)\right)^{-1} \tag{B.79}$$

$$\mathbb{E}[\mathbf{\Lambda}] = \nu\mathbf{W} \tag{B.80}$$

$$\mathbb{E}\left[\ln|\mathbf{\Lambda}|\right] = \sum_{i=1}^{D} \psi\left(\frac{\nu + 1 - i}{2}\right) + D\ln 2 + \ln|\mathbf{W}| \tag{B.81}$$

$$\mathrm{H}[\mathbf{\Lambda}] = -\ln B(\mathbf{W}, \nu) - \frac{(\nu - D - 1)}{2}\mathbb{E}\left[\ln|\mathbf{\Lambda}|\right] + \frac{\nu D}{2} \tag{B.82}$$

where $\mathbf{W}$ is a $D \times D$ symmetric, positive definite matrix, and $\psi(\cdot)$ is the digamma function defined by (B.25). The parameter ν is called the *number of degrees of freedom* of the distribution and is restricted to $\nu > D - 1$ to ensure that the Gamma function in the normalization factor is well-defined. In one dimension, the Wishart reduces to the gamma distribution $\mathrm{Gam}(\lambda|a, b)$ given by (B.26) with parameters $a = \nu/2$ and $b = 1/2W$.

Appendix C. Properties of Matrices

In this appendix, we gather together some useful properties and identities involving matrices and determinants. This is not intended to be an introductory tutorial, and it is assumed that the reader is already familiar with basic linear algebra. For some results, we indicate how to prove them, whereas in more complex cases we leave the interested reader to refer to standard textbooks on the subject. In all cases, we assume that inverses exist and that matrix dimensions are such that the formulae are correctly defined. A comprehensive discussion of linear algebra can be found in Golub and Van Loan (1996), and an extensive collection of matrix properties is given by Lütkepohl (1996). Matrix derivatives are discussed in Magnus and Neudecker (1999).

Basic Matrix Identities

A matrix $\mathbf{A}$ has elements A_{ij} where i indexes the rows, and j indexes the columns. We use $\mathbf{I}_N$ to denote the $N \times N$ identity matrix (also called the unit matrix), and where there is no ambiguity over dimensionality we simply use $\mathbf{I}$. The transpose matrix $\mathbf{A}^{\mathrm{T}}$ has elements $(\mathbf{A}^{\mathrm{T}})_{ij} = A_{ji}$. From the definition of transpose, we have

$$(\mathbf{AB})^{\mathrm{T}} = \mathbf{B}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}} \tag{C.1}$$

which can be verified by writing out the indices. The inverse of $\mathbf{A}$, denoted $\mathbf{A}^{-1}$, satisfies

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}. \tag{C.2}$$

Because $\mathbf{ABB}^{-1}\mathbf{A}^{-1} = \mathbf{I}$, we have

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}. \tag{C.3}$$

Also we have

$$\left(\mathbf{A}^{\mathrm{T}}\right)^{-1} = \left(\mathbf{A}^{-1}\right)^{\mathrm{T}} \tag{C.4}$$

which is easily proven by taking the transpose of (C.2) and applying (C.1).

A useful identity involving matrix inverses is the following

$$(\mathbf{P}^{-1} + \mathbf{B}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{B})^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{R}^{-1} = \mathbf{P}\mathbf{B}^{\mathrm{T}}(\mathbf{B}\mathbf{P}\mathbf{B}^{\mathrm{T}} + \mathbf{R})^{-1}. \tag{C.5}$$

which is easily verified by right multiplying both sides by $(\mathbf{B}\mathbf{P}\mathbf{B}^{\mathrm{T}} + \mathbf{R})$. Suppose that $\mathbf{P}$ has dimensionality $N \times N$ while $\mathbf{R}$ has dimensionality $M \times M$, so that $\mathbf{B}$ is $M \times N$. Then if $M \ll N$, it will be much cheaper to evaluate the right-hand side of (C.5) than the left-hand side. A special case that sometimes arises is

$$(\mathbf{I} + \mathbf{A}\mathbf{B})^{-1}\mathbf{A} = \mathbf{A}(\mathbf{I} + \mathbf{B}\mathbf{A})^{-1}. \tag{C.6}$$

Another useful identity involving inverses is the following:

$$(\mathbf{A} + \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{D} + \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} \tag{C.7}$$

which is known as the *Woodbury identity* and which can be verified by multiplying both sides by $(\mathbf{A} + \mathbf{B}\mathbf{D}^{-1}\mathbf{C})$. This is useful, for instance, when $\mathbf{A}$ is large and diagonal, and hence easy to invert, while $\mathbf{B}$ has many rows but few columns (and conversely for $\mathbf{C}$) so that the right-hand side is much cheaper to evaluate than the left-hand side.

A set of vectors $\{\mathbf{a}_1, \ldots, \mathbf{a}_N\}$ is said to be *linearly independent* if the relation $\sum_n \alpha_n \mathbf{a}_n = 0$ holds only if all $\alpha_n = 0$. This implies that none of the vectors can be expressed as a linear combination of the remainder. The rank of a matrix is the maximum number of linearly independent rows (or equivalently the maximum number of linearly independent columns).

Traces and Determinants

Trace and determinant apply to square matrices. The trace $\mathrm{Tr}(\mathbf{A})$ of a matrix $\mathbf{A}$ is defined as the sum of the elements on the leading diagonal. By writing out the indices, we see that

$$\mathrm{Tr}(\mathbf{A}\mathbf{B}) = \mathrm{Tr}(\mathbf{B}\mathbf{A}). \tag{C.8}$$

By applying this formula multiple times to the product of three matrices, we see that

$$\mathrm{Tr}(\mathbf{A}\mathbf{B}\mathbf{C}) = \mathrm{Tr}(\mathbf{C}\mathbf{A}\mathbf{B}) = \mathrm{Tr}(\mathbf{B}\mathbf{C}\mathbf{A}) \tag{C.9}$$

which is known as the *cyclic* property of the trace operator and which clearly extends to the product of any number of matrices. The determinant $|\mathbf{A}|$ of an $N \times N$ matrix $\mathbf{A}$ is defined by

$$|\mathbf{A}| = \sum (\pm 1) A_{1i_1} A_{2i_2} \cdots A_{Ni_N} \tag{C.10}$$

in which the sum is taken over all products consisting of precisely one element from each row and one element from each column, with a coefficient $+1$ or -1 according

to whether the permutation $i_1 i_2 \ldots i_N$ is even or odd, respectively. Note that $|\mathbf{I}| = 1$. Thus, for a 2×2 matrix, the determinant takes the form

$$|\mathbf{A}| = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}. \tag{C.11}$$

The determinant of a product of two matrices is given by

$$|\mathbf{AB}| = |\mathbf{A}||\mathbf{B}| \tag{C.12}$$

as can be shown from (C.10). Also, the determinant of an inverse matrix is given by

$$\left|\mathbf{A}^{-1}\right| = \frac{1}{|\mathbf{A}|} \tag{C.13}$$

which can be shown by taking the determinant of (C.2) and applying (C.12).

If $\mathbf{A}$ and $\mathbf{B}$ are matrices of size $N \times M$, then

$$\left|\mathbf{I}_N + \mathbf{AB}^{\mathrm{T}}\right| = \left|\mathbf{I}_M + \mathbf{A}^{\mathrm{T}}\mathbf{B}\right|. \tag{C.14}$$

A useful special case is

$$\left|\mathbf{I}_N + \mathbf{ab}^{\mathrm{T}}\right| = 1 + \mathbf{a}^{\mathrm{T}}\mathbf{b} \tag{C.15}$$

where $\mathbf{a}$ and $\mathbf{b}$ are N-dimensional column vectors.

Matrix Derivatives

Sometimes we need to consider derivatives of vectors and matrices with respect to scalars. The derivative of a vector $\mathbf{a}$ with respect to a scalar x is itself a vector whose components are given by

$$\left(\frac{\partial \mathbf{a}}{\partial x}\right)_i = \frac{\partial a_i}{\partial x} \tag{C.16}$$

with an analogous definition for the derivative of a matrix. Derivatives with respect to vectors and matrices can also be defined, for instance

$$\left(\frac{\partial x}{\partial \mathbf{a}}\right)_i = \frac{\partial x}{\partial a_i} \tag{C.17}$$

and similarly

$$\left(\frac{\partial \mathbf{a}}{\partial \mathbf{b}}\right)_{ij} = \frac{\partial a_i}{\partial b_j}. \tag{C.18}$$

The following is easily proven by writing out the components

$$\frac{\partial}{\partial \mathbf{x}}\left(\mathbf{x}^{\mathrm{T}}\mathbf{a}\right) = \frac{\partial}{\partial \mathbf{x}}\left(\mathbf{a}^{\mathrm{T}}\mathbf{x}\right) = \mathbf{a}. \tag{C.19}$$

Similarly

$$\frac{\partial}{\partial x}(\mathbf{AB}) = \frac{\partial \mathbf{A}}{\partial x}\mathbf{B} + \mathbf{A}\frac{\partial \mathbf{B}}{\partial x}. \tag{C.20}$$

The derivative of the inverse of a matrix can be expressed as

$$\frac{\partial}{\partial x}\left(\mathbf{A}^{-1}\right) = -\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x}\mathbf{A}^{-1} \tag{C.21}$$

as can be shown by differentiating the equation $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$ using (C.20) and then right multiplying by $\mathbf{A}^{-1}$. Also

$$\frac{\partial}{\partial x}\ln|\mathbf{A}| = \mathrm{Tr}\left(\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x}\right) \tag{C.22}$$

which we shall prove later. If we choose x to be one of the elements of $\mathbf{A}$, we have

$$\frac{\partial}{\partial A_{ij}}\mathrm{Tr}\left(\mathbf{AB}\right) = B_{ji} \tag{C.23}$$

as can be seen by writing out the matrices using index notation. We can write this result more compactly in the form

$$\frac{\partial}{\partial \mathbf{A}}\mathrm{Tr}\left(\mathbf{AB}\right) = \mathbf{B}^{\mathrm{T}}. \tag{C.24}$$

With this notation, we have the following properties

$$\frac{\partial}{\partial \mathbf{A}}\mathrm{Tr}\left(\mathbf{A}^{\mathrm{T}}\mathbf{B}\right) = \mathbf{B} \tag{C.25}$$

$$\frac{\partial}{\partial \mathbf{A}}\mathrm{Tr}(\mathbf{A}) = \mathbf{I} \tag{C.26}$$

$$\frac{\partial}{\partial \mathbf{A}}\mathrm{Tr}(\mathbf{ABA}^{\mathrm{T}}) = \mathbf{A}(\mathbf{B} + \mathbf{B}^{\mathrm{T}}) \tag{C.27}$$

which can again be proven by writing out the matrix indices. We also have

$$\frac{\partial}{\partial \mathbf{A}}\ln|\mathbf{A}| = \left(\mathbf{A}^{-1}\right)^{\mathrm{T}} \tag{C.28}$$

which follows from (C.22) and (C.26).

Eigenvector Equation

For a square matrix $\mathbf{A}$ of size $M \times M$, the eigenvector equation is defined by

$$\mathbf{A}\mathbf{u}_i = \lambda_i \mathbf{u}_i \tag{C.29}$$

for $i = 1, \ldots, M$, where $\mathbf{u}_i$ is an *eigenvector* and λ_i is the corresponding *eigenvalue*. This can be viewed as a set of M simultaneous homogeneous linear equations, and the condition for a solution is that

$$|\mathbf{A} - \lambda_i \mathbf{I}| = 0 \tag{C.30}$$

which is known as the *characteristic equation*. Because this is a polynomial of order M in λ_i, it must have M solutions (though these need not all be distinct). The rank of $\mathbf{A}$ is equal to the number of nonzero eigenvalues.

Of particular interest are symmetric matrices, which arise as covariance matrices, kernel matrices, and Hessians. Symmetric matrices have the property that $A_{ij} = A_{ji}$, or equivalently $\mathbf{A}^{\mathrm{T}} = \mathbf{A}$. The inverse of a symmetric matrix is also symmetric, as can be seen by taking the transpose of $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$ and using $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$ together with the symmetry of $\mathbf{I}$.

In general, the eigenvalues of a matrix are complex numbers, but for symmetric matrices the eigenvalues λ_i are real. This can be seen by first left multiplying (C.29) by $(\mathbf{u}_i^\star)^{\mathrm{T}}$, where $\star$ denotes the complex conjugate, to give

$$(\mathbf{u}_i^\star)^{\mathrm{T}} \mathbf{A}\mathbf{u}_i = \lambda_i (\mathbf{u}_i^\star)^{\mathrm{T}} \mathbf{u}_i. \tag{C.31}$$

Next we take the complex conjugate of (C.29) and left multiply by $\mathbf{u}_i^{\mathrm{T}}$ to give

$$\mathbf{u}_i^{\mathrm{T}} \mathbf{A}\mathbf{u}_i^\star = \lambda_i^\star \mathbf{u}_i^{\mathrm{T}} \mathbf{u}_i^\star. \tag{C.32}$$

where we have used $\mathbf{A}^\star = \mathbf{A}$ because we consider only real matrices $\mathbf{A}$. Taking the transpose of the second of these equations, and using $\mathbf{A}^{\mathrm{T}} = \mathbf{A}$, we see that the left-hand sides of the two equations are equal, and hence that $\lambda_i^\star = \lambda_i$ and so λ_i must be real.

The eigenvectors $\mathbf{u}_i$ of a real symmetric matrix can be chosen to be orthonormal (i.e., orthogonal and of unit length) so that

$$\mathbf{u}_i^{\mathrm{T}} \mathbf{u}_j = I_{ij} \tag{C.33}$$

where I_{ij} are the elements of the identity matrix $\mathbf{I}$. To show this, we first left multiply (C.29) by $\mathbf{u}_j^{\mathrm{T}}$ to give

$$\mathbf{u}_j^{\mathrm{T}} \mathbf{A}\mathbf{u}_i = \lambda_i \mathbf{u}_j^{\mathrm{T}} \mathbf{u}_i \tag{C.34}$$

and hence, by exchange of indices, we have

$$\mathbf{u}_i^{\mathrm{T}} \mathbf{A}\mathbf{u}_j = \lambda_j \mathbf{u}_i^{\mathrm{T}} \mathbf{u}_j. \tag{C.35}$$

We now take the transpose of the second equation and make use of the symmetry property $\mathbf{A}^{\mathrm{T}} = \mathbf{A}$, and then subtract the two equations to give

$$(\lambda_i - \lambda_j)\, \mathbf{u}_i^{\mathrm{T}} \mathbf{u}_j = 0. \tag{C.36}$$

Hence, for $\lambda_i \neq \lambda_j$, we have $\mathbf{u}_i^{\mathrm{T}} \mathbf{u}_j = 0$, and hence $\mathbf{u}_i$ and $\mathbf{u}_j$ are orthogonal. If the two eigenvalues are equal, then any linear combination $\alpha\mathbf{u}_i + \beta\mathbf{u}_j$ is also an eigenvector with the same eigenvalue, so we can select one linear combination arbitrarily,

and then choose the second to be orthogonal to the first (it can be shown that the degenerate eigenvectors are never linearly dependent). Hence the eigenvectors can be chosen to be orthogonal, and by normalizing can be set to unit length. Because there are M eigenvalues, the corresponding M orthogonal eigenvectors form a complete set and so any M-dimensional vector can be expressed as a linear combination of the eigenvectors.

We can take the eigenvectors $\mathbf{u}_i$ to be the columns of an $M \times M$ matrix $\mathbf{U}$, which from orthonormality satisfies

$$\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}. \tag{C.37}$$

Such a matrix is said to be *orthogonal*. Interestingly, the rows of this matrix are also orthogonal, so that $\mathbf{U}\mathbf{U}^{\mathrm{T}} = \mathbf{I}$. To show this, note that (C.37) implies $\mathbf{U}^{\mathrm{T}}\mathbf{U}\mathbf{U}^{-1} = \mathbf{U}^{-1} = \mathbf{U}^{\mathrm{T}}$ and so $\mathbf{U}\mathbf{U}^{-1} = \mathbf{U}\mathbf{U}^{\mathrm{T}} = \mathbf{I}$. Using (C.12), it also follows that $|\mathbf{U}| = 1$.

The eigenvector equation (C.29) can be expressed in terms of $\mathbf{U}$ in the form

$$\mathbf{A}\mathbf{U} = \mathbf{U}\boldsymbol{\Lambda} \tag{C.38}$$

where $\boldsymbol{\Lambda}$ is an $M \times M$ diagonal matrix whose diagonal elements are given by the eigenvalues λ_i.

If we consider a column vector $\mathbf{x}$ that is transformed by an orthogonal matrix $\mathbf{U}$ to give a new vector

$$\widetilde{\mathbf{x}} = \mathbf{U}\mathbf{x} \tag{C.39}$$

then the length of the vector is preserved because

$$\widetilde{\mathbf{x}}^{\mathrm{T}}\widetilde{\mathbf{x}} = \mathbf{x}^{\mathrm{T}}\mathbf{U}^{\mathrm{T}}\mathbf{U}\mathbf{x} = \mathbf{x}^{\mathrm{T}}\mathbf{x} \tag{C.40}$$

and similarly the angle between any two such vectors is preserved because

$$\widetilde{\mathbf{x}}^{\mathrm{T}}\widetilde{\mathbf{y}} = \mathbf{x}^{\mathrm{T}}\mathbf{U}^{\mathrm{T}}\mathbf{U}\mathbf{y} = \mathbf{x}^{\mathrm{T}}\mathbf{y}. \tag{C.41}$$

Thus, multiplication by $\mathbf{U}$ can be interpreted as a rigid rotation of the coordinate system.

From (C.38), it follows that

$$\mathbf{U}^{\mathrm{T}}\mathbf{A}\mathbf{U} = \boldsymbol{\Lambda} \tag{C.42}$$

and because $\boldsymbol{\Lambda}$ is a diagonal matrix, we say that the matrix $\mathbf{A}$ is *diagonalized* by the matrix $\mathbf{U}$. If we left multiply by $\mathbf{U}$ and right multiply by $\mathbf{U}^{\mathrm{T}}$, we obtain

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathrm{T}} \tag{C.43}$$

Taking the inverse of this equation, and using (C.3) together with $\mathbf{U}^{-1} = \mathbf{U}^{\mathrm{T}}$, we have

$$\mathbf{A}^{-1} = \mathbf{U}\boldsymbol{\Lambda}^{-1}\mathbf{U}^{\mathrm{T}}. \tag{C.44}$$

These last two equations can also be written in the form

$$\mathbf{A} = \sum_{i=1}^{M} \lambda_i \mathbf{u}_i \mathbf{u}_i^{\mathrm{T}} \tag{C.45}$$

$$\mathbf{A}^{-1} = \sum_{i=1}^{M} \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^{\mathrm{T}}. \tag{C.46}$$

If we take the determinant of (C.43), and use (C.12), we obtain

$$|\mathbf{A}| = \prod_{i=1}^{M} \lambda_i. \tag{C.47}$$

Similarly, taking the trace of (C.43), and using the cyclic property (C.8) of the trace operator together with $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}$, we have

$$\mathrm{Tr}(\mathbf{A}) = \sum_{i=1}^{M} \lambda_i. \tag{C.48}$$

We leave it as an exercise for the reader to verify (C.22) by making use of the results (C.33), (C.45), (C.46), and (C.47).

A matrix $\mathbf{A}$ is said to be *positive definite*, denoted by $\mathbf{A} \succ 0$, if $\mathbf{w}^{\mathrm{T}}\mathbf{A}\mathbf{w} > 0$ for all values of the vector $\mathbf{w}$. Equivalently, a positive definite matrix has $\lambda_i > 0$ for all of its eigenvalues (as can be seen by setting $\mathbf{w}$ to each of the eigenvectors in turn, and by noting that an arbitrary vector can be expanded as a linear combination of the eigenvectors). Note that positive definite is not the same as all the elements being positive. For example, the matrix

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \tag{C.49}$$

has eigenvalues $\lambda_1 \simeq 5.37$ and $\lambda_2 \simeq -0.37$. A matrix is said to be *positive semidefinite* if $\mathbf{w}^{\mathrm{T}}\mathbf{A}\mathbf{w} \geqslant 0$ holds for all values of $\mathbf{w}$, which is denoted $\mathbf{A} \succeq 0$, and is equivalent to $\lambda_i \geqslant 0$.

Appendix D. Calculus of Variations

We can think of a function $y(x)$ as being an operator that, for any input value x, returns an output value y. In the same way, we can define a *functional* $F[y]$ to be an operator that takes a function $y(x)$ and returns an output value F. An example of a functional is the length of a curve drawn in a two-dimensional plane in which the path of the curve is defined in terms of a function. In the context of machine learning, a widely used functional is the entropy $\mathrm{H}[x]$ for a continuous variable x because, for any choice of probability density function $p(x)$, it returns a scalar value representing the entropy of x under that density. Thus the entropy of $p(x)$ could equally well have been written as $\mathrm{H}[p]$.

A common problem in conventional calculus is to find a value of x that maximizes (or minimizes) a function $y(x)$. Similarly, in the calculus of variations we seek a function $y(x)$ that maximizes (or minimizes) a functional $F[y]$. That is, of all possible functions $y(x)$, we wish to find the particular function for which the functional $F[y]$ is a maximum (or minimum). The calculus of variations can be used, for instance, to show that the shortest path between two points is a straight line or that the maximum entropy distribution is a Gaussian.

If we weren't familiar with the rules of ordinary calculus, we could evaluate a conventional derivative $\mathrm{d}y/\mathrm{d}x$ by making a small change ϵ to the variable x and then expanding in powers of ϵ, so that

$$y(x+\epsilon) = y(x) + \frac{\mathrm{d}y}{\mathrm{d}x}\epsilon + O(\epsilon^2) \tag{D.1}$$

and finally taking the limit $\epsilon \to 0$. Similarly, for a function of several variables $y(x_1, \ldots, x_D)$, the corresponding partial derivatives are defined by

$$y(x_1+\epsilon_1, \ldots, x_D+\epsilon_D) = y(x_1, \ldots, x_D) + \sum_{i=1}^{D} \frac{\partial y}{\partial x_i}\epsilon_i + O(\epsilon^2). \tag{D.2}$$

The analogous definition of a functional derivative arises when we consider how much a functional $F[y]$ changes when we make a small change $\epsilon\eta(x)$ to the function

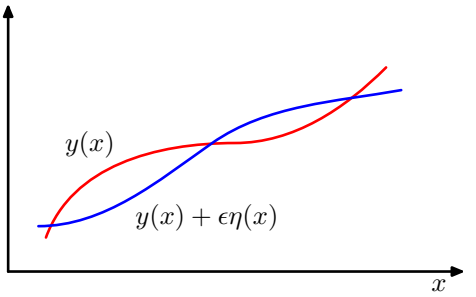


Figure D.1 A functional derivative can be defined by considering how the value of a functional $F[y]$ changes when the function $y(x)$ is changed to $y(x) + \epsilon\eta(x)$ where $\eta(x)$ is an arbitrary function of x.

$y(x)$, where $\eta(x)$ is an arbitrary function of x, as illustrated in Figure D.1. We denote the functional derivative of ~~$F[f]$~~ with respect to ~~$f(x)$~~ by $\delta F/\delta y(x)$, and define it by the following relation:

$$F[y(x) + \epsilon\eta(x)] = F[y(x)] + \epsilon \int \frac{\delta F}{\delta y(x)} \eta(x)\, \mathrm{d}x + O(\epsilon^2). \tag{D.3}$$

This can be seen as a natural extension of (D.2) in which $F[y]$ now depends on a continuous set of variables, namely the values of y at all points x. Requiring that the functional be stationary with respect to small variations in the function $y(x)$ gives

$$\int \frac{\delta F}{\delta y(x)} \eta(x)\, \mathrm{d}x = 0. \tag{D.4}$$

Because this must hold for an arbitrary choice of $\eta(x)$, it follows that the functional derivative must vanish. To see this, imagine choosing a perturbation $\eta(x)$ that is zero everywhere except in the neighbourhood of a point $\widehat{x}$, in which case the functional derivative must be zero at $x = \widehat{x}$. However, because this must be true for every choice of $\widehat{x}$, the functional derivative must vanish for all values of x.

Consider a functional that is defined by an integral over a function $G(y, y', x)$ that depends on both $y(x)$ and its derivative $y'(x)$ as well as having a direct dependence on x

$$F[y] = \int G\left(y(x), y'(x), x\right)\, \mathrm{d}x \tag{D.5}$$

where the value of $y(x)$ is assumed to be fixed at the boundary of the region of integration (which might be at infinity). If we now consider variations in the function $y(x)$, we obtain

$$F[y(x) + \epsilon\eta(x)] = F[y(x)] + \epsilon \int \left\{ \frac{\partial G}{\partial y}\eta(x) + \frac{\partial G}{\partial y'}\eta'(x) \right\} \mathrm{d}x + O(\epsilon^2). \tag{D.6}$$

We now have to cast this in the form (D.3). To do so, we integrate the second term by parts and make use of the fact that $\eta(x)$ must vanish at the boundary of the integral (because $y(x)$ is fixed at the boundary). This gives

$$F[y(x) + \epsilon\eta(x)] = F[y(x)] + \epsilon \int \left\{ \frac{\partial G}{\partial y} - \frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{\partial G}{\partial y'}\right) \right\} \eta(x)\, \mathrm{d}x + O(\epsilon^2) \tag{D.7}$$

from which we can read off the functional derivative by comparison with (D.3). Requiring that the functional derivative vanishes then gives

$$\frac{\partial G}{\partial y} - \frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{\partial G}{\partial y'}\right) = 0 \tag{D.8}$$

which are known as the *Euler-Lagrange* equations. For example, if

$$G = y(x)^2 + (y'(x))^2 \tag{D.9}$$

then the Euler-Lagrange equations take the form

$$y(x) - \frac{\mathrm{d}^2 y}{\mathrm{d}x^2} = 0. \tag{D.10}$$

This second order differential equation can be solved for $y(x)$ by making use of the boundary conditions on $y(x)$.

Often, we consider functionals defined by integrals whose integrands take the form $G(y, x)$ and that do not depend on the derivatives of $y(x)$. In this case, stationarity simply requires that $\partial G/\partial y(x) = 0$ for all values of x.

If we are optimizing a functional with respect to a probability distribution, then we need to maintain the normalization constraint on the probabilities. This is often most conveniently done using a Lagrange multiplier, which then allows an unconstrained optimization to be performed.

Appendix E

The extension of the above results to a multidimensional variable $\mathbf{x}$ is straightforward. For a more comprehensive discussion of the calculus of variations, see Sagan (1969).

Appendix E. Lagrange Multipliers

Lagrange multipliers, also sometimes called *undetermined multipliers*, are used to find the stationary points of a function of several variables subject to one or more constraints.

Consider the problem of finding the maximum of a function $f(x_1, x_2)$ subject to a constraint relating x_1 and x_2, which we write in the form

$$g(x_1, x_2) = 0. \tag{E.1}$$

One approach would be to solve the constraint equation (E.1) and thus express x_2 as a function of x_1 in the form $x_2 = h(x_1)$. This can then be substituted into $f(x_1, x_2)$ to give a function of x_1 alone of the form $f(x_1, h(x_1))$. The maximum with respect to x_1 could then be found by differentiation in the usual way, to give the stationary value $x_1^\star$, with the corresponding value of x_2 given by $x_2^\star = h(x_1^\star)$.

One problem with this approach is that it may be difficult to find an analytic solution of the constraint equation that allows x_2 to be expressed as an explicit function of x_1. Also, this approach treats x_1 and x_2 differently and so spoils the natural symmetry between these variables.

A more elegant, and often simpler, approach is based on the introduction of a parameter λ called a Lagrange multiplier. We shall motivate this technique from a geometrical perspective. Consider a D-dimensional variable $\mathbf{x}$ with components $x_1, \ldots, x_D$. The constraint equation $g(\mathbf{x}) = 0$ then represents a $(D-1)$-dimensional surface in $\mathbf{x}$-space as indicated in Figure E.1.

We first note that at any point on the constraint surface the gradient $\nabla g(\mathbf{x})$ of the constraint function will be orthogonal to the surface. To see this, consider a point $\mathbf{x}$ that lies on the constraint surface, and consider a nearby point $\mathbf{x} + \boldsymbol{\epsilon}$ that also lies on the surface. If we make a Taylor expansion around $\mathbf{x}$, we have

$$g(\mathbf{x} + \boldsymbol{\epsilon}) \simeq g(\mathbf{x}) + \boldsymbol{\epsilon}^{\mathrm{T}} \nabla g(\mathbf{x}). \tag{E.2}$$

Because both $\mathbf{x}$ and $\mathbf{x} + \boldsymbol{\epsilon}$ lie on the constraint surface, we have $g(\mathbf{x}) = g(\mathbf{x} + \boldsymbol{\epsilon})$ and hence $\boldsymbol{\epsilon}^{\mathrm{T}} \nabla g(\mathbf{x}) \simeq 0$. In the limit $\|\boldsymbol{\epsilon}\| \to 0$ we have $\boldsymbol{\epsilon}^{\mathrm{T}} \nabla g(\mathbf{x}) = 0$, and because $\boldsymbol{\epsilon}$ is

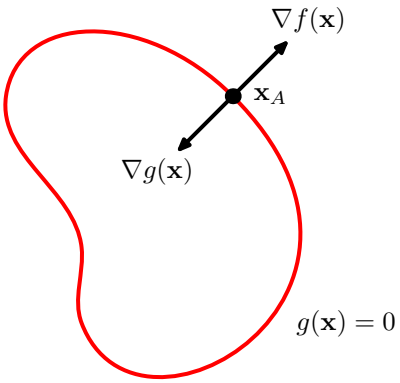


Figure E.1 A geometrical picture of the technique of Lagrange multipliers in which we seek to maximize a function $f(\mathbf{x})$, subject to the constraint $g(\mathbf{x}) = 0$. If $\mathbf{x}$ is D dimensional, the constraint $g(\mathbf{x}) = 0$ corresponds to a subspace of dimensionality $D - 1$, indicated by the red curve. The problem can be solved by optimizing the Lagrangian function $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$.

then parallel to the constraint surface $g(\mathbf{x}) = 0$, we see that the vector ∇g is normal to the surface.

Next we seek a point $\mathbf{x}^\star$ on the constraint surface such that $f(\mathbf{x})$ is maximized. Such a point must have the property that the vector $\nabla f(\mathbf{x})$ is also orthogonal to the constraint surface, as illustrated in Figure E.1, because otherwise we could increase the value of $f(\mathbf{x})$ by moving a short distance along the constraint surface. Thus ∇f and ∇g are parallel (or anti-parallel) vectors, and so there must exist a parameter λ such that

$$\nabla f + \lambda \nabla g = 0 \tag{E.3}$$

where $\lambda \neq 0$ is known as a *Lagrange multiplier*. Note that λ can have either sign.

At this point, it is convenient to introduce the *Lagrangian* function defined by

$$L(\mathbf{x}, \lambda) \equiv f(\mathbf{x}) + \lambda g(\mathbf{x}). \tag{E.4}$$

The constrained stationarity condition (E.3) is obtained by setting $\nabla_{\mathbf{x}} L = 0$. Furthermore, the condition $\partial L/\partial \lambda = 0$ leads to the constraint equation $g(\mathbf{x}) = 0$.

Thus to find the maximum of a function $f(\mathbf{x})$ subject to the constraint $g(\mathbf{x}) = 0$, we define the Lagrangian function given by (E.4) and we then find the stationary point of $L(\mathbf{x}, \lambda)$ with respect to both $\mathbf{x}$ and λ. For a D-dimensional vector $\mathbf{x}$, this gives $D + 1$ equations that determine both the stationary point $\mathbf{x}^\star$ and the value of λ. If we are only interested in $\mathbf{x}^\star$, then we can eliminate λ from the stationarity equations without needing to find its value (hence the term 'undetermined multiplier').

As a simple example, suppose we wish to find the stationary point of the function $f(x_1, x_2) = 1 - x_1^2 - x_2^2$ subject to the constraint $g(x_1, x_2) = x_1 + x_2 - 1 = 0$, as illustrated in Figure E.2. The corresponding Lagrangian function is given by

$$L(\mathbf{x}, \lambda) = 1 - x_1^2 - x_2^2 + \lambda(x_1 + x_2 - 1). \tag{E.5}$$

The conditions for this Lagrangian to be stationary with respect to x_1, x_2, and λ give the following coupled equations:

$$-2x_1 + \lambda = 0 \tag{E.6}$$

$$-2x_2 + \lambda = 0 \tag{E.7}$$

$$x_1 + x_2 - 1 = 0. \tag{E.8}$$

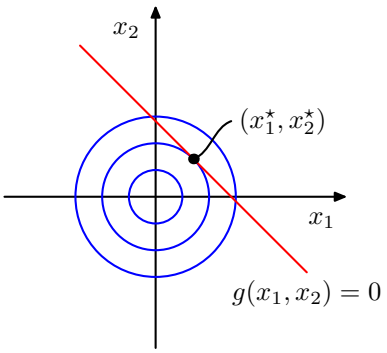


Figure E.2 A simple example of the use of Lagrange multipliers in which the aim is to maximize $f(x_1, x_2) = 1 - x_1^2 - x_2^2$ subject to the constraint $g(x_1, x_2) = 0$ where $g(x_1, x_2) = x_1 + x_2 - 1$. The circles show contours of the function $f(x_1, x_2)$, and the diagonal line shows the constraint surface $g(x_1, x_2) = 0$.

Solution of these equations then gives the stationary point as $(x_1^\star, x_2^\star) = (\frac{1}{2}, \frac{1}{2})$, and the corresponding value for the Lagrange multiplier is $\lambda = 1$.

So far, we have considered the problem of maximizing a function subject to an *equality constraint* of the form $g(\mathbf{x}) = 0$. We now consider the problem of maximizing $f(\mathbf{x})$ subject to an *inequality constraint* of the form $g(\mathbf{x}) \geqslant 0$, as illustrated in Figure E.3.

There are now two kinds of solution possible, according to whether the constrained stationary point lies in the region where $g(\mathbf{x}) > 0$, in which case the constraint is *inactive*, or whether it lies on the boundary $g(\mathbf{x}) = 0$, in which case the constraint is said to be *active*. In the former case, the function $g(\mathbf{x})$ plays no role and so the stationary condition is simply $\nabla f(\mathbf{x}) = 0$. This again corresponds to a stationary point of the Lagrange function (E.4) but this time with $\lambda = 0$. The latter case, where the solution lies on the boundary, is analogous to the equality constraint discussed previously and corresponds to a stationary point of the Lagrange function (E.4) with $\lambda \neq 0$. Now, however, the sign of the Lagrange multiplier is crucial, because the function $f(\mathbf{x})$ will only be at a maximum if its gradient is oriented away from the region $g(\mathbf{x}) > 0$, as illustrated in Figure E.3. We therefore have $\nabla f(\mathbf{x}) = -\lambda \nabla g(\mathbf{x})$ for some value of $\lambda > 0$.

For either of these two cases, the product $\lambda g(\mathbf{x}) = 0$. Thus the solution to the

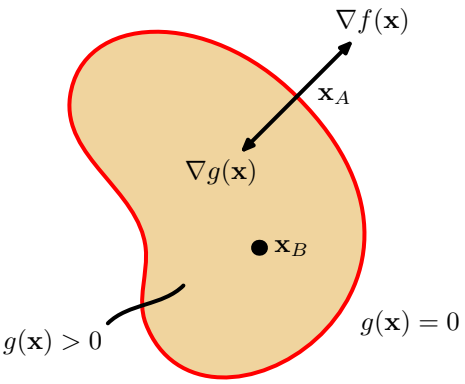


Figure E.3 Illustration of the problem of maximizing $f(\mathbf{x})$ subject to the inequality constraint $g(\mathbf{x}) \geqslant 0$.

problem of maximizing $f(\mathbf{x})$ subject to $g(\mathbf{x}) \geqslant 0$ is obtained by optimizing the Lagrange function (E.4) with respect to $\mathbf{x}$ and λ subject to the conditions

$$g(\mathbf{x}) \geqslant 0 \tag{E.9}$$
$$\lambda \geqslant 0 \tag{E.10}$$
$$\lambda g(\mathbf{x}) = 0 \tag{E.11}$$

These are known as the *Karush-Kuhn-Tucker* (KKT) conditions (Karush, 1939; Kuhn and Tucker, 1951).

Note that if we wish to minimize (rather than maximize) the function $f(\mathbf{x})$ subject to an inequality constraint $g(\mathbf{x}) \geqslant 0$, then we minimize the Lagrangian function $L(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda g(\mathbf{x})$ with respect to $\mathbf{x}$, again subject to $\lambda \geqslant 0$.

Finally, it is straightforward to extend the technique of Lagrange multipliers to the case of multiple equality and inequality constraints. Suppose we wish to maximize $f(\mathbf{x})$ subject to $g_j(\mathbf{x}) = 0$ for $j = 1, \ldots, J$, and $h_k(\mathbf{x}) \geqslant 0$ for $k = 1, \ldots, K$. We then introduce Lagrange multipliers $\{\lambda_j\}$ and $\{\mu_k\}$, and then optimize the Lagrangian function given by

$$L(\mathbf{x}, \{\lambda_j\}, \{\mu_k\}) = f(\mathbf{x}) + \sum_{j=1}^{J} \lambda_j g_j(\mathbf{x}) + \sum_{k=1}^{K} \mu_k h_k(\mathbf{x}) \tag{E.12}$$

subject to $\mu_k \geqslant 0$ and $\mu_k h_k(\mathbf{x}) = 0$ for $k = 1, \ldots, K$. Extensions to constrained functional derivatives are similarly straightforward. For a more detailed discussion of the technique of Lagrange multipliers, see Nocedal and Wright (1999).

Appendix D

References

Abramowitz, M. and I. A. Stegun (1965). *Handbook of Mathematical Functions*. Dover.

Adler, S. L. (1981). Over-relaxation method for the Monte Carlo evaluation of the partition function for multiquadratic actions. *Physical Review D* **23**, 2901–2904.

Ahn, J. H. and J. H. Oh (2003). A constrained EM algorithm for principal component analysis. *Neural Computation* **15**(1), 57–65.

Aizerman, M. A., E. M. Braverman, and L. I. Rozonoer (1964). The probability problem of pattern recognition learning and the method of potential functions. *Automation and Remote Control* **25**, 1175–1190.

Akaike, H. (1974). A new look at statistical model identification. *IEEE Transactions on Automatic Control* **19**, 716–723.

Ali, S. M. and S. D. Silvey (1966). A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society, B* **28**(1), 131–142.

Allwein, E. L., R. E. Schapire, and Y. Singer (2000). Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research* **1**, 113–141.

Amari, S. (1985). *Differential-Geometrical Methods in Statistics*. Springer.

Amari, S., A. Cichocki, and H. H. Yang (1996). A new learning algorithm for blind signal separation. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems*, Volume 8, pp. 757–763. MIT Press.

Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation* **10**, 251–276.

Anderson, J. A. and E. Rosenfeld (Eds.) (1988). *Neurocomputing: Foundations of Research*. MIT Press.

Anderson, T. W. (1963). Asymptotic theory for principal component analysis. *Annals of Mathematical Statistics* **34**, 122–148.

Andrieu, C., N. de Freitas, A. Doucet, and M. I. Jordan (2003). An introduction to MCMC for machine learning. *Machine Learning* **50**, 5–43.

Anthony, M. and N. Biggs (1992). *An Introduction to Computational Learning Theory*. Cambridge University Press.

Attias, H. (1999a). Independent factor analysis. *Neural Computation* **11**(4), 803–851.

Attias, H. (1999b). Inferring parameters and structure of latent variable models by variational Bayes. In K. B. Laskey and H. Prade (Eds.),

Uncertainty in Artificial Intelligence: Proceedings of the Fifth Conference, pp. 21–30. Morgan Kaufmann.

Bach, F. R. and M. I. Jordan (2002). Kernel independent component analysis. *Journal of Machine Learning Research* **3**, 1–48.

Bakir, G. H., J. Weston, and B. Schölkopf (2004). Learning to find pre-images. In S. Thrun, L. K. Saul, and B. Schölkopf (Eds.), *Advances in Neural Information Processing Systems*, Volume 16, pp. 449–456. MIT Press.

Baldi, P. and S. Brunak (2001). *Bioinformatics: The Machine Learning Approach* (Second ed.). MIT Press.

Baldi, P. and K. Hornik (1989). Neural networks and principal component analysis: learning from examples without local minima. *Neural Networks* **2**(1), 53–58.

Barber, D. and C. M. Bishop (1997). Bayesian model comparison by Monte Carlo chaining. In M. Mozer, M. Jordan, and T. Petsche (Eds.), *Advances in Neural Information Processing Systems*, Volume 9, pp. 333–339. MIT Press.

Barber, D. and C. M. Bishop (1998a). Ensemble learning for multi-layer networks. In M. I. Jordan, K. J. Kearns, and S. A. Solla (Eds.), *Advances in Neural Information Processing Systems*, Volume 10, pp. 395–401.

Barber, D. and C. M. Bishop (1998b). Ensemble learning in Bayesian neural networks. In C. M. Bishop (Ed.), *Generalization in Neural Networks and Machine Learning*, pp. 215–237. Springer.

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

Basilevsky, A. (1994). *Statistical Factor Analysis and Related Methods: Theory and Applications*. Wiley.

Bather, J. (2000). *Decision Theory: An Introduction to Dynamic Programming and Sequential Decisions*. Wiley.

Baudat, G. and F. Anouar (2000). Generalized discriminant analysis using a kernel approach. *Neural Computation* **12**(10), 2385–2404.

Baum, L. E. (1972). An inequality and associated maximization technique in statistical estimation of probabilistic functions of Markov processes. *Inequalities* **3**, 1–8.

Becker, S. and Y. Le Cun (1989). Improving the convergence of back-propagation learning with second order methods. In D. Touretzky, G. E. Hinton, and T. J. Sejnowski (Eds.), *Proceedings of the 1988 Connectionist Models Summer School*, pp. 29–37. Morgan Kaufmann.

Bell, A. J. and T. J. Sejnowski (1995). An information maximization approach to blind separation and blind deconvolution. *Neural Computation* **7**(6), 1129–1159.

Bellman, R. (1961). *Adaptive Control Processes: A Guided Tour*. Princeton University Press.

Bengio, Y. and P. Frasconi (1995). An input output HMM architecture. In G. Tesauro, D. S. Touretzky, and T. K. Leen (Eds.), *Advances in Neural Information Processing Systems*, Volume 7, pp. 427–434. MIT Press.

Bennett, K. P. (1992). Robust linear programming discrimination of two linearly separable sets. *Optimization Methods and Software* **1**, 23–34.

Berger, J. O. (1985). *Statistical Decision Theory and Bayesian Analysis* (Second ed.). Springer.

Bernardo, J. M. and A. F. M. Smith (1994). *Bayesian Theory*. Wiley.

Berrou, C., A. Glavieux, and P. Thitimajshima (1993). Near Shannon limit error-correcting coding and decoding: Turbo-codes (1). In *Proceedings ICC'93*, pp. 1064–1070.

Besag, J. (1974). On spatio-temporal models and Markov fields. In *Transactions of the 7th Prague Conference on Information Theory, Statistical Decision Functions and Random Processes*, pp. 47–75. Academia.

Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society* **B-48**, 259–302.

Besag, J., P. J. Green, D. Hidgon, and K. Megersen (1995). Bayesian computation and stochastic systems. *Statistical Science* **10**(1), 3–66.

Bishop, C. M. (1991). A fast procedure for retraining the multilayer perceptron. *International Journal of Neural Systems* **2**(3), 229–236.

Bishop, C. M. (1992). Exact calculation of the Hessian matrix for the multilayer perceptron. *Neural Computation* **4**(4), 494–501.

Bishop, C. M. (1993). Curvature-driven smoothing: a learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks* **4**(5), 882–884.

Bishop, C. M. (1994). Novelty detection and neural network validation. *IEE Proceedings: Vision, Image and Signal Processing* **141**(4), 217–222. Special issue on applications of neural networks.

Bishop, C. M. (1995a). *Neural Networks for Pattern Recognition*. Oxford University Press.

Bishop, C. M. (1995b). Training with noise is equivalent to Tikhonov regularization. *Neural Computation* **7**(1), 108–116.

Bishop, C. M. (1999a). Bayesian PCA. In M. S. Kearns, S. A. Solla, and D. A. Cohn (Eds.), *Advances in Neural Information Processing Systems*, Volume 11, pp. 382–388. MIT Press.

Bishop, C. M. (1999b). Variational principal components. In *Proceedings Ninth International Conference on Artificial Neural Networks, ICANN'99*, Volume 1, pp. 509–514. IEE.

Bishop, C. M. and G. D. James (1993). Analysis of multiphase flows using dual-energy gamma densitometry and neural networks. *Nuclear Instruments and Methods in Physics Research* **A327**, 580–593.

Bishop, C. M. and I. T. Nabney (1996). Modelling conditional probability distributions for periodic variables. *Neural Computation* **8**(5), 1123–1133.

Bishop, C. M. and I. T. Nabney (2008). ~~*Pattern Recognition and Machine Learning: A Matlab Companion*. Springer~~. In preparation.

Bishop, C. M., D. Spiegelhalter, and J. Winn (2003). VIBES: A variational inference engine for Bayesian networks. In S. Becker, S. Thrun, and K. Obermeyer (Eds.), *Advances in Neural Information Processing Systems*, Volume 15, pp. 793–800. MIT Press.

Bishop, C. M. and M. Svensén (2003). Bayesian hierarchical mixtures of experts. In U. Kjaerulff and C. Meek (Eds.), *Proceedings Nineteenth Conference on Uncertainty in Artificial Intelligence*, pp. 57–64. Morgan Kaufmann.

Bishop, C. M., M. Svensén, and G. E. Hinton (2004). Distinguishing text from graphics in online handwritten ink. In F. Kimura and H. Fujisawa (Eds.), *Proceedings Ninth International Workshop on Frontiers in Handwriting Recognition, IWFHR-9*, Tokyo, Japan, pp. 142–147.

Bishop, C. M., M. Svensén, and C. K. I. Williams (1996). EM optimization of latent variable density models. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems*, Volume 8, pp. 465–471. MIT Press.

Bishop, C. M., M. Svensén, and C. K. I. Williams (1997a). GTM: a principled alternative to the Self-Organizing Map. In M. C. Mozer, M. I. Jordan, and T. Petche (Eds.), *Advances in Neural Information Processing Systems*, Volume 9, pp. 354–360. MIT Press.

Bishop, C. M., M. Svensén, and C. K. I. Williams (1997b). Magnification factors for the GTM algorithm. In *Proceedings IEE Fifth International Conference on Artificial Neural Networks, Cambridge, U.K.*, pp. 64–69. Institute of Electrical Engineers.

Bishop, C. M., M. Svensén, and C. K. I. Williams (1998a). Developments of the Generative Topographic Mapping. *Neurocomputing* **21**, 203–224.

Bishop, C. M., M. Svensén, and C. K. I. Williams (1998b). GTM: the Generative Topographic Mapping. *Neural Computation* **10**(1), 215–234.

Bishop, C. M. and M. E. Tipping (1998). A hierarchical latent variable model for data visualization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20**(3), 281–293.

Optimization Algorithms for Machine Learning

Bishop, C. M. and J. Winn (2000). Non-linear Bayesian image modelling. In *Proceedings Sixth European Conference on Computer Vision, Dublin*, Volume 1, pp. 3–17. Springer.

Blei, D. M., M. I. Jordan, and A. Y. Ng (2003). Hierarchical Bayesian models for applications in information retrieval. In J. M. B, *et al.* (Ed.), *Bayesian Statistics, 7*, pp. 25–43. Oxford University Press.

Block, H. D. (1962). The perceptron: a model for brain functioning. *Reviews of Modern Physics* **34**(1), 123–135. Reprinted in Anderson and Rosenfeld (1988).

Blum, J. A. (1965). Multidimensional stochastic approximation methods. *Annals of Mathematical Statistics* **25**, 737–744.

Bodlaender, H. (1993). A tourist guide through treewidth. *Acta Cybernetica* **11**, 1–21.

Boser, B. E., I. M. Guyon, and V. N. Vapnik (1992). A training algorithm for optimal margin classifiers. In D. Haussler (Ed.), *Proceedings Fifth Annual Workshop on Computational Learning Theory (COLT)*, pp. 144–152. ACM.

Bourlard, H. and Y. Kamp (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics* **59**, 291–294.

Box, G. E. P., G. M. Jenkins, and G. C. Reinsel (1994). *Time Series Analysis*. Prentice Hall.

Box, G. E. P. and G. C. Tao (1973). *Bayesian Inference in Statistical Analysis*. Wiley.

Boyd, S. and L. Vandenberghe (2004). *Convex Optimization*. Cambridge University Press.

Boyen, X. and D. Koller (1998). Tractable inference for complex stochastic processes. In G. F. Cooper and S. Moral (Eds.), *Proceedings 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 33–42. Morgan Kaufmann.

Boykov, Y., O. Veksler, and R. Zabih (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23**(11), 1222–1239.

Breiman, L. (1996). Bagging predictors. *Machine Learning* **26**, 123–140.

Breiman, L., J. H. Friedman, R. A. Olshen, and P. J. Stone (1984). *Classification and Regression Trees*. Wadsworth.

Brooks, S. P. (1998). Markov chain Monte Carlo method and its application. *The Statistician* **47**(1), 69–100.

Broomhead, D. S. and D. Lowe (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems* **2**, 321–355.

Buntine, W. and A. Weigend (1991). Bayesian backpropagation. *Complex Systems* **5**, 603–643.

Buntine, W. L. and A. S. Weigend (1993). Computing second derivatives in feed-forward networks: a review. *IEEE Transactions on Neural Networks* **5**(3), 480–488.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining* **2**(2), 121–167.

Cardoso, J.-F. (1998). Blind signal separation: statistical principles. *Proceedings of the IEEE* **9**(10), 2009–2025.

Casella, G. and R. L. Berger (2002). *Statistical Inference* (Second ed.). Duxbury.

Castillo, E., J. M. Gutiérrez, and A. S. Hadi (1997). *Expert Systems and Probabilistic Network Models*. Springer.

Chan, K., T. Lee, and T. J. Sejnowski (2003). Variational Bayesian learning of ICA with missing data. *Neural Computation* **15**(8), 1991–2011.

Chen, A. M., H. Lu, and R. Hecht-Nielsen (1993). On the geometry of feedforward neural network error surfaces. *Neural Computation* **5**(6), 910–927.

Chen, M. H., Q. M. Shao, and J. G. Ibrahim (Eds.) (2001). *Monte Carlo Methods for Bayesian Computation*. Springer.

Chen, S., C. F. N. Cowan, and P. M. Grant (1991). Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks* **2**(2), 302–309.

Choudrey, R. A. and S. J. Roberts (2003). Variational mixture of Bayesian independent component analyzers. *Neural Computation* **15**(1), 213–252.

Clifford, P. (1990). Markov random fields in statistics. In G. R. Grimmett and D. J. A. Welsh (Eds.), *Disorder in Physical Systems. A Volume in Honour of John M. Hammersley*, pp. 19–32. Oxford University Press.

Collins, M., S. Dasgupta, and R. E. Schapire (2002). A generalization of principal component analysis to the exponential family. In T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems*, Volume 14, pp. 617–624. MIT Press.

Comon, P., C. Jutten, and J. Herault (1991). Blind source separation, 2: problems statement. *Signal Processing* **24**(1), 11–20.

Corduneanu, A. and C. M. Bishop (2001). Variational Bayesian model selection for mixture distributions. In T. Richardson and T. Jaakkola (Eds.), *Proceedings Eighth International Conference on Artificial Intelligence and Statistics*, pp. 27–34. Morgan Kaufmann.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein (2001). *Introduction to Algorithms* (Second ed.). MIT Press.

Cortes, C. and V. N. Vapnik (1995). Support vector networks. *Machine Learning* **20**, 273–297.

Cotter, N. E. (1990). The Stone-Weierstrass theorem and its application to neural networks. *IEEE Transactions on Neural Networks* **1**(4), 290–295.

Cover, T. and P. Hart (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* **IT-11**, 21–27.

Cover, T. M. and J. A. Thomas (1991). *Elements of Information Theory*. Wiley.

Cowell, R. G., A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter (1999). *Probabilistic Networks and Expert Systems*. Springer.

Cox, R. T. (1946). Probability, frequency and reasonable expectation. *American Journal of Physics* **14**(1), 1–13.

Cox, T. F. and M. A. A. Cox (2000). *Multidimensional Scaling* (Second ed.). Chapman and Hall.

Cressie, N. (1993). *Statistics for Spatial Data*. Wiley.

Cristianini, N. and J. Shawe-Taylor (2000). *Support vector machines and other kernel-based learning methods*. Cambridge University Press.

Csató, L. and M. Opper (2002). Sparse on-line Gaussian processes. *Neural Computation* **14**(3), 641–668.

Csiszàr, I. and G. Tusnàdy (1984). Information geometry and alternating minimization procedures. *Statistics and Decisions* **1**(1), 205–237.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems* **2**, 304–314.

Dawid, A. P. (1979). Conditional independence in statistical theory (with discussion). *Journal of the Royal Statistical Society, Series B* **4**, 1–31.

Dawid, A. P. (1980). Conditional independence for statistical operations. *Annals of Statistics* **8**, 598–617.

deFinetti, B. (1970). *Theory of Probability*. Wiley and Sons.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B* **39**(1), 1–38.

Denison, D. G. T., C. C. Holmes, B. K. Mallick, and A. F. M. Smith (2002). *Bayesian Methods for Nonlinear Classification and Regression*. Wiley.

Diaconis, P. and L. Saloff-Coste (1998). What do we know about the Metropolis algorithm? *Journal of Computer and System Sciences* **57**, 20–36.

Dietterich, T. G. and G. Bakiri (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research* **2**, 263–286.

Duane, S., A. D. Kennedy, B. J. Pendleton, and D. Roweth (1987). Hybrid Monte Carlo. *Physics Letters B* **195**(2), 216–222.

Duda, R. O. and P. E. Hart (1973). *Pattern Classification and Scene Analysis*. Wiley.

Duda, R. O., P. E. Hart, and D. G. Stork (2001). *Pattern Classification* (Second ed.). Wiley.

Durbin, R., S. Eddy, A. Krogh, and G. Mitchison (1998). *Biological Sequence Analysis*. Cambridge University Press.

Dybowski, R. and S. Roberts (2005). An anthology of probabilistic models for medical informatics. In D. Husmeier, R. Dybowski, and S. Roberts (Eds.), *Probabilistic Modeling in Bioinformatics and Medical Informatics*, pp. 297–349. Springer.

Efron, B. (1979). Bootstrap methods: another look at the jackknife. *Annals of Statistics* **7**, 1–26.

Elkan, C. (2003). Using the triangle inequality to accelerate k-means. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 147–153. AAAI.

Elliott, R. J., L. Aggoun, and J. B. Moore (1995). *Hidden Markov Models: Estimation and Control*. Springer.

Ephraim, Y., D. Malah, and B. H. Juang (1989). On the application of hidden Markov models for enhancing noisy speech. *IEEE Transactions on Acoustics, Speech and Signal Processing* **37**(12), 1846–1856.

Erwin, E., K. Obermayer, and K. Schulten (1992). Self-organizing maps: ordering, convergence properties and energy functions. *Biological Cybernetics* **67**, 47–55.

Everitt, B. S. (1984). *An Introduction to Latent Variable Models*. Chapman and Hall.

Faul, A. C. and M. E. Tipping (2002). Analysis of sparse Bayesian learning. In T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems*, Volume 14, pp. 383–389. MIT Press.

Feller, W. (1966). *An Introduction to Probability Theory and its Applications* (Second ed.), Volume 2. Wiley.

Feynman, R. P., R. B. Leighton, and M. Sands (1964). *The Feynman Lectures of Physics*, Volume Two. Addison-Wesley. Chapter 19.

Fletcher, R. (1987). *Practical Methods of Optimization* (Second ed.). Wiley.

Forsyth, D. A. and J. Ponce (2003). *Computer Vision: A Modern Approach*. Prentice Hall.

Freund, Y. and R. E. Schapire (1996). Experiments with a new boosting algorithm. In L. Saitta (Ed.), *Thirteenth International Conference on Machine Learning*, pp. 148–156. Morgan Kaufmann.

Frey, B. J. (1998). *Graphical Models for Machine Learning and Digital Communication*. MIT Press.

Frey, B. J. and D. J. C. MacKay (1998). A revolution: Belief propagation in graphs with cycles. In M. I. Jordan, M. J. Kearns, and S. A. Solla (Eds.), *Advances in Neural Information Processing Systems*, Volume 10. MIT Press.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics* **29**(5), 1189–1232.

Friedman, J. H., T. Hastie, and R. Tibshirani (2000). Additive logistic regression: a statistical view of boosting. *Annals of Statistics* **28**, 337–407.

Friedman, N. and D. Koller (2003). Being Bayesian about network structure: A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning* **50**, 95–126.

Frydenberg, M. (1990). The chain graph Markov property. *Scandinavian Journal of Statistics* **17**, 333–353.

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition* (Second ed.). Academic Press.

Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks* **2**(3), 183–192.

Fung, R. and K. C. Chang (1990). Weighting and integrating evidence for stochastic simulation in Bayesian networks. In P. P. Bonissone, M. Henrion, L. N. Kanal, and J. F. Lemmer (Eds.), *Uncertainty in Artificial Intelligence*, Volume 5, pp. 208–219. Elsevier.

Gallager, R. G. (1963). *Low-Density Parity-Check Codes*. MIT Press.

Gamerman, D. (1997). *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. Chapman and Hall.

Gelman, A., J. B. Carlin, H. S. Stern, and D. B. Rubin (2004). *Bayesian Data Analysis* (Second ed.). Chapman and Hall.

Geman, S. and D. Geman (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**(1), 721–741.

Ghahramani, Z. and M. J. Beal (2000). Variational inference for Bayesian mixtures of factor analyzers. In S. A. Solla, T. K. Leen, and K. R. Müller (Eds.), *Advances in Neural Information Processing Systems*, Volume 12, pp. 449–455. MIT Press.

Ghahramani, Z. and G. E. Hinton (1996a). The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, University of Toronto.

Ghahramani, Z. and G. E. Hinton (1996b). Parameter estimation for linear dynamical systems. Technical Report CRG-TR-96-2, University of Toronto.

Ghahramani, Z. and G. E. Hinton (1998). Variational learning for switching state-space models. *Neural Computation* **12**(4), 963–996.

Ghahramani, Z. and M. I. Jordan (1994). Supervised learning from incomplete data via an EM appproach. In J. D. Cowan, G. T. Tesauro, and J. Alspector (Eds.), *Advances in Neural Information Processing Systems*, Volume 6, pp. 120–127. Morgan Kaufmann.

Ghahramani, Z. and M. I. Jordan (1997). Factorial hidden Markov models. *Machine Learning* **29**, 245–275.

Gibbs, M. N. (1997). *Bayesian Gaussian processes for regression and classification*. Phd thesis, University of Cambridge.

Gibbs, M. N. and D. J. C. MacKay (2000). Variational Gaussian process classifiers. *IEEE Transactions on Neural Networks* **11**, 1458–1464.

Gilks, W. R. (1992). Derivative-free adaptive rejection sampling for Gibbs sampling. In J. Bernardo, J. Berger, A. P. Dawid, and A. F. M. Smith (Eds.), *Bayesian Statistics*, Volume 4. Oxford University Press.

Gilks, W. R., N. G. Best, and K. K. C. Tan (1995). Adaptive rejection Metropolis sampling. *Applied Statistics* **44**, 455–472.

Gilks, W. R., S. Richardson, and D. J. Spiegelhalter (Eds.) (1996). *Markov Chain Monte Carlo in Practice*. Chapman and Hall.

Gilks, W. R. and P. Wild (1992). Adaptive rejection sampling for Gibbs sampling. *Applied Statistics* **41**, 337–348.

Gill, P. E., W. Murray, and M. H. Wright (1981). *Practical Optimization*. Academic Press.

Goldberg, P. W., C. K. I. Williams, and C. M. Bishop (1998). Regression with input-dependent noise: A Gaussian process treatment. In *Advances in Neural Information Processing Systems*, Volume 10, pp. 493–499. MIT Press.

Golub, G. H. and C. F. Van Loan (1996). *Matrix Computations* (Third ed.). John Hopkins University Press.

Good, I. (1950). *Probability and the Weighing of Evidence*. Hafners.

Gordon, N. J., D. J. Salmond, and A. F. M. Smith (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings-F* **140**(2), 107–113.

Graepel, T. (2003). Solving noisy linear operator equations by Gaussian processes: Application to ordinary and partial differential equations. In *Proceedings of the Twentieth International Conference on Machine Learning*, pp. 234–241.

Greig, D., B. Porteous, and A. Seheult (1989). Exact maximum a-posteriori estimation for binary images. *Journal of the Royal Statistical Society, Series B* **51**(2), 271–279.

Gull, S. F. (1989). Developments in maximum entropy data analysis. In J. Skilling (Ed.), *Maximum Entropy and Bayesian Methods*, pp. 53–71. Kluwer.

Hassibi, B. and D. G. Stork (1993). Second order derivatives for network pruning: optimal brain surgeon. In S. J. Hanson, J. D. Cowan, and C. L. Giles (Eds.), *Advances in Neural Information Processing Systems*, Volume 5, pp. 164–171. Morgan Kaufmann.

Hastie, T. and W. Stuetzle (1989). Principal curves. *Journal of the American Statistical Association* **84**(106), 502–516.

Hastie, T., R. Tibshirani, and J. Friedman (2001). *The Elements of Statistical Learning*. Springer.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **57**, 97–109.

Hathaway, R. J. (1986). Another interpretation of the EM algorithm for mixture distributions. *Statistics and Probability Letters* **4**, 53–56.

Haussler, D. (1999). Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California, Santa Cruz, Computer Science Department.

Henrion, M. (1988). Propagation of uncertainty by logic sampling in Bayes' networks. In J. F. Lemmer and L. N. Kanal (Eds.), *Uncertainty in Artificial Intelligence*, Volume 2, pp. 149–164. North Holland.

Herbrich, R. (2002). *Learning Kernel Classifiers*. MIT Press.

Hertz, J., A. Krogh, and R. G. Palmer (1991). *Introduction to the Theory of Neural Computation*. Addison Wesley.

Hinton, G. E., P. Dayan, and M. Revow (1997). Modelling the manifolds of images of handwritten digits. *IEEE Transactions on Neural Networks* **8**(1), 65–74.

Hinton, G. E. and D. van Camp (1993). Keeping neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pp. 5–13. ACM.

Hinton, G. E., M. Welling, Y. W. Teh, and S. Osindero (2001). A new view of ICA. In *Proceedings of the International Conference on Independent Component Analysis and Blind Signal Separation*, Volume 3.

Hodgson, M. E. (1998). Reducing computational requirements of the minimum-distance classifier. *Remote Sensing of Environments* **25**, 117–128.

Hoerl, A. E. and R. Kennard (1970). Ridge regression: biased estimation for nonorthogonal problems. *Technometrics* **12**, 55–67.

Hofmann, T. (2000). Learning the similarity of documents: an information-geometric approach to document retrieval and classification. In S. A. Solla, T. K. Leen, and K. R. Müller (Eds.), *Advances in Neural Information Processing Systems*, Volume 12, pp. 914–920. MIT Press.

Hojen-Sorensen, P. A., O. Winther, and L. K. Hansen (2002). Mean field approaches to independent component analysis. *Neural Computation* **14**(4), 889–918.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks* **4**(2), 251–257.

Hornik, K., M. Stinchcombe, and H. White (1989). Multilayer feedforward networks are universal approximators. *Neural Networks* **2**(5), 359–366.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology* **24**, 417–441.

Hotelling, H. (1936). Relations between two sets of variables. *Biometrika* **28**, 321–377.

Hyvärinen, A. and E. Oja (1997). A fast fixed-point algorithm for independent component analysis. *Neural Computation* **9**(7), 1483–1492.

Isard, M. and A. Blake (1998). CONDENSATION – conditional density propagation for visual tracking. *International Journal of Computer Vision* **29**(1), 5–18.

Ito, Y. (1991). Representation of functions by superpositions of a step or sigmoid function and their applications to neural network theory. *Neural Networks* **4**(3), 385–394.

Jaakkola, T. and M. I. Jordan (2000). Bayesian parameter estimation via variational methods. *Statistics and Computing* **10**, 25–37.

Jaakkola, T. S. (2001). Tutorial on variational approximation methods. In M. Opper and D. Saad (Eds.), *Advances in Mean Field Methods*, pp. 129–159. MIT Press.

Jaakkola, T. S. and D. Haussler (1999). Exploiting generative models in discriminative classifiers. In M. S. Kearns, S. A. Solla, and D. A. Cohn (Eds.), *Advances in Neural Information Processing Systems*, Volume 11. MIT Press.

Jacobs, R. A., M. I. Jordan, S. J. Nowlan, and G. E. Hinton (1991). Adaptive mixtures of local experts. *Neural Computation* **3**(1), 79–87.

Jaynes, E. T. (2003). *Probability Theory: The Logic of Science*. Cambridge University Press.

Jebara, T. (2004). *Machine Learning: Discriminative and Generative*. Kluwer.

Jeffreys

Jeffries, H. (1946). An invariant form for the prior probability in estimation problems. *Pro. Roy. Soc. AA* **186**, 453–461.

Jelinek, F. (1997). *Statistical Methods for Speech Recognition*. MIT Press.

Jensen, C., A. Kong, and U. Kjaerulff (1995). Blocking gibbs sampling in very large probabilistic expert systems. *International Journal of Human Computer Studies. Special Issue on Real-World Applications of Uncertain Reasoning.* **42**, 647–666.

Jensen, F. V. (1996). *An Introduction to Bayesian Networks*. UCL Press.

Jerrum, M. and A. Sinclair (1996). The Markov chain Monte Carlo method: an approach to approximate counting and integration. In D. S. Hochbaum (Ed.), *Approximation Algorithms for NP-Hard Problems*. PWS Publishing.

Jolliffe, I. T. (2002). *Principal Component Analysis* (Second ed.). Springer.

Jordan, M. I. (1999). *Learning in Graphical Models*. MIT Press.

Jordan, M. I. (2007). *An Introduction to Probabilistic Graphical Models*. In preparation.

Jordan, M. I., Z. Ghahramani, T. S. Jaakkola, and L. K. Saul (1999). An introduction to variational methods for graphical models. In M. I. Jordan (Ed.), *Learning in Graphical Models*, pp. 105–162. MIT Press.

Jordan, M. I. and R. A. Jacobs (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation* **6**(2), 181–214.

Jutten, C. and J. Herault (1991). Blind separation of sources, 1: An adaptive algorithm based on neuromimetic architecture. *Signal Processing* **24**(1), 1–10.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the American Society for Mechanical Engineering, Series D, Journal of Basic Engineering* **82**, 35–45.

Kambhatla, N. and T. K. Leen (1997). Dimension reduction by local principal component analysis. *Neural Computation* **9**(7), 1493–1516.

Kanazawa, K., D. Koller, and S. Russel (1995). Stochastic simulation algorithms for dynamic probabilistic networks. In *Uncertainty in Artificial Intelligence*, Volume 11. Morgan Kaufmann.

Kapadia, S. (1998). *Discriminative Training of Hidden Markov Models*. Phd thesis, University of Cambridge, U.K.

Kapur, J. (1989). *Maximum entropy methods in science and engineering*. Wiley.

Karush, W. (1939). Minima of functions of several variables with inequalities as side constraints. Master's thesis, Department of Mathematics, University of Chicago.

Kass, R. E. and A. E. Raftery (1995). Bayes factors. *Journal of the American Statistical Association* **90**, 377–395.

Kearns, M. J. and U. V. Vazirani (1994). *An Introduction to Computational Learning Theory*. MIT Press.

Kindermann, R. and J. L. Snell (1980). *Markov Random Fields and Their Applications*. American Mathematical Society.

Kittler, J. and J. Föglein (1984). Contextual classification of multispectral pixel data. *Image and Vision Computing* **2**, 13–29.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics* **43**, 59–69.

Kohonen, T. (1995). *Self-Organizing Maps*. Springer.

Kolmogorov, V. and R. Zabih (2004). What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26**(2), 147–159.

Kreinovich, V. Y. (1991). Arbitrary nonlinearity is sufficient to represent all functions by neural networks: a theorem. *Neural Networks* **4**(3), 381–383.

Krogh, A., M. Brown, I. S. Mian, K. Sjölander, and D. Haussler (1994). Hidden Markov models in computational biology: Applications to protein modelling. *Journal of Molecular Biology* **235**, 1501–1531.

Kschischnang, F. R., B. J. Frey, and H. A. Loeliger (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory* **47**(2), 498–519.

Kuhn, H. W. and A. W. Tucker (1951). Nonlinear programming. In *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probabilities*, pp. 481–492. University of California Press.

Kullback, S. and R. A. Leibler (1951). On information and sufficiency. *Annals of Mathematical Statistics* **22**(1), 79–86.

Kùrková, V. and P. C. Kainen (1994). Functionally equivalent feed-forward neural networks. *Neural Computation* **6**(3), 543–558.

Kuss, M. and C. Rasmussen (2006). Assessing approximations for Gaussian process classification. In *Advances in Neural Information Processing Systems*, Number 18. MIT Press. in press.

Lasserre, J., C. M. Bishop, and T. Minka (2006). Principled hybrids of generative and discriminative models. In *Proceedings 2006 IEEE Conference on Computer Vision and Pattern Recognition, New York*.

Lauritzen, S. and N. Wermuth (1989). Graphical models for association between variables, some of which are qualitative some quantitative. *Annals of Statistics* **17**, 31–57.

Lauritzen, S. L. (1992). Propagation of probabilities, means and variances in mixed graphical association models. *Journal of the American Statistical Association* **87**, 1098–1108.

Lauritzen, S. L. (1996). *Graphical Models*. Oxford University Press.

Lauritzen, S. L. and D. J. Spiegelhalter (1988). Local computations with probabailities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society* **50**, 157–224.

Lawley, D. N. (1953). A modified method of estimation in factor analysis and some large sample results. In *Uppsala Symposium on Psychological Factor Analysis*, Number 3 in Nordisk Psykologi Monograph Series, pp. 35–42. Uppsala: Almqvist and Wiksell.

Lawrence, N. D., A. I. T. Rowstron, C. M. Bishop, and M. J. Taylor (2002). Optimising synchronisation times for mobile devices. In T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems*, Volume 14, pp. 1401–1408. MIT Press.

Lazarsfeld, P. F. and N. W. Henry (1968). *Latent Structure Analysis*. Houghton Mifflin.

Le Cun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation* **1**(4), 541–551.

Le Cun, Y., J. S. Denker, and S. A. Solla (1990). Optimal brain damage. In D. S. Touretzky (Ed.),

Advances in Neural Information Processing Systems, Volume 2, pp. 598–605. Morgan Kaufmann.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86**, 2278–2324.

Lee, Y., Y. Lin, and G. Wahba (2001). Multicategory support vector machines. Technical Report 1040, Department of Statistics, University of Madison, Wisconsin.

Leen, T. K. (1995). From data distributions to regularization in invariant learning. *Neural Computation* **7**, 974–981.

Lindley, D. V. (1982). Scoring rules and the inevitability of probability. *International Statistical Review* **50**, 1–26.

Liu, J. S. (Ed.) (2001). *Monte Carlo Strategies in Scientific Computing*. Springer.

Lloyd, S. P. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory* **28**(2), 129–137.

Lütkepohl, H. (1996). *Handbook of Matrices*. Wiley.

MacKay, D. J. C. (1992a). Bayesian interpolation. *Neural Computation* **4**(3), 415–447.

MacKay, D. J. C. (1992b). The evidence framework applied to classification networks. *Neural Computation* **4**(5), 720–736.

MacKay, D. J. C. (1992c). A practical Bayesian framework for back-propagation networks. *Neural Computation* **4**(3), 448–472.

MacKay, D. J. C. (1994). Bayesian methods for backprop networks. In E. Domany, J. L. van Hemmen, and K. Schulten (Eds.), *Models of Neural Networks, III*, Chapter 6, pp. 211–254. Springer.

MacKay, D. J. C. (1995). Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research, A* **354**(1), 73–80.

MacKay, D. J. C. (1997). Ensemble learning for hidden Markov models. Unpublished manuscript, Department of Physics, University of Cambridge.

MacKay, D. J. C. (1998). Introduction to Gaussian processes. In C. M. Bishop (Ed.), *Neural Networks and Machine Learning*, pp. 133–166. Springer.

MacKay, D. J. C. (1999). Comparison of approximate methods for handling hyperparameters. *Neural Computation* **11**(5), 1035–1068.

MacKay, D. J. C. (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

MacKay, D. J. C. and M. N. Gibbs (1999). Density networks. In J. W. Kay and D. M. Titterington (Eds.), *Statistics and Neural Networks: Advances at the Interface*, Chapter 5, pp. 129–145. Oxford University Press.

MacKay, D. J. C. and R. M. Neal (1999). Good error-correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory* **45**, 399–431.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In L. M. LeCam and J. Neyman (Eds.), *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Volume I, pp. 281–297. University of California Press.

Magnus, J. R. and H. Neudecker (1999). *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Wiley.

Mallat, S. (1999). *A Wavelet Tour of Signal Processing* (Second ed.). Academic Press.

Manning, C. D. and H. Schütze (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

Mardia, K. V. and P. E. Jupp (2000). *Directional Statistics*. Wiley.

Maybeck, P. S. (1982). *Stochastic models, estimation and control*. Academic Press.

McAllester, D. A. (2003). PAC-Bayesian stochastic model selection. *Machine Learning* **51**(1), 5–21.

McCullagh, P. and J. A. Nelder (1989). *Generalized Linear Models* (Second ed.). Chapman and Hall.

McCulloch, W. S. and W. Pitts (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* **5**, 115–133. Reprinted in Anderson and Rosenfeld (1988).

McEliece, R. J., D. J. C. MacKay, and J. F. Cheng (1998). Turbo decoding as an instance of Pearl's 'Belief Ppropagation' algorithm. *IEEE Journal on Selected Areas in Communications* **16**, 140–152.

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

McLachlan, G. J. and T. Krishnan (1997). *The EM Algorithm and its Extensions*. Wiley.

McLachlan, G. J. and D. Peel (2000). *Finite Mixture Models*. Wiley.

Meng, X. L. and D. B. Rubin (1993). Maximum likelihood estimation via the ECM algorithm: a general framework. *Biometrika* **80**, 267–278.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics* **21**(6), 1087–1092.

Metropolis, N. and S. Ulam (1949). The Monte Carlo method. *Journal of the American Statistical Association* **44**(247), 335–341.

Mika, S., G. Rätsch, J. Weston, and B. Schölkopf (1999). Fisher discriminant analysis with kernels. In Y. H. Hu, J. Larsen, E. Wilson, and S. Douglas (Eds.), *Neural Networks for Signal Processing IX*, pp. 41–48. IEEE.

Minka, T. (2001a). Expectation propagation for approximate Bayesian inference. In J. Breese and D. Koller (Eds.), *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pp. 362–369. Morgan Kaufmann.

Minka, T. (2001b). *A family of approximate algorithms for Bayesian inference*. Ph. D. thesis, MIT.

Minka, T. (1998) Inferring a Gaussian distribution. MIT Media Lab note. Available from http://research.microsoft.com/~minka/.

Minka, T. (2004). Power EP. Technical Report MSR-TR-2004-149, Microsoft Research Cambridge.

Minka, T. (2005). Divergence measures and message passing. Technical Report MSR-TR-2005-173, Microsoft Research Cambridge.

Minka, T. P. (2001c). Automatic choice of dimensionality for PCA. In T. K. Leen, T. G. Dietterich, and V. Tresp (Eds.), *Advances in Neural Information Processing Systems*, Volume 13, pp. 598–604. MIT Press.

Minsky, M. L. and S. A. Papert (1969). *Perceptrons*. MIT Press. Expanded edition 1990.

Miskin, J. W. and D. J. C. MacKay (2001). Ensemble learning for blind source separation. In S. J. Roberts and R. M. Everson (Eds.), *Independent Component Analysis: Principles and Practice*. Cambridge University Press.

Møller, M. (1993). Efficient Training of Feed-Forward Neural Networks. Ph. D. thesis, Aarhus University, Denmark.

Moody, J. and C. J. Darken (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation* **1**(2), 281–294.

Moore, A. W. (2000). The anchors hierarch: using the triangle inequality to survive high dimensional data. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pp. 397–405.

Müller, K. R., S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf (2001). An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks* **12**(2), 181–202.

Müller, P. and F. A. Quintana (2004). Nonparametric Bayesian data analysis. *Statistical Science* **19**(1), 95–110.

Nabney, I. T. (2002). *Netlab: Algorithms for Pattern Recognition*. Springer.

Nadaraya, É. A. (1964). On estimating regression. *Theory of Probability and its Applications* **9**(1), 141–142.

Nag, R., K. Wong, and F. Fallside (1986). Script recognition using hidden markov models. In *ICASSP86*, pp. 2071–2074. IEEE.

Neal, R. M. (1993). Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, Canada.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Springer. Lecture Notes in Statistics 118.

Neal, R. M. (1997). Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. Technical Report 9702, Department of Computer Statistics, University of Toronto.

Neal, R. M. (1999). Suppressing random walks in Markov chain Monte Carlo using ordered overrelaxation. In M. I. Jordan (Ed.), *Learning in Graphical Models*, pp. 205–228. MIT Press.

Neal, R. M. (2000). Markov chain sampling for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics* **9**, 249–265.

Neal, R. M. (2003). Slice sampling. *Annals of Statistics* **31**, 705–767.

Neal, R. M. and G. E. Hinton (1999). A new view of the EM algorithm that justifies incremental and other variants. In M. I. Jordan (Ed.), *Learning in Graphical Models*, pp. 355–368. MIT Press.

Nelder, J. A. and R. W. M. Wedderburn (1972). Generalized linear models. *Journal of the Royal Statistical Society, A* **135**, 370–384.

Nilsson, N. J. (1965). *Learning Machines*. McGraw-Hill. Reprinted as *The Mathematical Foundations of Learning Machines*, Morgan Kaufmann, (1990).

Nocedal, J. and S. J. Wright (1999). *Numerical Optimization*. Springer.

Nowlan, S. J. and G. E. Hinton (1992). Simplifying neural networks by soft weight sharing. *Neural Computation* **4**(4), 473–493.

Ogden, R. T. (1997). *Essential Wavelets for Statistical Applications and Data Analysis*. Birkhäuser.

Opper, M. and O. Winther (1999). A Bayesian approach to on-line learning. In D. Saad (Ed.), *On-Line Learning in Neural Networks*, pp. 363–378. Cambridge University Press.

Opper, M. and O. Winther (2000a). Gaussian processes and SVM: mean field theory and leave-one-out. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Shuurmans (Eds.), *Advances in Large Margin Classifiers*, pp. 311–326. MIT Press.

Opper, M. and O. Winther (2000b). Gaussian processes for classification. *Neural Computation* **12**(11), 2655–2684.

Osuna, E., R. Freund, and F. Girosi (1996). Support vector machines: training and applications. A.I. Memo AIM-1602, MIT.

Papoulis, A. (1984). *Probability, Random Variables, and Stochastic Processes* (Second ed.). McGraw-Hill.

Parisi, G. (1988). *Statistical Field Theory*. Addison-Wesley.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

Pearlmutter, B. A. (1994). Fast exact multiplication by the Hessian. *Neural Computation* **6**(1), 147–160.

Pearlmutter, B. A. and L. C. Parra (1997). Maximum likelihood source separation: a context-sensitive generalization of ICA. In M. C. Mozer, M. I. Jordan, and T. Petsche (Eds.), *Advances in Neural Information Processing Systems*, Volume 9, pp. 613–619. MIT Press.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science, Sixth Series* **2**, 559–572.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola (Eds.), *Advances in Kernel Methods – Support Vector Learning*, pp. 185–208. MIT Press.

Platt, J. C. (2000). Probabilities for SV machines. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Shuurmans (Eds.), *Advances in Large Margin Classifiers*, pp. 61–73. MIT Press.

Platt, J. C., N. Cristianini, and J. Shawe-Taylor (2000). Large margin DAGs for multiclass classification. In S. A. Solla, T. K. Leen, and K. R. Müller (Eds.), *Advances in Neural Information Processing Systems*, Volume 12, pp. 547–553. MIT Press.

Poggio, T. and F. Girosi (1990). Networks for approximation and learning. *Proceedings of the IEEE* **78**(9), 1481–1497.

Powell, M. J. D. (1987). Radial basis functions for multivariable interpolation: a review. In J. C. Mason and M. G. Cox (Eds.), *Algorithms for Approximation*, pp. 143–167. Oxford University Press.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery (1992). *Numerical Recipes in C: The Art of Scientific Computing* (Second ed.). Cambridge University Press.

Qazaz, C. S., C. K. I. Williams, and C. M. Bishop (1997). An upper bound on the Bayesian error bars for generalized linear regression. In S. W. Ellacott, J. C. Mason, and I. J. Anderson (Eds.), *Mathematics of Neural Networks: Models, Algorithms and Applications*, pp. 295–299. Kluwer.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning* **1**(1), 81–106.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Rabiner, L. and B. H. Juang (1993). *Fundamentals of Speech Recognition*. Prentice Hall.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* **77**(2), 257–285.

Ramasubramanian, V. and K. K. Paliwal (1990). A generalized optimization of the k-d tree for fast nearest-neighbour search. In *Proceedings Fourth IEEE Region 10 International Conference (TENCON'89)*, pp. 565–568.

Ramsey, F. (1931). Truth and probability. In R. Braithwaite (Ed.), *The Foundations of Mathematics and other Logical Essays*. Humanities Press.

Rao, C. R. and S. K. Mitra (1971). *Generalized Inverse of Matrices and Its Applications*. Wiley.

Rasmussen, C. E. (1996). *Evaluation of Gaussian Processes and Other Methods for Non-Linear Regression*. Ph. D. thesis, University of Toronto.

Rasmussen, C. E. and J. Quiñonero-Candela (2005). Healing the relevance vector machine by augmentation. In L. D. Raedt and S. Wrobel (Eds.), *Proceedings of the 22nd International Conference on Machine Learning*, pp. 689–696.

Rasmussen, C. E. and C. K. I. Williams (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Rauch, H. E., F. Tung, and C. T. Striebel (1965). Maximum likelihood estimates of linear dynamical systems. *AIAA Journal* **3**, 1445–1450.

Ricotti, L. P., S. Ragazzini, and G. Martinelli (1988). Learning of word stress in a sub-optimal second order backpropagation neural network. In *Proceedings of the IEEE International Conference on Neural Networks*, Volume 1, pp. 355–361. IEEE.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press.

Robbins, H. and S. Monro (1951). A stochastic approximation method. *Annals of Mathematical Statistics* **22**, 400–407.

Robert, C. P. and G. Casella (1999). *Monte Carlo Statistical Methods*. Springer.

Rockafellar, R. (1972). *Convex Analysis*. Princeton University Press.

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan.

Roth, V. and V. Steinhage (2000). Nonlinear discriminant analysis using kernel functions. In S. A.

Solla, T. K. Leen, and K. R. Müller (Eds.), *Advances in Neural Information Processing Systems*, Volume 12. MIT Press.

Roweis, S. (1998). EM algorithms for PCA and SPCA. In M. I. Jordan, M. J. Kearns, and S. A. Solla (Eds.), *Advances in Neural Information Processing Systems*, Volume 10, pp. 626–632. MIT Press.

Roweis, S. and Z. Ghahramani (1999). A unifying review of linear Gaussian models. *Neural Computation* **11**(2), 305–345.

Roweis, S. and L. Saul (2000, December). Nonlinear dimensionality reduction by locally linear embedding. *Science* **290**, 2323–2326.

Rubin, D. B. (1983). Iteratively reweighted least squares. In *Encyclopedia of Statistical Sciences*, Volume 4, pp. 272–275. Wiley.

Rubin, D. B. and D. T. Thayer (1982). EM algorithms for ML factor analysis. *Psychometrika* **47**(1), 69–76.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Volume 1: Foundations, pp. 318–362. MIT Press. Reprinted in Anderson and Rosenfeld (1988).

Rumelhart, D. E., J. L. McClelland, and the PDP Research Group (Eds.) (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Volume 1: Foundations. MIT Press.

Sagan, H. (1969). *Introduction to the Calculus of Variations*. Dover.

Savage, L. J. (1961). The subjective basis of statistical practice. Technical report, Department of Statistics, University of Michigan, Ann Arbor.

Schölkopf, B., J. Platt, J. Shawe-Taylor, A. Smola, and R. C. Williamson (2001). Estimating the support of a high-dimensional distribution. *Neural Computation* **13**(7), 1433–1471.

Schölkopf, B., A. Smola, and K.-R. Müller (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation* **10**(5), 1299–1319.

Schölkopf, B., A. Smola, R. C. Williamson, and P. L. Bartlett (2000). New support vector algorithms. *Neural Computation* **12**(5), 1207–1245.

Schölkopf, B. and A. J. Smola (2002). *Learning with Kernels*. MIT Press.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics* **6**, 461–464.

Schwarz, H. R. (1988). *Finite element methods*. Academic Press.

Seeger, M. (2003). *Bayesian Gaussian Process Models: PAC-Bayesian Generalization Error Bounds and Sparse Approximations*. Ph. D. thesis, University of Edinburg.

Seeger, M., C. K. I. Williams, and N. Lawrence (2003). Fast forward selection to speed up sparse Gaussian processes. In C. M. Bishop and B. Frey (Eds.), *Proceedings Ninth International Workshop on Artificial Intelligence and Statistics, Key West, Florida*.

Shachter, R. D. and M. Peot (1990). Simulation approaches to general probabilistic inference on belief networks. In P. P. Bonissone, M. Henrion, L. N. Kanal, and J. F. Lemmer (Eds.), *Uncertainty in Artificial Intelligence*, Volume 5. Elsevier.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal* **27**(3), 379–423 and 623–656.

Shawe-Taylor, J. and N. Cristianini (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Sietsma, J. and R. J. F. Dow (1991). Creating artificial neural networks that generalize. *Neural Networks* **4**(1), 67–79.

Simard, P., Y. Le Cun, and J. Denker (1993). Efficient pattern recognition using a new transformation distance. In S. J. Hanson, J. D. Cowan, and

C. L. Giles (Eds.), *Advances in Neural Information Processing Systems*, Volume 5, pp. 50–58. Morgan Kaufmann.

Simard, P., B. Victorri, Y. Le Cun, and J. Denker (1992). Tangent prop – a formalism for specifying selected invariances in an adaptive network. In J. E. Moody, S. J. Hanson, and R. P. Lippmann (Eds.), *Advances in Neural Information Processing Systems*, Volume 4, pp. 895–903. Morgan Kaufmann.

Simard, P. Y., D. Steinkraus, and J. Platt (2003). Best practice for convolutional neural networks applied to visual document analysis. In *Proceedings International Conference on Document Analysis and Recognition (ICDAR)*, pp. 958–962. IEEE Computer Society.

Sirovich, L. (1987). Turbulence and the dynamics of coherent structures. *Quarterly Applied Mathematics* **45**(3), 561–590.

Smola, A. J. and P. Bartlett (2001). Sparse greedy Gaussian process regression. In T. K. Leen, T. G. Dietterich, and V. Tresp (Eds.), *Advances in Neural Information Processing Systems*, Volume 13, pp. 619–625. MIT Press.

Spiegelhalter, D. and S. Lauritzen (1990). Sequential updating of conditional probabilities on directed graphical structures. *Networks* **20**, 579–605.

Stinchecombe, M. and H. White (1989). Universal approximation using feed-forward networks with non-sigmoid hidden layer activation functions. In *International Joint Conference on Neural Networks*, Volume 1, pp. 613–618. IEEE.

Stone, J. V. (2004). *Independent Component Analysis: A Tutorial Introduction*. MIT Press.

Sung, K. K. and T. Poggio (1994). Example-based learning for view-based human face detection. A.I. Memo 1521, MIT.

Sutton, R. S. and A. G. Barto (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Svensén, M. and C. M. Bishop (2004). Robust Bayesian mixture modelling. *Neurocomputing* **64**, 235–252.

Tarassenko, L. (1995). Novelty detection for the identification of masses in mamograms. In *Proceedings Fourth IEE International Conference on Artificial Neural Networks*, Volume 4, pp. 442–447. IEE.

Tax, D. and R. Duin (1999). Data domain description by support vectors. In M. Verleysen (Ed.), *Proceedings European Symposium on Artificial Neural Networks, ESANN*, pp. 251–256. D. Facto Press.

Teh, Y. W., M. I. Jordan, M. J. Beal, and D. M. Blei (2006). Hierarchical Dirichlet processes. *Journal of the Americal Statistical Association*. to appear.

Tenenbaum, J. B., V. de Silva, and J. C. Langford (2000, December). A global framework for nonlinear dimensionality reduction. *Science* **290**, 2319–2323.

Tesauro, G. (1994). TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation* **6**(2), 215–219.

Thiesson, B., D. M. Chickering, D. Heckerman, and C. Meek (2004). ARMA time-series modelling with graphical models. In M. Chickering and J. Halpern (Eds.), *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence, Banff, Canada*, pp. 552–560. AUAI Press.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, B* **58**, 267–288.

Tierney, L. (1994). Markov chains for exploring posterior distributions. *Annals of Statistics* **22**(4), 1701–1762.

Tikhonov, A. N. and V. Y. Arsenin (1977). *Solutions of Ill-Posed Problems*. V. H. Winston.

Tino, P. and I. T. Nabney (2002). Hierarchical GTM: constructing localized non-linear projection manifolds in a principled way. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(5), 639–656.

Tino, P., I. T. Nabney, and Y. Sun (2001). Using directional curvatures to visualize folding patterns of the GTM projection manifolds. In

G. Dorffner, H. Bischof, and K. Hornik (Eds.), *Artificial Neural Networks – ICANN 2001*, pp. 421–428. Springer.

Tipping, M. E. (1999). Probabilistic visualisation of high-dimensional binary data. In M. S. Kearns, S. A. Solla, and D. A. Cohn (Eds.), *Advances in Neural Information Processing Systems*, Volume 11, pp. 592–598. MIT Press.

Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research* **1**, 211–244.

Tipping, M. E. and C. M. Bishop (1997). Probabilistic principal component analysis. Technical Report NCRG/97/010, Neural Computing Research Group, Aston University.

Tipping, M. E. and C. M. Bishop (1999a). Mixtures of probabilistic principal component analyzers. *Neural Computation* **11**(2), 443–482.

Tipping, M. E. and C. M. Bishop (1999b). Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B* **21**(3), 611–622.

Tipping, M. E. and A. Faul (2003). Fast marginal likelihood maximization for sparse Bayesian models. In C. M. Bishop and B. Frey (Eds.), *Proceedings Ninth International Workshop on Artificial Intelligence and Statistics, Key West, Florida*.

Tong, S. and D. Koller (2000). Restricted Bayes optimal classifiers. In *Proceedings 17th National Conference on Artificial Intelligence*, pp. 658–664. AAAI.

Tresp, V. (2001). Scaling kernel-based systems to large data sets. *Data Mining and Knowledge Discovery* **5**(3), 197–211.

Uhlenbeck, G. E. and L. S. Ornstein (1930). On the theory of Brownian motion. *Phys. Rev.* **36**, 823–841.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the Association for Computing Machinery* **27**, 1134–1142.

Vapnik, V. N. (1982). *Estimation of dependences based on empirical data*. Springer.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer.

Vapnik, V. N. (1998). *Statistical learning theory*. Wiley.

Veropoulos, K., C. Campbell, and N. Cristianini (1999). Controlling the sensitivity of support vector machines. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI99), Workshop ML3*, pp. 55–60.

Vidakovic, B. (1999). *Statistical Modelling by Wavelets*. Wiley.

Viola, P. and M. Jones (2004). Robust real-time face detection. *International Journal of Computer Vision* **57**(2), 137–154.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory* **IT-13**, 260–267.

Viterbi, A. J. and J. K. Omura (1979). *Principles of Digital Communication and Coding*. McGraw-Hill.

Wahba, G. (1975). A comparison of GCV and GML for choosing the smoothing parameter in the generalized spline smoothing problem. *Numerical Mathematics* **24**, 383–393.

Wainwright, M. J., T. S. Jaakkola, and A. S. Willsky (2005). A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory* **51**, 2313–2335.

Walker, A. M. (1969). On the asymptotic behaviour of posterior distributions. *Journal of the Royal Statistical Society, B* **31**(1), 80–88.

Walker, S. G., P. Damien, P. W. Laud, and A. F. M. Smith (1999). Bayesian nonparametric inference for random distributions and related functions (with discussion). *Journal of the Royal Statistical Society, B* **61**(3), 485–527.

Watson, G. S. (1964). Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics. Series A* **26**, 359–372.

Webb, A. R. (1994). Functional approximation by feed-forward networks: a least-squares approach to generalisation. *IEEE Transactions on Neural Networks* **5**(3), 363–371.

Weisstein, E. W. (1999). *CRC Concise Encyclopedia of Mathematics*. Chapman and Hall, and CRC.

Weston, J. and C. Watkins (1999). Multi-class support vector machines. In M. Verlysen (Ed.), *Proceedings ESANN'99, Brussels*. D-Facto Publications.

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

Widrow, B. and M. E. Hoff (1960). Adaptive switching circuits. In *IRE WESCON Convention Record*, Volume 4, pp. 96–104. Reprinted in Anderson and Rosenfeld (1988).

Widrow, B. and M. A. Lehr (1990). 30 years of adaptive neural networks: perceptron, madeline, and backpropagation. *Proceedings of the IEEE* **78**(9), 1415–1442.

Wiegerinck, W. and T. Heskes (2003). Fractional belief propagation. In S. Becker, S. Thrun, and K. Obermayer (Eds.), *Advances in Neural Information Processing Systems*, Volume 15, pp. 455–462. MIT Press.

Williams, C. K. I. (1998). Computation with infinite neural networks. *Neural Computation* **10**(5), 1203–1216.

Williams, C. K. I. (1999). Prediction with Gaussian processes: from linear regression to linear prediction and beyond. In M. I. Jordan (Ed.), *Learning in Graphical Models*, pp. 599–621. MIT Press.

Williams, C. K. I. and D. Barber (1998). Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20**, 1342–1351.

Williams, C. K. I. and M. Seeger (2001). Using the Nystrom method to speed up kernel machines. In T. K. Leen, T. G. Dietterich, and V. Tresp (Eds.), *Advances in Neural Information Processing Systems*, Volume 13, pp. 682–688. MIT Press.

Williams, O., A. Blake, and R. Cipolla (2005). Sparse Bayesian learning for efficient visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27**(8), 1292–1304.

Williams, P. M. (1996). Using neural networks to model conditional multivariate densities. *Neural Computation* **8**(4), 843–854.

Winn, J. and C. M. Bishop (2005). Variational message passing. *Journal of Machine Learning Research* **6**, 661–694.

Zarchan, P. and H. Musoff (2005). *Fundamentals of Kalman Filtering: A Practical Approach* (Second ed.). AIAA.

Index

Page numbers in **bold** indicate the primary source of information for the corresponding topic.