

9

Mixture Models and EM

If we define a joint distribution over observed and latent variables, the corresponding distribution of the observed variables alone is obtained by marginalization. This allows relatively complex marginal distributions over observed variables to be expressed in terms of more tractable joint distributions over the expanded space of observed and latent variables. The introduction of latent variables thereby allows complicated distributions to be formed from simpler components. In this chapter, we shall see that mixture distributions, such as the Gaussian mixture discussed in Section 2.3.9, can be interpreted in terms of discrete latent variables. Continuous latent variables will form the subject of Chapter 12.

As well as providing a framework for building more complex probability distributions, mixture models can also be used to cluster data. We therefore begin our discussion of mixture distributions by considering the problem of finding clusters in a set of data points, which we approach first using a nonprobabilistic technique called the K -means algorithm (Lloyd, 1982). Then we introduce the latent variable

Section 9.1

Section 9.2

Section 9.3

Section 9.4

view of mixture distributions in which the discrete latent variables can be interpreted as defining assignments of data points to specific components of the mixture. A general technique for finding maximum likelihood estimators in latent variable models is the expectation-maximization (EM) algorithm. We first of all use the Gaussian mixture distribution to motivate the EM algorithm in a fairly informal way, and then we give a more careful treatment based on the latent variable viewpoint. We shall see that the K -means algorithm corresponds to a particular nonprobabilistic limit of EM applied to mixtures of Gaussians. Finally, we discuss EM in some generality.

Gaussian mixture models are widely used in data mining, pattern recognition, machine learning, and statistical analysis. In many applications, their parameters are determined by maximum likelihood, typically using the EM algorithm. However, as we shall see there are some significant limitations to the maximum likelihood approach, and in Chapter 10 we shall show that an elegant Bayesian treatment can be given using the framework of variational inference. This requires little additional computation compared with EM, and it resolves the principal difficulties of maximum likelihood while also allowing the number of components in the mixture to be inferred automatically from the data.

9.1. K -means Clustering

We begin by considering the problem of identifying groups, or clusters, of data points in a multidimensional space. Suppose we have a data set $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ consisting of N observations of a random D -dimensional Euclidean variable \mathbf{x} . Our goal is to partition the data set into some number K of clusters, where we shall suppose for the moment that the value of K is given. Intuitively, we might think of a cluster as comprising a group of data points whose inter-point distances are small compared with the distances to points outside of the cluster. We can formalize this notion by first introducing a set of D -dimensional vectors $\boldsymbol{\mu}_k$, where $k = 1, \dots, K$, in which $\boldsymbol{\mu}_k$ is a prototype associated with the k^{th} cluster. As we shall see shortly, we can think of the $\boldsymbol{\mu}_k$ as representing the centres of the clusters. Our goal is then to find an assignment of data points to clusters, as well as a set of vectors $\{\boldsymbol{\mu}_k\}$, such that the sum of the squares of the distances of each data point to its closest vector $\boldsymbol{\mu}_k$, is a minimum.

It is convenient at this point to define some notation to describe the assignment of data points to clusters. For each data point \mathbf{x}_n , we introduce a corresponding set of binary indicator variables $r_{nk} \in \{0, 1\}$, where $k = 1, \dots, K$ describing which of the K clusters the data point \mathbf{x}_n is assigned to, so that if data point \mathbf{x}_n is assigned to cluster k then $r_{nk} = 1$, and $r_{nj} = 0$ for $j \neq k$. This is known as the 1-of- K coding scheme. We can then define an objective function, sometimes called a *distortion measure*, given by

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 \quad (9.1)$$

which represents the sum of the squares of the distances of each data point to its

assigned vector μ_k . Our goal is to find values for the $\{r_{nk}\}$ and the $\{\mu_k\}$ so as to minimize J . We can do this through an iterative procedure in which each iteration involves two successive steps corresponding to successive optimizations with respect to the r_{nk} and the μ_k . First we choose some initial values for the μ_k . Then in the first phase we minimize J with respect to the r_{nk} , keeping the μ_k fixed. In the second phase we minimize J with respect to the μ_k , keeping r_{nk} fixed. This two-stage optimization is then repeated until convergence. We shall see that these two stages of updating r_{nk} and updating μ_k correspond respectively to the E (expectation) and M (maximization) steps of the EM algorithm, and to emphasize this we shall use the terms E step and M step in the context of the *K*-means algorithm.

Consider first the determination of the r_{nk} . Because J in (9.1) is a linear function of r_{nk} , this optimization can be performed easily to give a closed form solution. The terms involving different n are independent and so we can optimize for each n separately by choosing r_{nk} to be 1 for whichever value of k gives the minimum value of $\|\mathbf{x}_n - \mu_k\|^2$. In other words, we simply assign the n^{th} data point to the closest cluster centre. More formally, this can be expressed as

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \|\mathbf{x}_n - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases} \quad (9.2)$$

Now consider the optimization of the μ_k with the r_{nk} held fixed. The objective function J is a quadratic function of μ_k , and it can be minimized by setting its derivative with respect to μ_k to zero giving

$$2 \sum_{n=1}^N r_{nk} (\mathbf{x}_n - \mu_k) = 0 \quad (9.3)$$

which we can easily solve for μ_k to give

$$\mu_k = \frac{\sum_n r_{nk} \mathbf{x}_n}{\sum_n r_{nk}}. \quad (9.4)$$

The denominator in this expression is equal to the number of points assigned to cluster k , and so this result has a simple interpretation, namely set μ_k equal to the mean of all of the data points \mathbf{x}_n assigned to cluster k . For this reason, the procedure is known as the *K*-means algorithm.

The two phases of re-assigning data points to clusters and re-computing the cluster means are repeated in turn until there is no further change in the assignments (or until some maximum number of iterations is exceeded). Because each phase reduces the value of the objective function J , convergence of the algorithm is assured. However, it may converge to a local rather than global minimum of J . The convergence properties of the *K*-means algorithm were studied by MacQueen (1967).

The *K*-means algorithm is illustrated using the Old Faithful data set in Figure 9.1. For the purposes of this example, we have made a linear re-scaling of the data, known as *standardizing*, such that each of the variables has zero mean and unit standard deviation. For this example, we have chosen $K = 2$, and so in this

Section 9.4

Exercise 9.1

Appendix A

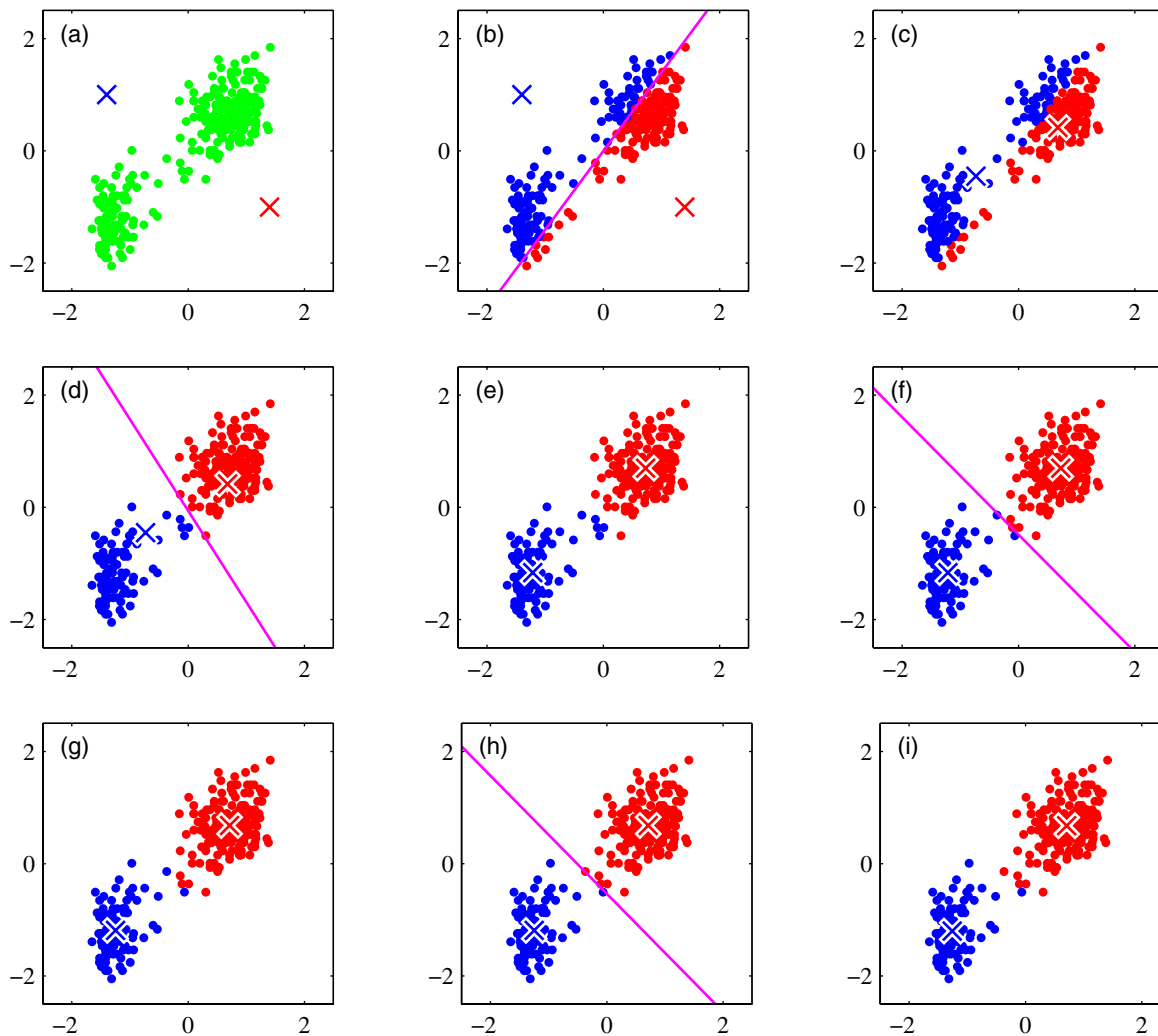
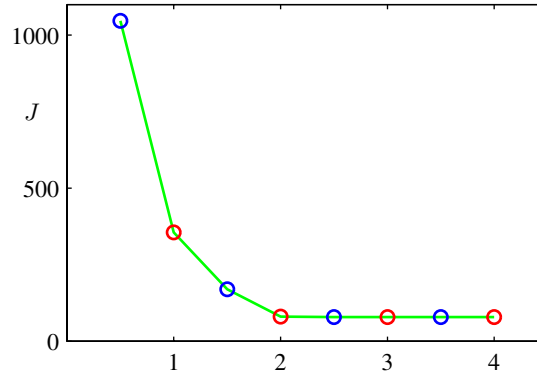


Figure 9.1 Illustration of the K -means algorithm using the re-scaled Old Faithful data set. (a) Green points denote the data set in a two-dimensional Euclidean space. The initial choices for centres μ_1 and μ_2 are shown by the red and blue crosses, respectively. (b) In the initial E step, each data point is assigned either to the red cluster or to the blue cluster, according to which cluster centre is nearer. This is equivalent to classifying the points according to which side of the perpendicular bisector of the two cluster centres, shown by the magenta line, they lie on. (c) In the subsequent M step, each cluster centre is re-computed to be the mean of the points assigned to the corresponding cluster. (d)–(i) show successive E and M steps through to final convergence of the algorithm.

Figure 9.2 Plot of the cost function J given by (9.1) after each E step (blue points) and M step (red points) of the K -means algorithm for the example shown in Figure 9.1. The algorithm has converged after the third M step, and the final EM cycle produces no changes in either the assignments or the prototype vectors.



case, the assignment of each data point to the nearest cluster centre is equivalent to a classification of the data points according to which side they lie of the perpendicular bisector of the two cluster centres. A plot of the cost function J given by (9.1) for the Old Faithful example is shown in Figure 9.2.

Note that we have deliberately chosen poor initial values for the cluster centres so that the algorithm takes several steps before convergence. In practice, a better initialization procedure would be to choose the cluster centres μ_k to be equal to a random subset of K data points. It is also worth noting that the K -means algorithm itself is often used to initialize the parameters in a Gaussian mixture model before applying the EM algorithm.

Section 9.2.2

A direct implementation of the K -means algorithm as discussed here can be relatively slow, because in each E step it is necessary to compute the Euclidean distance between every prototype vector and every data point. Various schemes have been proposed for speeding up the K -means algorithm, some of which are based on precomputing a data structure such as a tree such that nearby points are in the same subtree (Ramasubramanian and Paliwal, 1990; Moore, 2000). Other approaches make use of the triangle inequality for distances, thereby avoiding unnecessary distance calculations (Hodgson, 1998; Elkan, 2003).

So far, we have considered a batch version of K -means in which the whole data set is used together to update the prototype vectors. We can also derive an on-line stochastic algorithm (MacQueen, 1967) by applying the Robbins-Monro procedure to the problem of finding the roots of the regression function given by the derivatives of J in (9.1) with respect to μ_k . This leads to a sequential update in which, for each data point \mathbf{x}_n in turn, we update the nearest prototype μ_k using

Section 2.3.5

Exercise 9.2

$$\mu_k^{\text{new}} = \mu_k^{\text{old}} + \eta_n(\mathbf{x}_n - \mu_k^{\text{old}}) \quad (9.5)$$

where η_n is the learning rate parameter, which is typically made to decrease monotonically as more data points are considered.

The K -means algorithm is based on the use of squared Euclidean distance as the measure of dissimilarity between a data point and a prototype vector. Not only does this limit the type of data variables that can be considered (it would be inappropriate for cases where some or all of the variables represent categorical labels for instance),

Section 2.3.7

but it can also make the determination of the cluster means nonrobust to outliers. We can generalize the K -means algorithm by introducing a more general dissimilarity measure $\mathcal{V}(\mathbf{x}, \mathbf{x}')$ between two vectors \mathbf{x} and \mathbf{x}' and then minimizing the following distortion measure

$$\tilde{J} = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \mathcal{V}(\mathbf{x}_n, \boldsymbol{\mu}_k) \quad (9.6)$$

which gives the K -medoids algorithm. The E step again involves, for given cluster prototypes $\boldsymbol{\mu}_k$, assigning each data point to the cluster for which the dissimilarity to the corresponding prototype is smallest. The computational cost of this is $O(KN)$, as is the case for the standard K -means algorithm. For a general choice of dissimilarity measure, the M step is potentially more complex than for K -means, and so it is common to restrict each cluster prototype to be equal to one of the data vectors assigned to that cluster, as this allows the algorithm to be implemented for any choice of dissimilarity measure $\mathcal{V}(\cdot, \cdot)$ so long as it can be readily evaluated. Thus the M step involves, for each cluster k , a discrete search over the N_k points assigned to that cluster, which requires $O(N_k^2)$ evaluations of $\mathcal{V}(\cdot, \cdot)$.

One notable feature of the K -means algorithm is that at each iteration, every data point is assigned uniquely to one, and only one, of the clusters. Whereas some data points will be much closer to a particular centre $\boldsymbol{\mu}_k$ than to any other centre, there may be other data points that lie roughly midway between cluster centres. In the latter case, it is not clear that the hard assignment to the nearest cluster is the most appropriate. We shall see in the next section that by adopting a probabilistic approach, we obtain ‘soft’ assignments of data points to clusters in a way that reflects the level of uncertainty over the most appropriate assignment. This probabilistic formulation brings with it numerous benefits.

9.1.1 Image segmentation and compression

As an illustration of the application of the K -means algorithm, we consider the related problems of image segmentation and image compression. The goal of segmentation is to partition an image into regions each of which has a reasonably homogeneous visual appearance or which corresponds to objects or parts of objects (Forsyth and Ponce, 2003). Each pixel in an image is a point in a 3-dimensional space comprising the intensities of the red, blue, and green channels, and our segmentation algorithm simply treats each pixel in the image as a separate data point. Note that strictly this space is not Euclidean because the channel intensities are bounded by the interval $[0, 1]$. Nevertheless, we can apply the K -means algorithm without difficulty. We illustrate the result of running K -means to convergence, for any particular value of K , by re-drawing the image replacing each pixel vector with the $\{R, G, B\}$ intensity triplet given by the centre $\boldsymbol{\mu}_k$ to which that pixel has been assigned. Results for various values of K are shown in Figure 9.3. We see that for a given value of K , the algorithm is representing the image using a palette of only K colours. It should be emphasized that this use of K -means is not a particularly sophisticated approach to image segmentation, not least because it takes no account of the spatial proximity of different pixels. The image segmentation problem is in general extremely difficult

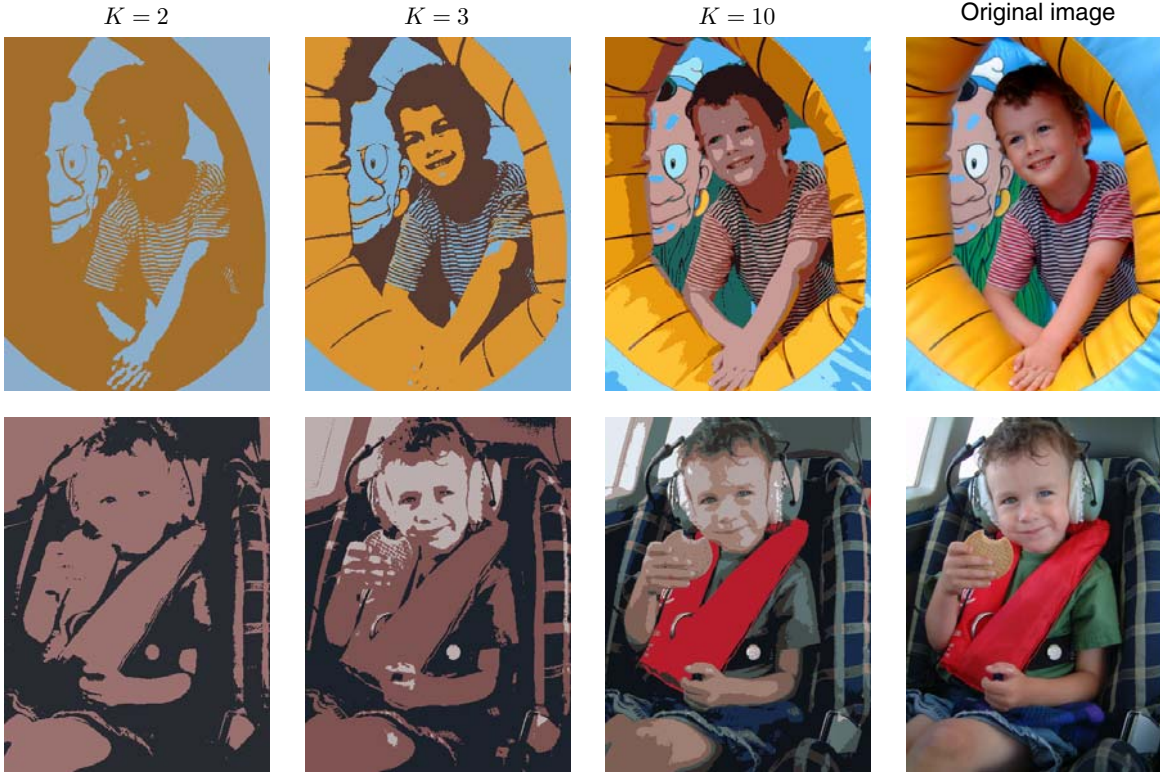


Figure 9.3 Two examples of the application of the K -means clustering algorithm to image segmentation showing the initial images together with their K -means segmentations obtained using various values of K . This also illustrates the use of vector quantization for data compression, in which smaller values of K give higher compression at the expense of poorer image quality.

and remains the subject of active research and is introduced here simply to illustrate the behaviour of the K -means algorithm.

We can also use the result of a clustering algorithm to perform data compression. It is important to distinguish between *lossless data compression*, in which the goal is to be able to reconstruct the original data exactly from the compressed representation, and *lossy data compression*, in which we accept some errors in the reconstruction in return for higher levels of compression than can be achieved in the lossless case. We can apply the K -means algorithm to the problem of lossy data compression as follows. For each of the N data points, we store only the identity k of the cluster to which it is assigned. We also store the values of the K cluster centres μ_k , which typically requires significantly less data, provided we choose $K \ll N$. Each data point is then approximated by its nearest centre μ_k . New data points can similarly be compressed by first finding the nearest μ_k and then storing the label k instead of the original data vector. This framework is often called *vector quantization*, and the vectors μ_k are called *code-book vectors*.

The image segmentation problem discussed above also provides an illustration of the use of clustering for data compression. Suppose the original image has N pixels comprising $\{R, G, B\}$ values each of which is stored with 8 bits of precision. Then to transmit the whole image directly would cost $24N$ bits. Now suppose we first run K -means on the image data, and then instead of transmitting the original pixel intensity vectors we transmit the identity of the nearest vector μ_k . Because there are K such vectors, this requires $\log_2 K$ bits per pixel. We must also transmit the K code book vectors μ_k , which requires $24K$ bits, and so the total number of bits required to transmit the image is $24K + N \log_2 K$ (rounding up to the nearest integer). The original image shown in Figure 9.3 has $240 \times 180 = 43,200$ pixels and so requires $24 \times 43,200 = 1,036,800$ bits to transmit directly. By comparison, the compressed images require 43,248 bits ($K = 2$), 86,472 bits ($K = 3$), and 173,040 bits ($K = 10$), respectively, to transmit. These represent compression ratios compared to the original image of 4.2%, 8.3%, and 16.7%, respectively. We see that there is a trade-off between degree of compression and image quality. Note that our aim in this example is to illustrate the K -means algorithm. If we had been aiming to produce a good image compressor, then it would be more fruitful to consider small blocks of adjacent pixels, for instance 5×5 , and thereby exploit the correlations that exist in natural images between nearby pixels.

9.2. Mixtures of Gaussians

In Section 2.3.9 we motivated the Gaussian mixture model as a simple linear superposition of Gaussian components, aimed at providing a richer class of density models than the single Gaussian. We now turn to a formulation of Gaussian mixtures in terms of discrete *latent* variables. This will provide us with a deeper insight into this important distribution, and will also serve to motivate the expectation-maximization algorithm.

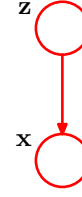
Recall from (2.188) that the Gaussian mixture distribution can be written as a linear superposition of Gaussians in the form

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \mu_k, \Sigma_k). \quad (9.7)$$

Let us introduce a K -dimensional binary random variable \mathbf{z} having a 1-of- K representation in which a particular element z_k is equal to 1 and all other elements are equal to 0. The values of z_k therefore satisfy $z_k \in \{0, 1\}$ and $\sum_k z_k = 1$, and we see that there are K possible states for the vector \mathbf{z} according to which element is nonzero. We shall define the joint distribution $p(\mathbf{x}, \mathbf{z})$ in terms of a marginal distribution $p(\mathbf{z})$ and a conditional distribution $p(\mathbf{x} | \mathbf{z})$, corresponding to the graphical model in Figure 9.4. The marginal distribution over \mathbf{z} is specified in terms of the mixing coefficients π_k , such that

$$p(z_k = 1) = \pi_k$$

Figure 9.4 Graphical representation of a mixture model, in which the joint distribution is expressed in the form $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$.



where the parameters $\{\pi_k\}$ must satisfy

$$0 \leq \pi_k \leq 1 \quad (9.8)$$

together with

$$\sum_{k=1}^K \pi_k = 1 \quad (9.9)$$

in order to be valid probabilities. Because \mathbf{z} uses a 1-of- K representation, we can also write this distribution in the form

$$p(\mathbf{z}) = \prod_{k=1}^K \pi_k^{z_k}. \quad (9.10)$$

Similarly, the conditional distribution of \mathbf{x} given a particular value for \mathbf{z} is a Gaussian

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

which can also be written in the form

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^K \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}. \quad (9.11)$$

The joint distribution is given by $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$, and the marginal distribution of \mathbf{x} is then obtained by summing the joint distribution over all possible states of \mathbf{z} to give

Exercise 9.3

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (9.12)$$

where we have made use of (9.10) and (9.11). Thus the marginal distribution of \mathbf{x} is a Gaussian mixture of the form (9.7). If we have several observations $\mathbf{x}_1, \dots, \mathbf{x}_N$, then, because we have represented the marginal distribution in the form $p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z})$, it follows that for every observed data point \mathbf{x}_n there is a corresponding latent variable \mathbf{z}_n .

We have therefore found an equivalent formulation of the Gaussian mixture involving an explicit latent variable. It might seem that we have not gained much by doing so. However, we are now able to work with the joint distribution $p(\mathbf{x}, \mathbf{z})$

instead of the marginal distribution $p(\mathbf{x})$, and this will lead to significant simplifications, most notably through the introduction of the expectation-maximization (EM) algorithm.

Another quantity that will play an important role is the conditional probability of \mathbf{z} given \mathbf{x} . We shall use $\gamma(z_k)$ to denote $p(z_k = 1|\mathbf{x})$, whose value can be found using Bayes' theorem

$$\begin{aligned}\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) &= \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^K p(z_j = 1)p(\mathbf{x}|z_j = 1)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.\end{aligned}\quad (9.13)$$

We shall view π_k as the prior probability of $z_k = 1$, and the quantity $\gamma(z_k)$ as the corresponding posterior probability once we have observed \mathbf{x} . As we shall see later, $\gamma(z_k)$ can also be viewed as the *responsibility* that component k takes for ‘explaining’ the observation \mathbf{x} .

Section 8.1.2

We can use the technique of ancestral sampling to generate random samples distributed according to the Gaussian mixture model. To do this, we first generate a value for \mathbf{z} , which we denote $\hat{\mathbf{z}}$, from the marginal distribution $p(\mathbf{z})$ and then generate a value for \mathbf{x} from the conditional distribution $p(\mathbf{x}|\hat{\mathbf{z}})$. Techniques for sampling from standard distributions are discussed in Chapter 11. We can depict samples from the joint distribution $p(\mathbf{x}, \mathbf{z})$ by plotting points at the corresponding values of \mathbf{x} and then colouring them according to the value of \mathbf{z} , in other words according to which Gaussian component was responsible for generating them, as shown in Figure 9.5(a). Similarly samples from the marginal distribution $p(\mathbf{x})$ are obtained by taking the samples from the joint distribution and ignoring the values of \mathbf{z} . These are illustrated in Figure 9.5(b) by plotting the \mathbf{x} values without any coloured labels.

We can also use this synthetic data set to illustrate the ‘responsibilities’ by evaluating, for every data point, the posterior probability for each component in the mixture distribution from which this data set was generated. In particular, we can represent the value of the responsibilities $\gamma(z_{nk})$ associated with data point \mathbf{x}_n by plotting the corresponding point using proportions of red, blue, and green ink given by $\gamma(z_{nk})$ for $k = 1, 2, 3$, respectively, as shown in Figure 9.5(c). So, for instance, a data point for which $\gamma(z_{n1}) = 1$ will be coloured red, whereas one for which $\gamma(z_{n2}) = \gamma(z_{n3}) = 0.5$ will be coloured with equal proportions of blue and green ink and so will appear cyan. This should be compared with Figure 9.5(a) in which the data points were labelled using the true identity of the component from which they were generated.

9.2.1 Maximum likelihood

Suppose we have a data set of observations $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, and we wish to model this data using a mixture of Gaussians. We can represent this data set as an $N \times D$

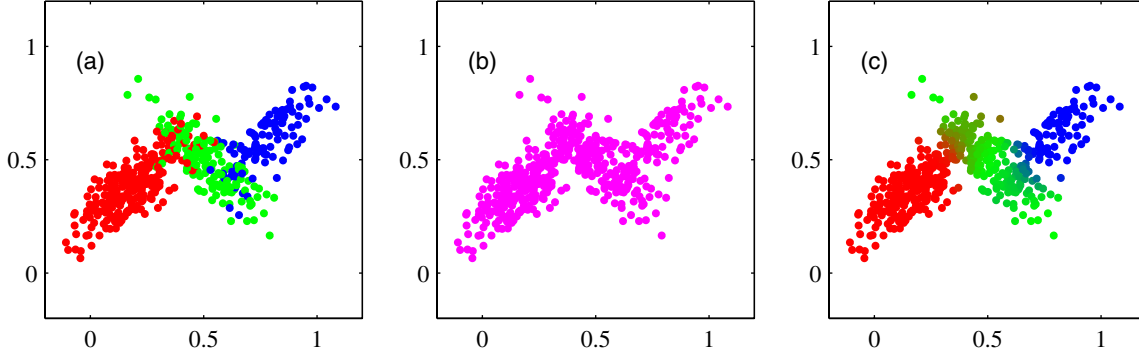


Figure 9.5 Example of 500 points drawn from the mixture of 3 Gaussians shown in Figure 2.23. (a) Samples from the joint distribution $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$ in which the three states of \mathbf{z} , corresponding to the three components of the mixture, are depicted in red, green, and blue, and (b) the corresponding samples from the marginal distribution $p(\mathbf{x})$, which is obtained by simply ignoring the values of \mathbf{z} and just plotting the \mathbf{x} values. The data set in (a) is said to be *complete*, whereas that in (b) is *incomplete*. (c) The same samples in which the colours represent the value of the responsibilities $\gamma(z_{nk})$ associated with data point \mathbf{x}_n , obtained by plotting the corresponding point using proportions of red, blue, and green ink given by $\gamma(z_{nk})$ for $k = 1, 2, 3$, respectively

matrix \mathbf{X} in which the n^{th} row is given by \mathbf{x}_n^T . Similarly, the corresponding latent variables will be denoted by an $N \times K$ matrix \mathbf{Z} with rows \mathbf{z}_n^T . If we assume that the data points are drawn independently from the distribution, then we can express the Gaussian mixture model for this i.i.d. data set using the graphical representation shown in Figure 9.6. From (9.7) the log of the likelihood function is given by

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}. \quad (9.14)$$

Before discussing how to maximize this function, it is worth emphasizing that there is a significant problem associated with the maximum likelihood framework applied to Gaussian mixture models, due to the presence of singularities. For simplicity, consider a Gaussian mixture whose components have covariance matrices given by $\boldsymbol{\Sigma}_k = \sigma_k^2 \mathbf{I}$, where \mathbf{I} is the unit matrix, although the conclusions will hold for general covariance matrices. Suppose that one of the components of the mixture model, let us say the j^{th} component, has its mean $\boldsymbol{\mu}_j$ exactly equal to one of the data

Figure 9.6 Graphical representation of a Gaussian mixture model for a set of N i.i.d. data points $\{\mathbf{x}_n\}$, with corresponding latent points $\{\mathbf{z}_n\}$, where $n = 1, \dots, N$.

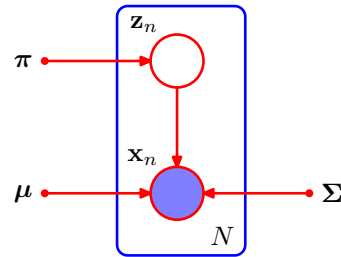
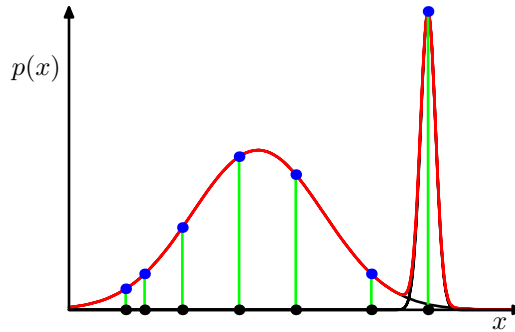


Figure 9.7 Illustration of how singularities in the likelihood function arise with mixtures of Gaussians. This should be compared with the case of a single Gaussian shown in Figure 1.14 for which no singularities arise.



points so that $\mu_j = \mathbf{x}_n$ for some value of n . This data point will then contribute a term in the likelihood function of the form

$$\mathcal{N}(\mathbf{x}_n | \mathbf{x}_n, \sigma_j^2 \mathbf{I}) = \frac{1}{(2\pi)^{1/2}} \frac{1}{\sigma_j}. \quad (9.15)$$

If we consider the limit $\sigma_j \rightarrow 0$, then we see that this term goes to infinity and so the log likelihood function will also go to infinity. Thus the maximization of the log likelihood function is not a well posed problem because such singularities will always be present and will occur whenever one of the Gaussian components ‘collapses’ onto a specific data point. Recall that this problem did not arise in the case of a single Gaussian distribution. To understand the difference, note that if a single Gaussian collapses onto a data point it will contribute multiplicative factors to the likelihood function arising from the other data points and these factors will go to zero exponentially fast, giving an overall likelihood that goes to zero rather than infinity. However, once we have (at least) two components in the mixture, one of the components can have a finite variance and therefore assign finite probability to all of the data points while the other component can shrink onto one specific data point and thereby contribute an ever increasing additive value to the log likelihood. This is illustrated in Figure 9.7. These singularities provide another example of the severe over-fitting that can occur in a maximum likelihood approach. We shall see that this difficulty does not occur if we adopt a Bayesian approach. For the moment, however, we simply note that in applying maximum likelihood to Gaussian mixture models we must take steps to avoid finding such pathological solutions and instead seek local maxima of the likelihood function that are well behaved. We can hope to avoid the singularities by using suitable heuristics, for instance by detecting when a Gaussian component is collapsing and resetting its mean to a randomly chosen value while also resetting its covariance to some large value, and then continuing with the optimization.

A further issue in finding maximum likelihood solutions arises from the fact that for any given maximum likelihood solution, a K -component mixture will have a total of $K!$ equivalent solutions corresponding to the $K!$ ways of assigning K sets of parameters to K components. In other words, for any given (nondegenerate) point in the space of parameter values there will be a further $K! - 1$ additional points all of which give rise to exactly the same distribution. This problem is known as

Section 10.1

identifiability (Casella and Berger, 2002) and is an important issue when we wish to interpret the parameter values discovered by a model. Identifiability will also arise when we discuss models having continuous latent variables in Chapter 12. However, for the purposes of finding a good density model, it is irrelevant because any of the equivalent solutions is as good as any other.

Maximizing the log likelihood function (9.14) for a Gaussian mixture model turns out to be a more complex problem than for the case of a single Gaussian. The difficulty arises from the presence of the summation over k that appears inside the logarithm in (9.14), so that the logarithm function no longer acts directly on the Gaussian. If we set the derivatives of the log likelihood to zero, we will no longer obtain a closed form solution, as we shall see shortly.

One approach is to apply gradient-based optimization techniques (Fletcher, 1987; Nocedal and Wright, 1999; Bishop and Nabney, 2008). Although gradient-based techniques are feasible, and indeed will play an important role when we discuss mixture density networks in Chapter 5, we now consider an alternative approach known as the EM algorithm which has broad applicability and which will lay the foundations for a discussion of variational inference techniques in Chapter 10.

9.2.2 EM for Gaussian mixtures

An elegant and powerful method for finding maximum likelihood solutions for models with latent variables is called the *expectation-maximization* algorithm, or *EM* algorithm (Dempster *et al.*, 1977; McLachlan and Krishnan, 1997). Later we shall give a general treatment of EM, and we shall also show how EM can be generalized to obtain the variational inference framework. Initially, we shall motivate the EM algorithm by giving a relatively informal treatment in the context of the Gaussian mixture model. We emphasize, however, that EM has broad applicability, and indeed it will be encountered in the context of a variety of different models in this book.

Let us begin by writing down the conditions that must be satisfied at a maximum of the likelihood function. Setting the derivatives of $\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ in (9.14) with respect to the means $\boldsymbol{\mu}_k$ of the Gaussian components to zero, we obtain

$$0 = - \sum_{n=1}^N \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\underbrace{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}_{\gamma(z_{nk})}} \boldsymbol{\Sigma}_k (\mathbf{x}_n - \boldsymbol{\mu}_k) \quad (9.16)$$

where we have made use of the form (2.43) for the Gaussian distribution. Note that the posterior probabilities, or responsibilities, given by (9.13) appear naturally on the right-hand side. Multiplying by $\boldsymbol{\Sigma}_k^{-1}$ (which we assume to be nonsingular) and rearranging we obtain

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \quad (9.17)$$

where we have defined

$$N_k = \sum_{n=1}^N \gamma(z_{nk}). \quad (9.18)$$

We can interpret N_k as the effective number of points assigned to cluster k . Note carefully the form of this solution. We see that the mean $\boldsymbol{\mu}_k$ for the k^{th} Gaussian component is obtained by taking a weighted mean of all of the points in the data set, in which the weighting factor for data point \mathbf{x}_n is given by the posterior probability $\gamma(z_{nk})$ that component k was responsible for generating \mathbf{x}_n .

If we set the derivative of $\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to $\boldsymbol{\Sigma}_k$ to zero, and follow a similar line of reasoning, making use of the result for the maximum likelihood solution for the covariance matrix of a single Gaussian, we obtain

Section 2.3.4

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T \quad (9.19)$$

which has the same form as the corresponding result for a single Gaussian fitted to the data set, but again with each data point weighted by the corresponding posterior probability and with the denominator given by the effective number of points associated with the component.

Finally, we maximize $\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to the mixing coefficients π_k . Here we must take account of the constraint (9.9), which requires the mixing coefficients to sum to one. This can be achieved using a Lagrange multiplier and maximizing the following quantity

Appendix E

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \lambda \left(\sum_{k=1}^K \pi_k - 1 \right) \quad (9.20)$$

which gives

$$0 = \sum_{n=1}^N \frac{\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} + \lambda \quad (9.21)$$

where again we see the appearance of the responsibilities. If we now multiply both sides by π_k and sum over k making use of the constraint (9.9), we find $\lambda = -N$. Using this to eliminate λ and rearranging we obtain

$$\pi_k = \frac{N_k}{N} \quad (9.22)$$

so that the mixing coefficient for the k^{th} component is given by the average responsibility which that component takes for explaining the data points.

It is worth emphasizing that the results (9.17), (9.19), and (9.22) do not constitute a closed-form solution for the parameters of the mixture model because the responsibilities $\gamma(z_{nk})$ depend on those parameters in a complex way through (9.13). However, these results do suggest a simple iterative scheme for finding a solution to the maximum likelihood problem, which as we shall see turns out to be an instance of the EM algorithm for the particular case of the Gaussian mixture model. We first choose some initial values for the means, covariances, and mixing coefficients. Then we alternate between the following two updates that we shall call the E step

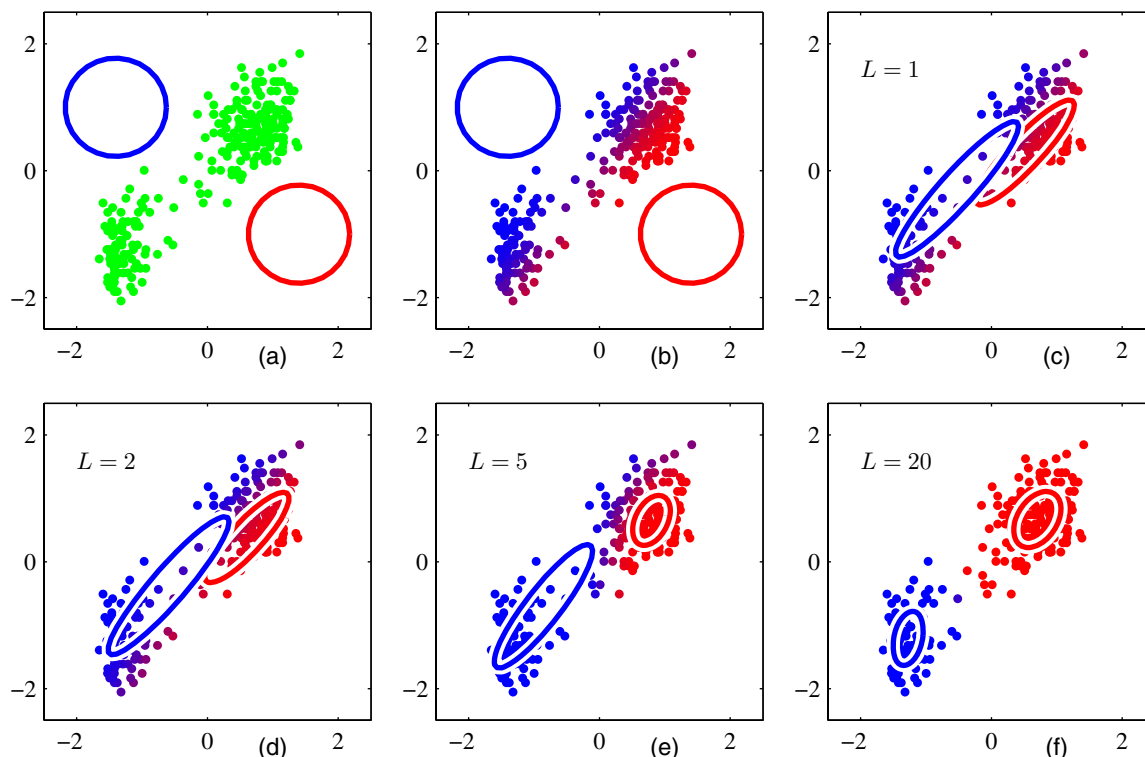


Figure 9.8 Illustration of the EM algorithm using the Old Faithful set as used for the illustration of the K -means algorithm in Figure 9.1. See the text for details.

and the M step, for reasons that will become apparent shortly. In the *expectation* step, or E step, we use the current values for the parameters to evaluate the posterior probabilities, or responsibilities, given by (9.13). We then use these probabilities in the *maximization* step, or M step, to re-estimate the means, covariances, and mixing coefficients using the results (9.17), (9.19), and (9.22). Note that in so doing we first evaluate the new means using (9.17) and then use these new values to find the covariances using (9.19), in keeping with the corresponding result for a single Gaussian distribution. We shall show that each update to the parameters resulting from an E step followed by an M step is guaranteed to increase the log likelihood function. In practice, the algorithm is deemed to have converged when the change in the log likelihood function, or alternatively in the parameters, falls below some threshold. We illustrate the EM algorithm for a mixture of two Gaussians applied to the rescaled Old Faithful data set in Figure 9.8. Here a mixture of two Gaussians is used, with centres initialized using the same values as for the K -means algorithm in Figure 9.1, and with precision matrices initialized to be proportional to the unit matrix. Plot (a) shows the data points in green, together with the initial configuration of the mixture model in which the one standard-deviation contours for the two

Gaussian components are shown as blue and red circles. Plot (b) shows the result of the initial E step, in which each data point is depicted using a proportion of blue ink equal to the posterior probability of having been generated from the blue component, and a corresponding proportion of red ink given by the posterior probability of having been generated by the red component. Thus, points that have a significant probability for belonging to either cluster appear purple. The situation after the first M step is shown in plot (c), in which the mean of the blue Gaussian has moved to the mean of the data set, weighted by the probabilities of each data point belonging to the blue cluster, in other words it has moved to the centre of mass of the blue ink. Similarly, the covariance of the blue Gaussian is set equal to the covariance of the blue ink. Analogous results hold for the red component. Plots (d), (e), and (f) show the results after 2, 5, and 20 complete cycles of EM, respectively. In plot (f) the algorithm is close to convergence.

Note that the EM algorithm takes many more iterations to reach (approximate) convergence compared with the K -means algorithm, and that each cycle requires significantly more computation. It is therefore common to run the K -means algorithm in order to find a suitable initialization for a Gaussian mixture model that is subsequently adapted using EM. The covariance matrices can conveniently be initialized to the sample covariances of the clusters found by the K -means algorithm, and the mixing coefficients can be set to the fractions of data points assigned to the respective clusters. As with gradient-based approaches for maximizing the log likelihood, techniques must be employed to avoid singularities of the likelihood function in which a Gaussian component collapses onto a particular data point. It should be emphasized that there will generally be multiple local maxima of the log likelihood function, and that EM is not guaranteed to find the largest of these maxima. Because the EM algorithm for Gaussian mixtures plays such an important role, we summarize it below.

EM for Gaussian Mixtures

Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters (comprising the means and covariances of the components and the mixing coefficients).

1. Initialize the means μ_k , covariances Σ_k and mixing coefficients π_k , and evaluate the initial value of the log likelihood.
2. **E step.** Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \mu_j, \Sigma_j)}. \quad (9.23)$$

3. **M step.** Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \quad (9.24)$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^T \quad (9.25)$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \quad (9.26)$$

where

$$N_k = \sum_{n=1}^N \gamma(z_{nk}). \quad (9.27)$$

4. Evaluate the log likelihood

$$\ln p(\mathbf{X} | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\} \quad (9.28)$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

9.3. An Alternative View of EM

In this section, we present a complementary view of the EM algorithm that recognizes the key role played by latent variables. We discuss this approach first of all in an abstract setting, and then for illustration we consider once again the case of Gaussian mixtures.

The goal of the EM algorithm is to find maximum likelihood solutions for models having latent variables. We denote the set of all observed data by \mathbf{X} , in which the n^{th} row represents \mathbf{x}_n^T , and similarly we denote the set of all latent variables by \mathbf{Z} , with a corresponding row \mathbf{z}_n^T . The set of all model parameters is denoted by $\boldsymbol{\theta}$, and so the log likelihood function is given by

$$\ln p(\mathbf{X} | \boldsymbol{\theta}) = \ln \left\{ \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}) \right\}. \quad (9.29)$$

Note that our discussion will apply equally well to continuous latent variables simply by replacing the sum over \mathbf{Z} with an integral.

A key observation is that the summation over the latent variables appears inside the logarithm. Even if the joint distribution $p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$ belongs to the exponential