An Introduction to Probabilistic Graphical Models

Michael I. Jordan
University of California, Berkeley

June 30, 2003

Chapter 10

Mixtures and conditional mixtures

In this chapter we begin the study of models with *latent* or *hidden* variables. Latent variables are simply random variables whose values are not specified in the observed data—in the graphical model formalism these variables are the unshaded nodes. Our focus in the current chapter is the simple case of latent variables that can take one of a finite set of values.

Let us take a moment to pose the question of why would one include a node in a model if the value of that node cannot be observed in the data. Shouldn't we include variables in our model only if their values can be observed? One answer to this question is philosophical—surely much human knowledge involves explaining observed data in terms of unobserved concepts.[1] For example, we often introduce *distinctions* into our reasoning in order to simplify relationships between observables. Thus a doctor may group patients into those with a certain "syndrome" and those without, and this grouping may make it easier to understand the relationships between observed symptoms. A biologist may wish to group animals into distinct species, because it may be easier to explain behavioral or physiological patterns within each species than to explain such patterns without the help of the distinction. Although such distinctions may exist only in the mind of the doctor or biologist, at least at the outset, their utility for modeling the data may provide the motivation for further study in which one tries to uncover a "real" physical or biological interpretation of the distinction.

Viewing a "distinction" as a discrete random variable ranging over a finite, unordered set of values leads to the mixture models studied in the current chapter.

In Chapter **??** we study continuous latent variable models in which the latent variable parameterizes a k-dimensional subspace of the d-dimensional input space; here the latent variable achieves a "dimensionality reduction." Chapter 12 and Chapter 15 discuss models in which latent variables are used in the time series setting to summarize past data; that is, the latent variables are "state variables." In all of these cases, and in others that we will meet, the general idea is the same—models with latent variables can often be simpler than models without latent variables. In statistical terms we often find that we can get by with fewer parameters using a latent variable model, or we can avail ourselves of simple parametric distributions that have advantageous com-

[1] One should not, however, expect the philosophers to have agreed on this. Indeed, the philosophical school of *logical positivism* explicitly denied the meaningfulness of using unobservable concepts in scientific reasoning.

putational or analytical properties. We will not attempt to define "simplicity" more rigorously for now—that is the task of Chapter 26. Instead we proceed by example, describing latent variable models that have been shown to be useful in practice.

In this chapter we discuss two kinds of models based on discrete latent variables—*unconditional mixture models* and *conditional mixture models*. Roughly speaking, unconditional mixture models are used to solve density estimation problems, whereas conditional mixture models are used to solve regression and classification problems. One useful perspective to take on the latent variable methodology in both of these kinds of problems is that it allows us to break problems into subproblems. Thus, in unconditional mixture modeling, for each value of the latent variable we obtain a (presumably simpler) density estimation subproblem. In conditional mixture modeling, for each value of the latent variable we obtain a (presumably simpler) regression or classification subproblem. In general, mixture modeling can be viewed as a "divide-and-conquer" approach to statistical modeling.

10.1 Unconditional mixture models

We begin by discussing unconditional mixture models. While regression and classification models require the observation of (X, Y) pairs, unconditional mixture models make do with observations of X alone.

As we discussed in Chapter 5, mixture models can be used to solve density estimation problems, allowing us to answer questions about whether query vectors are "typical" or "untypical." This has many applications, including the detection of outliers and the design of algorithms for data compression. Note that in such applications we are not necessarily interested in identifying or interpreting the structure of the probability distribution generating the data; rather we are simply interested in a flexible model that allows us to obtain a good estimate of the probability density.

In other problems, however, we may have a more "structural" interest in the mixture model. In particular, as we discussed in Chapter 5, we may wish to take the point of view that there are "subpopulations" underlying the data. In this setting, mixture modeling is closely linked to classification, in particular to the generative classification models discussed in Chapter 7. Indeed, treating the class label of a generative classification model as a latent variable converts the model into a mixture model. Reserving the term *classification* for the setting in which the labels are in fact observed, we use the term *clustering* for the problem of inferring the labels of data points when such labels are absent in the data. Mixture models provide a popular and widely used methodology for clustering.

In Chapter 5 we presented the following general formulation of an unconditional mixture model (see Figure 10.1). Let Z represent a multinomial random variable with components Z^i. We have:

$$\begin{aligned} p(x \mid \theta) &= \sum_i p(Z^i = 1 \mid \pi_i) p(x \mid Z^i = 1, \theta_i) && (10.1) \\ &= \sum_i \pi_i p(x \mid Z^i = 1, \theta_i). && (10.2) \end{aligned}$$

where $\theta = (\pi_1, \ldots, \pi_K, \theta_1, \ldots, \theta_K)$ and where the π_i are constrained to sum to one. Recall the

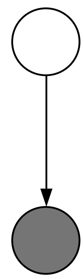

Figure 10.1: A mixture model represented as a graphical model. The latent variable Z is a multinomial node taking on one of K values.

terminology—the parameters π_i are referred to as *mixing proportions* and the densities $p(x \mid Z^i = 1, \theta_i)$ are referred to as *mixture components*.

10.1.1 Gaussian mixture models

Let us begin by discussing the important special case of the Gaussian mixture model. In this model the mixture components are Gaussian distributions with parameters $\theta_i \triangleq (\mu_i, \Sigma_i)$. Note that we allow the covariance to vary across the mixture components. One can also consider models in which the covariance matrices are constrained to be equal.

From Eq. (10.2) we obtain the following probability model for a Gaussian mixture:

$$p(x \mid \theta) = \sum_i \pi_i \frac{1}{(2\pi)^{m/2}|\Sigma_i|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)\right\}. \tag{10.3}$$

We will also write this as:

$$p(x \mid \theta) = \sum_i \pi_i \mathcal{N}(x \mid \mu_i, \Sigma_i) \tag{10.4}$$

to simplify notation.

Figure 10.2 shows a simple illustration of a Gaussian mixture model, together with a sample from the marginal distribution.

Let us calculate the probability of the latent variable Z conditioned on the observed variable X. This calculation is of obvious interest if we wish to use the mixture model in the clustering setting—the conditional probability of Z can be used to assign X to one of the clusters. We will also find that this conditional probability plays an important role in parameter estimation.

We let τ^i denote the conditional probability that the ith component of Z is equal to one. From Bayes rule we have:

$$\tau^i \triangleq p(Z^i = 1 \mid x, \theta) \tag{10.5}$$

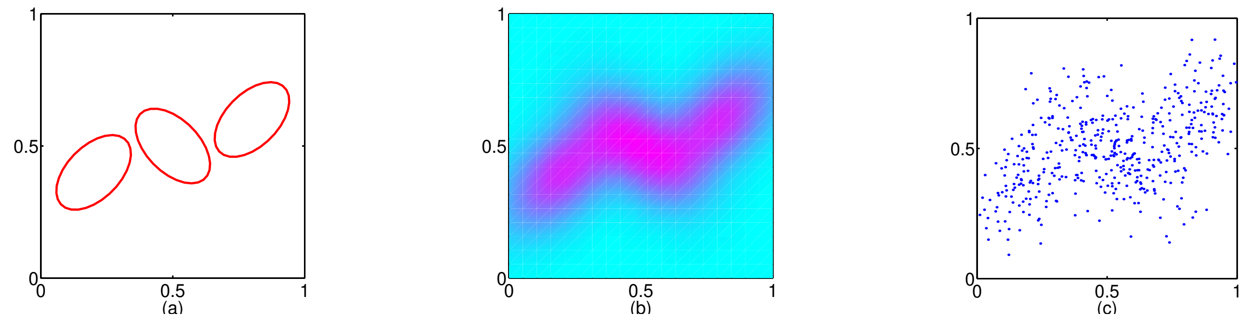

Figure 10.2: Illustration of a mixture of 3 Gaussians in a two-dimensional space showing (a) contours representing one standard deviation for each of the mixture components, (b) the marginal probability density of the mixture distribution, and (c) a sample of 500 points drawn from the marginal distribution.

$$= \frac{p(x \mid Z^i = 1, \theta_i) p(Z^i = 1 \mid \pi_i)}{p(x \mid \theta)} \tag{10.6}$$

$$= \frac{\pi_i \mathcal{N}(x, \mu_i \mid \Sigma_i)}{\sum_j \pi_j \mathcal{N}(x, \mu_j \mid \Sigma_j)} \tag{10.7}$$

Note the relationship to the generative classification models of Section 7.2. In particular, if we let the Σ_k be equal, then the quadratic terms cancel and we obtain the linear-softmax function as in that section.

It is common to refer to π_i as a "prior probability" and τ^i as a "posterior probability." This is a convenient terminology that reflects the fact that these probabilities are linked via Bayes rule. Please note, however, that the use of this terminology is unrelated to whether or not we use Bayesian methods to estimate the parameters θ. Indeed, in this chapter our focus will be maximum likelihood estimation.

Let us now consider the problem of estimating θ from an IID set of observations $\mathcal{D} = \{x_n : n = 1, \ldots, N\}$. The model is shown in Figure 10.3, where we see that each data point x_n is accompanied by a multinomial latent variable Z_n that represents the "assignment" of x_n to one of the mixture components. We form the log likelihood:

$$l(\theta \mid \mathcal{D}) = \sum_n \log p(x_n \mid \theta) \tag{10.8}$$

$$= \sum_n \log \sum_i \pi_i \mathcal{N}(x_n \mid \mu_i, \Sigma_i) \tag{10.9}$$

by taking the log of the product of N copies of the probability model in Eq. (10.3). Note the disconcerting fact that the logarithm stops in front of the sum. In all of the models that we have considered up until now, the logarithm acted directly on the basic probability distributions in our model, which, given the exponential family distributions that we have worked with, yielded simple expressions such as squared error or cross entropy. Here the likelihood is a marginal probability.

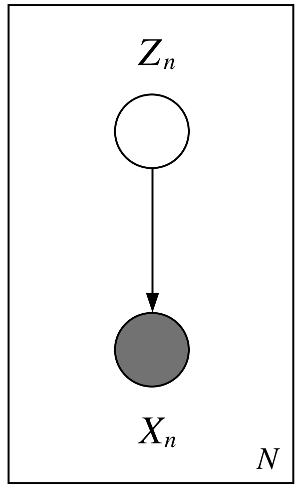

Figure 10.3: The mixture model under an IID sampling assumption.

This prevents the logarithm from acting directly on the component probability distributions and leaves us with a tangled nonlinear function to maximize.

One approach to maximizing this likelihood is to hand Eq. (10.9) to a nonlinear optimization algorithm such as conjugate gradient or Newton-Raphson. In Appendix XXX we provide some details regarding this approach. Our main focus, however, will be on an alternative approach to maximizing the likelihood known as the Expectation-Maximization (EM) algorithm. This algorithm is applicable far beyond the Gaussian mixture setting; indeed, it is applicable to arbitrary graphical models with latent variables. Its important virtue in the graphical model setting is that it allows us to take full advantage of the graphical structure underlying the likelihood; in particular, we will be able to exploit the inference algorithms discussed in Chapter 3 and Chapter 17. By relating the problem of parameter estimation and the problem of efficient inference, the EM algorithm brings together two of our major themes. It will play an important role throughout the book.

In this chapter we provide a heuristic introduction to the EM algorithm for Gaussian mixture models. Chapter 11 provides a rigorous derivation of EM, not only for Gaussian mixture models, but for the general case.

10.1.2 The K-means algorithm

To motivate the EM algorithm for Gaussian mixtures, it is useful to step briefly outside of the Gaussian mixture framework to consider an even simpler approach to clustering.

Recall that we have a set of observations $\mathcal{D} = \{x_n : n = 1, \ldots, N\}$. Our goal is to group the data points into a set of K clusters, where we suppose that the value of K is given.

The *K-means algorithm* represents each cluster with a single vector, which we refer to as a "cluster mean." The basic idea is to assign data points to clusters by finding the nearest cluster mean and assigning the data point to that cluster.

Note that we do not have a probabilistic model in mind, so "cluster mean" is perhaps a poor terminology. "Cluster centroid" is better; the idea is that if we knew which data points were assigned to the ith cluster, then the cluster mean would be the centroid (the sample average) of those data points.

We are faced with a "chicken-and-egg" problem—if we knew the assignments we could find the means, or if we knew the means we could find the assignments. The basic idea of the K-means algorithm is to make an initial guess for one of these quantities (the means) and iterate back and forth.

The algorithm maintains two kinds of variables—means and assignments. Let μ_i denote the cluster mean for the ith cluster. For each data point x_n let z_n be an indicator vector that represents the assignment of x_n to one of the clusters. Thus, if the x_n is assigned to the ith cluster, we set the component z_n^i equal to one, and all other components of z_n equal to zero.

The K-means algorithm begins by making some initial assignments for the μ_i, for example taking the μ_i to be given by a subset of the data vectors themselves. The algorithm then alternates between two phases. In the first phase, values for the indicator variables z_n^i are evaluated by assigning each data point x_n to the closest mean μ_i (where distance is typically measured using a simple Euclidean metric) so that, for each n,

$$z_n^i = \begin{cases} 1 & \text{if } i = \arg\min_j \|x_n - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases} \tag{10.10}$$

In the second phase, the values of the means are recomputed by taking μ_i to be equal to the sample mean of those vectors x_n which have been assigned to the i^{th} cluster:

$$\mu_i = \frac{\sum_n z_n^i x_n}{\sum_n z_n^i}. \tag{10.11}$$

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). It is easily seen that the K-means algorithm must converge after a finite number of iterations, since there are only a finite number of possible assignments for the set of discrete variables z_n^i and for each such assignment there is a unique value for the $\{\mu_i\}$.

Although we have motivated the K-means algorithm heuristically, the algorithm can also be motivated as the solution to an optimization problem. In particular, it turns out that the algorithm can be viewed as minimizing the *distortion measure* given by:

$$J = \sum_{n=1}^{N} \sum_{i=1}^{K} z_n^i \|x_n - \mu_i\|^2. \tag{10.12}$$

As we ask the reader to show in Exercise **??**, Eq. (10.10) is obtained by minimizing J with respect to z_n^i while keeping μ_i fixed, while Eq. (10.11) is obtained by minimizing J with respect to μ_i while keeping z_n^i fixed. Thus K-means can be viewed as a *coordinate descent* algorithm.

The K-means algorithm is illustrated using a simple example in Figure 10.4. The data set, shown in plot (a), consists of 40 data points in two dimensions. We now apply the K-means algorithm, with $K = 2$, using the initial mean vectors shown as the red and blue crosses in plot (b).

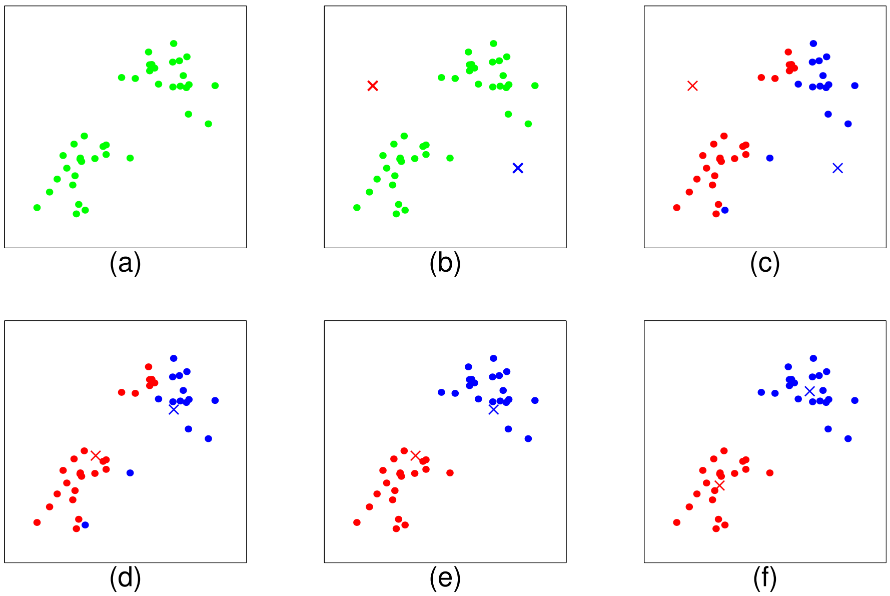

Figure 10.4: Illustration of the K-means algorithm. See the text for a full discussion.

Note that this is in fact a particularly poor initialization and has been chosen in order to provide a clear illustration of the operation of the algorithm. A better initialization would involve selecting random data points as the initial means, and would typically give faster convergence. The first stage of the algorithm involves assigning each data point to one of the two clusters according to its distance to each of the means. This results in the partitioning of the data set shown in plot (c), in which each data point has been colored according to the cluster (red or blue) to which it is assigned. Next we re-compute the mean vectors using the current partitioning, so that the blue mean is re-assigned to the mean of the blue-colored data points, and similarly for the red mean, giving the new estimates for the means shown in plot (d). The two phases of the algorithm continue alternately, with repartioning shown in plot (e), re-assignment in plot (f). On the next re-partitioning the assignment of data points does not change and hence the algorithm has converged.

10.1.3 The EM algorithm

Let us return to the probabilistic framework of mixture models. Adopting the language of the K-means algorithm, let us view the latent variables Z_n as "assignment variables." These are random variables in the mixture model setting, reflecting our uncertainty about the cluster assignments.

If the Z_n were observed we would have a classification problem in which each data point X_n is assigned a "class label." The estimate of the mean of the ith Gaussian would simply be the sample mean for the data points in the ith class (cf. Eq. (7.20)):

$$\hat{\mu}_i = \frac{\sum_n z_n^i x_n}{\sum_n z_n^i}. \tag{10.13}$$

This is identical to the K-means update formula, but here we are interpreting the variables z_n^i not as quantities to be manipulated by our algorithm, but rather as observed values of random variables.

Of course, we do not know the values of the Z_n variables. Our approach will be to replace these values with their conditional expectations, conditioning on the data (the x_n values). Recall that we use the notation τ_n^i for these conditional expectations.[2] We try the following idea—let us replace z_n^i by τ_n^i in Eq. (10.13):

$$\hat{\mu}_i = \frac{\sum_n \tau_n^i x_n}{\sum_n \tau_n^i}. \tag{10.14}$$

Thus, we have replaced a sample mean with a weighted sample mean. Each data point x_n contributes to the estimate of the ith mean in proportion to its posterior probability τ_n^i. The quantity τ_n^i is often referred to as a "soft assignment," a natural terminology both from the point of view of Eq. (10.14) and the definition of τ_n^i in Eq. (10.7).

We still have a chicken-and-egg problem, however. As seen in Eq. (10.7), the posterior probabilities τ_n^i depend on the parameter estimates, which, according to Eq. (10.14), depend on the posterior probabilities.

Once again, the way out of this chicken-and-egg problem is to start with an initial guess (for the parameters) and to iterate. Given a set of parameters we calculate the posterior probabilities. Given a set of posterior probabilities, we compute new parameter estimates. This is the basic structure of the EM algorithm for Gaussian mixtures.

To clarify, let us augment our notation for τ_n^i to include reference to the iteration number t:

$$\tau_n^{i(t)} = \frac{\pi_i^{(t)} \mathcal{N}(x_n \,|\, \mu_i^{(t)}, \Sigma_i^{(t)})}{\sum_j \pi_j^{(t)} \mathcal{N}(x_n \,|\, \mu_j^{(t)}, \Sigma_j^{(t)})}, \tag{10.15}$$

where we have also indexed the parameter estimates with a superscript to indicate the iteration number. We now define update equations for all of the parameters—the mixing proportions, the means and the covariance matrices. Motivated by the K-means algorithm we have the following formula for the means:

$$\mu_i^{(t+1)} = \frac{\sum_n \tau_n^{i(t)} x_n}{\sum_n \tau_n^{i(t)}}. \tag{10.16}$$

For the covariance matrices we use an analogous formula:

$$\Sigma_i^{(t+1)} = \frac{\sum_n \tau_n^{i(t)} (x_n - \mu_i^{(t+1)})(x_n - \mu_i^{(t+1)})^T}{\sum_n \tau_n^{i(t)}}. \tag{10.17}$$

defining the update as a weighted sample covariance, with the posterior probabilities again serving as weights. Finally, viewing $\tau_n^{i(t)}$ as a "soft assignment" of data point x_n to cluster i, it is natural to

[2] We use the elementary fact that conditional expectations and conditional probabilities are the same for binary-valued variables: $E[Z_n^i \,|\, x_n] = p(Z_n^i = 1 \,|\, x_n)$.

estimate π_i as the sum of these assignments across the data, divided by the number of data points:

$$\pi_i^{(t+1)} = \frac{1}{N} \sum_n \tau_n^{i(t)}. \tag{10.18}$$

Note that if we sum these estimates $\pi_i^{(t+1)}$ with respect to i we obtain one; thus our "soft counting" has not undercounted or overcounted.

Equations 10.15, 10.18, 10.16, and 10.17 define the EM algorithm for Gaussian mixtures.

The first phase of the algorithm—the calculation of the posterior probability in Eq. (10.15)—is generally referred to as the "Expectation step," or "E step." The second phase of the algorithm—the parameter updates in Equations 10.18, 10.16, and 10.17—is generally referred to as the "Maximization step," or "M step." The explanation for this choice of terminology will be provided in Chapter 11.

In Figure 10.5 we illustrate the EM algorithm applied to a mixture of Gaussians using the same data set, shown in plot (a), as used to illustrate the K-means algorithm in Figure 10.4. Here a mixture of 2 Gaussians is used, with centers initialized using the same values as for the K-means algorithm, and with covariance matrices initialized to be proportional to the unit matrix. Contours of 1 standard deviation for each of the Gaussian components are shown in plot (b). In plot (c) we show the result of applying the initial E-step, in which points have been colored according to the posterior probabilities for the two components, such that the color ranges from blue to red as the probability $P(\text{blue}|x_n)$ ranges from 1 to 0. We see in plot (c) that some points have a significant probability for belonging to either cluster and so appear purple. Plot (d) shows the result of the first M-step. We see that the mean of, say, the blue Gaussian is 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 moves to the mean of the blue ink. Similarly the covariance of the blue Gaussian becomes the sample covariance of the blue ink, with analogous results for the red component. Subsequent plots show the situation after various numbers L of complete EM cycles. In plot (i) the model is close to the final, converged state. 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 which 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 proportions can be set to the fractions of data points assigned to the respective clusters.

10.1.4 Necessary conditions

Although we have defined a simple, intuitively appealing algorithm, it is not yet clear what relationship this algorithm has to the quantity that we are trying to maximize, the log likelihood in Eq. (10.9). In this section and the following section, we take initial steps toward working out this relationship, and in so doing providing a more rigorous justification of the EM algorithm. The full justification will appear in Chapter 11, where we show that the EM algorithm—like the K-means algorithm—is a form of coordinate ascent.

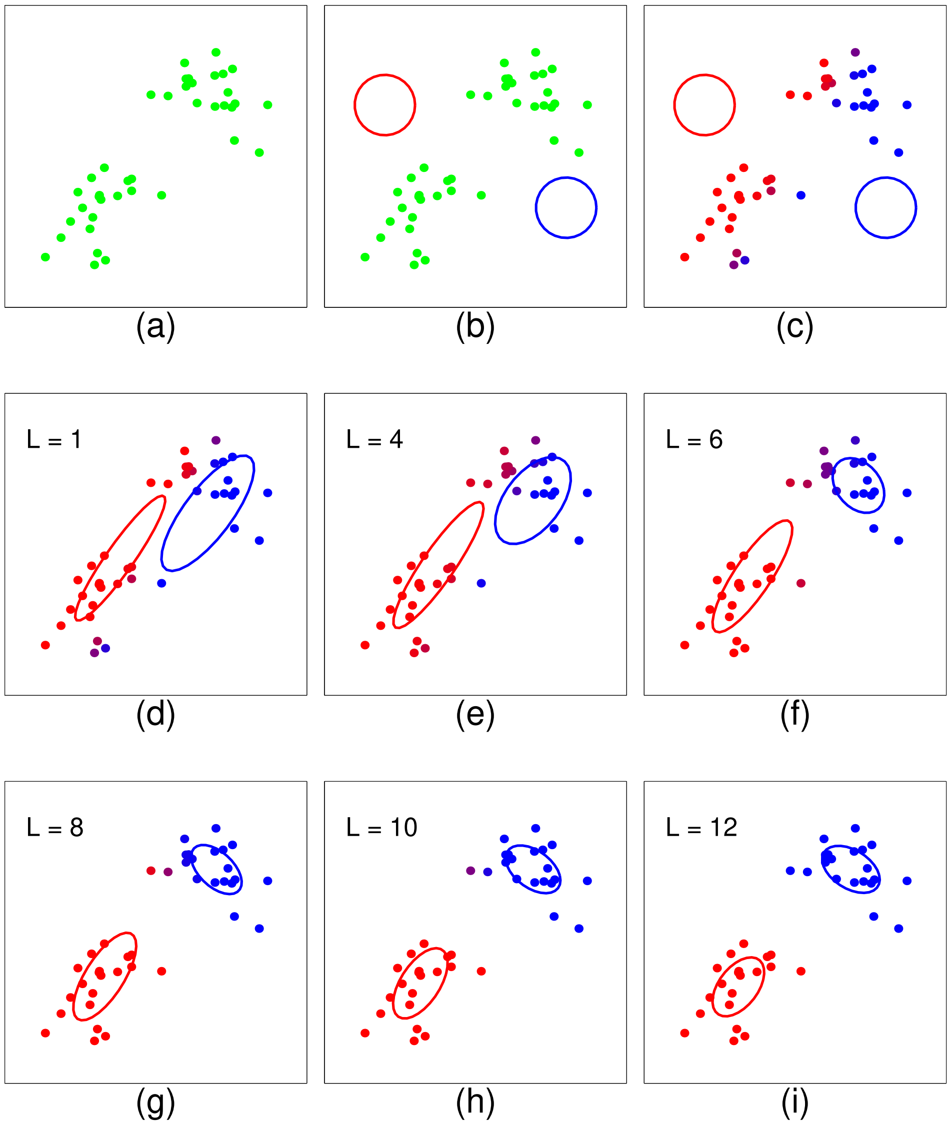

Figure 10.5: Illustration of the EM algorithm using the same data set as used for the illustration of the K-means algorithm in Figure 10.4. The value of L denotes the number of complete EM cycles. See the text for a full discussion.

In this section we write down a set of equations characterizing the stationary points of the log likelihood function. We show that the stationary points can be viewed as fixed points of the EM iteration.

We need to obtain the derivatives of l with respect to the parameters. Let us first take the derivative with respect to μ_i:

$$\begin{aligned}
\frac{\partial l}{\partial \mu_i} &= \frac{\partial}{\partial \mu_i}\left\{\sum_n \log \sum_i \pi_i \mathcal{N}(x_n \mid \mu_i, \Sigma_i)\right\} && (10.19)\\
&= \sum_n \frac{\pi_i \mathcal{N}(x_n \mid \mu_i, \Sigma_i)}{\sum_j \pi_j \mathcal{N}(x_n \mid \mu_j, \Sigma_j)} \frac{\partial}{\partial \mu_i} \log \mathcal{N}(x_n \mid \mu_i, \Sigma_i) && (10.20)\\
&= \sum_n \tau_n^i \frac{\partial}{\partial \mu_i} \log \mathcal{N}(x_n \mid \mu_i, \Sigma_i) && (10.21)\\
&= \sum_n \tau_n^i \Sigma_i^{-1}(x_n - \mu_i). && (10.22)
\end{aligned}$$

Setting to zero yields

$$\mu_i = \frac{\sum_{n=1}^N \tau_n^i x_n}{\sum_{n=1}^N \tau_n^i} \tag{10.23}$$

at a stationary point of the log likelihood.

A very similar calculation yields the following conditions for the covariance matrices:

$$\Sigma_i = \frac{\sum_{n=1}^N \tau_n^i (x_n - \mu_i)(x_n - \mu_i)^T}{\sum_{n=1}^N \tau_n^i} \tag{10.24}$$

and the mixing proportions:

$$\pi_i = \frac{1}{N} \sum_{n=1}^N \tau_n^i, \tag{10.25}$$

where in the latter case we use Lagrange multipliers.

These equations do not of course constitute an explicit solution since the posterior probabilities are themselves functions of the parameters, and so Equations 10.25, 10.23 and 10.24 represent a system of coupled, nonlinear equations. We can, however, attempt to solve these equations iteratively. In particular, given a parameter vector $\theta^{(t)}$, we plug into the right-hand side of Equations 10.25, 10.23 and 10.24 and obtain an updated parameter vector, which we define to be $\theta^{(t+1)}$.

Comparing to Equations 10.16, 10.17, and 10.18, we see that we have derived the EM update equations.

While this derivation of the EM iterations is perhaps preferable to our earlier heuristic arguments, it is still rather heuristic, leaving us with a number of questions regarding convergence. Moreover, the derivation provides us with little insight—the key quantity, the posterior probability τ_n^i, emerges somewhat mysteriously from the algebra. To develop a deeper understanding and to apply the EM algorithm to more general graphical models, we will need some new concepts.

10.1.5 The expected complete log likelihood

In this section we derive the EM equations for the Gaussian mixture model simply and systematically, by introducing a key player in the EM story—the *expected complete log likelihood*. The full treatment of the role played by the expected complete log likelihood in the EM algorithm will have to wait for Chapter 11, but we provide some initial intuition in this section, paving the way for the general presentation in Chapter 11.

To introduce the key idea, let us pretend for a moment that we are able to observe the latent variables Z_n. This is a pretense, but it will turn out to be a useful pretense. In particular, let us define a (fictional) data set $\mathcal{D}_c = \{(x_n, z_n) : n = 1, \ldots, N\}$ that we refer to as the *complete data.*

If we were to actually have such a data set, we would define the following likelihood, which we refer to as the *complete log likelihood*:

$$\begin{aligned}
l_c(\theta \,|\, \mathcal{D}_c) &= \sum_n \log p(x_n, z_n \,|\, \theta) && (10.26)\\
&= \sum_n \log \prod_i \left[\pi_i \mathcal{N}(x_n \,|\, \mu_i, \Sigma_i)\right]^{z_n^i} && (10.27)\\
&= \sum_n \sum_i z_n^i \log \left[\pi_i \mathcal{N}(x_n \,|\, \mu_i, \Sigma_i)\right] && (10.28)
\end{aligned}$$

Note the difference between this log likelihood and the original log likelihood for our problem, which we repeat here for convenience:

$$l(\theta \,|\, \mathcal{D}) = \sum_n \log \sum_i \pi_i \mathcal{N}(x_n \,|\, \mu_i, \Sigma_i). \qquad (10.29)$$

In the latter log likelihood, the logarithm is outside of the summation over i, which, as we have remarked before, reflects the fact that the likelihood is a marginal probability. The complete log likelihood, on the other hand, is not a marginal probability, and thus the logarithm is inside the sum. This logarithm acts on the probabilities π_i and $\mathcal{N}(x_n \,|\, \mu_i, \Sigma_i)$, leading to the simple maximum likelihood formulas for generative classification that we studied in Chapter 7.

Of course the Z_n variables are not observed. The next step is the key one—as in our earlier discussion let us treat the values z_n in the complete log likelihood as random variables Z_n and take expectations. In calculating these expectations we condition on the observed data x_n, also fixing a particular parameter vector $\theta^{(t)}$. Using the operator notation $\langle \cdot \rangle_{\theta^{(t)}}$ to denote these conditional expectations, we define an important quantity known as the *expected complete log likelihood*:

$$\begin{aligned}
\langle l_c(\theta \,|\, \mathcal{D}_c) \rangle_{\theta^{(t)}} &= \left\langle \sum_n \sum_i Z_n^i \log \left[\pi_i \mathcal{N}(x_n \,|\, \mu_i, \Sigma_i)\right] \right\rangle_{\theta^{(t)}} && (10.30)\\
&= \sum_n \sum_i \left\langle Z_n^i \right\rangle_{\theta^{(t)}} \log \left\{\pi_i \mathcal{N}(x_n \,|\, \mu_i, \Sigma_i)\right\} && (10.31)\\
&= \sum_n \sum_i \tau_n^{i(t)} \log \left\{\pi_i \mathcal{N}(x_n \,|\, \mu_i, \Sigma_i)\right\} && (10.32)
\end{aligned}$$

Comparing Eq. (10.28) and Eq. (10.32), we see that the expected complete log likelihood is obtained from the complete log likelihood by replacing the fictional "observations" z_n^i with the posterior probabilities τ_n^i, where the latter are evaluated using the parameter vector $\theta^{(t)}$.

In general we define the E step of the EM algorithm to be the "calculation of the expected complete log likelihood." In the Gaussian mixture problem this simply reduces to calculating the posterior probabilities τ_n^i, and it may not be clear why we need the fancier language. In problems with multiple latent variables, however, there are generally interactions to account for, and the preferred method for defining the E step in the general setting is to calculate the expected complete log likelihood.

Moreover, the preferred method for obtaining the M step of an EM algorithm is to maximize the expected complete log likelihood with respect to the parameters. We will explain why this is the case in Chapter 11, but in the meantime let us treat it as a recipe and verify that we obtain the M step updates in Equations 10.18, 10.16, and 10.17 from the expected complete log likelihood in Eq. (10.32).

Let us first consider the update for the means. Collecting together the terms in Eq. (10.32) that depend on μ_i, and denoting the result as $J(\mu_i)$, we obtain:

$$J(\mu_i) = -\frac{1}{2}\sum_n \tau_n^{i(t)}(x_n - \mu_i)^T \Sigma_i^{-1}(x_n - \mu_i). \tag{10.33}$$

We see that we have a weighted least-squares problem. Calculating the derivative of $J(\mu_i)$ with respect to μ_i and setting to zero yields:

$$\mu_i^{(t+1)} = \frac{\sum_n \tau_n^{i(t)} x_n}{\sum_n \tau_n^{i(t)}}, \tag{10.34}$$

which is Eq. (10.16).

Similarly, collecting together the terms that reference the covariance matrix Σ_i, we have:

$$J(\Sigma_i) = -\frac{1}{2}\sum_n \tau_n^{i(t)} \left\{\log|\Sigma_i| + (x_n - \mu_i)^T \Sigma_i^{-1}(x_n - \mu_i)\right\}. \tag{10.35}$$

This is a weighted variant of the problem of estimating the covariance matrix of a Gaussian. Taking the derivative with respect to Σ_i and setting to zero yields:

$$\Sigma_i^{(t+1)} = \frac{\sum_n \tau_n^{i(t)}(x_n - \mu_i^{(t+1)})(x_n - \mu_i^{(t+1)})^T}{\sum_n \tau_n^{i(t)}}. \tag{10.36}$$

which is Eq. (10.17).

Finally, the terms in the expected complete log likelihood that reference π are:

$$J(\pi) = \sum_n \sum_i \tau_n^{i(t)} \log \pi_i. \tag{10.37}$$

Adding a Lagrangian term to account for the constraint that the π_i sum to one, taking derivatives and setting to zero yields:

$$\pi_i^{(t+1)} = \frac{1}{N}\sum_n \tau_n^{i(t)}, \tag{10.38}$$

Figure 10.6:

Figure 10.7:

which is Eq. (10.18).

Although this derivation of the EM equations is no more intuitive than our earlier work, it has the virtue of yielding a simple algebraic recipe, both for the E step and the M step. It is also more general than our earlier work; indeed, the derivation that we have carried out here will extend readily to arbitrary graphical models. The key concepts that underly the usefulness of this approach are: (1) the decoupled form of the complete log likelihood, and (2) the linearity of the expectation operator.

This brief discussion suggests an important role for the "expected complete log likelihood," particularly in providing a simple method for deriving EM update equations, but it still leaves us with a number of questions. How does the maximization of the expected complete log likelihood relate to the maximization of the actual log likelihood, which is after all our goal? We have claimed that the EM algorithm is a coordinate ascent algorithm—how does the expected complete log likelihood emerge in this picture? How general is the algorithm? Will the algorithm converge? These are the topics of Chapter 11. Before turning to these general considerations, however, let us consider another application of mixture model ideas.

10.2 Conditional mixture models

The "divide-and-conquer" approach to statistical modeling applies equally well in the regression and classification domains. In this section we study conditional mixture models, which are the analogs for regression and classification of the unconditional mixture models studied thus far.[3]

Consider the data set shown in Figure 17.16, where we clearly have a nonlinearity in the mapping from X to Y. We might utilize a rich set of basis functions to allow us to capture this nonlinearity, but it may be difficult to capture the sharp kink in the middle of the figure without requiring an overly large number of basis functions to the left and right of the kink, where the function would seem to be well modeled as a simple linear function.

An alternative way to model these data is to utilize a conditional mixture. Within the conditional mixture framework, we in essence split the problem into two subproblems, each of which can be treated as a simple linear regression. We must find the parameters of these regressions, and we must also decide where to split. Assuming that we can model the splitting decision using a simple parametric model, we may be able to model the overall nonlinear dependence of Y on X using a small number of parameters.

The conditional mixture model is shown as a graphical model in Figure 17.16. The model includes nodes for the observed variables X and Y and also incorporates a node for a multinomial

[3]Conditional mixture models are also referred to as "mixture of experts" models; where the term "expert" is used to designate a regression, classification, or other generalized regression model.

latent variable Z. The response variable Y is conditioned not only on X but also on the latent variable Z. This latent variable indexes the set of possible regressions of Y on X—for each value of Z, we obtain a possibly different parameterized regression. Note moreover that there is a link from X to Z. It is this dependency that allows us to obtain different regressions in different regions of the input space.

Let us consider how to parameterize each of the nodes in the model. Z is a multinomial variable, and its parent is the input variable X. The fact that the edge between these nodes points from X to Z suggests using the ideas that we discussed in the section on discriminative classification (Section **??**) to parameterize the dependency. For example, we may use a softmax regression:

$$p(Z^i = 1 \mid x, \xi) = \frac{e^{\xi_i^T x}}{\sum_j e^{\xi_j^T x}}, \tag{10.39}$$

where $\xi = (\xi_1, \xi_2, \ldots, \xi_M$ is a parameter vector.

The node Y has X and Z as parents, and thus we have a conditional probability $p(Y \mid X, Z^i = 1, \theta_i)$. The mathematical form of this conditional probability depends on the nature of the data Y. Let us not specify a particular model at this point, but assume that we will bring to bear the machinery of generalized linear models (GLIM's). For example we could consider a binary classification model in which $p(Y \mid X, Z^i = 1, \theta_i)$ is a logistic regression model. Note that we have one such model for each value of Z.

As is usual in the regression or discriminative classification setting we treat the observations of X as fixed constants. Thus we do not incorporate a marginal probability for X into our model.

Putting together the pieces, we obtain the following model for the conditional probability of Y given X:

$$p(y \mid x, \theta) = \sum_i p(Z^i = 1 \mid x, \xi) p(y \mid Z^i = 1, x, \theta_i), \tag{10.40}$$

where $\theta = (\xi_1, \ldots, \xi_M, theta_1, \ldots, \theta_M)$. This is a conditional mixture model, where both the *mixing proportions*, $p(Z^i = 1 \mid x, \xi)$, and the *mixture components*, $p(y \mid Z^i = 1, x, \theta_i)$, are conditional probabilities—both are conditioned on $\{X = x\}$.

As in the unconditional mixtures, a key quantity in conditional mixture modeling is the posterior probability of the latent variable Z. Here the notions of "prior" and "posterior" are relative to the observation of Y; the variable X is taken to be always observed. Thus, let us define $\pi_i(x, \xi) \triangleq p(Z^i = 1 \mid x, \xi)$ as the *prior probability* of the ith mixture component, conditioned solely on X. We now define the *posterior probability* $\tau^i(x, y, \theta)$:

$$\begin{aligned}
\tau^i(x, y, \theta) &\triangleq p(Z^i = 1 \mid x, y, \xi) && (10.41)\\
&= \frac{p(Z^i = 1 \mid x, \xi) p(y \mid Z^i = 1, x, \theta_i)}{\sum_j p(Z^j = 1 \mid x, \xi) p(y \mid Z^j = 1, x, \theta_j)} && (10.42)\\
&= \frac{\pi_i(x, \xi) p(y \mid Z^i = 1, x, \theta_i)}{\sum_j \pi_j(x, \xi) p(y \mid Z^j = 1, x, \theta_j)} && (10.43)
\end{aligned}$$

where we see that the prior probability of the ith class is updated by how probable the observation $\{Y = y\}$ is under the ith model.

Figure 10.8:

10.2.1 Examples

Let us consider some specific choices for the mixture components $p(y \mid Z^i = 1, x, \theta_i)$.

Mixtures of linear regressions

For continuous variables Y it is natural to consider a mixture of linear regressions:

$$p(y \mid x, \theta) = \sum_i \pi_i(x, \xi) \mathcal{N}(y \mid \beta_i^T x, \sigma_i^2), \tag{10.44}$$

where we have allowed each linear regression to have a possibly different error variance σ_i^2.

Figure 17.16 shows a depiction of this model in the case of two mixture components. In this case the variable Z can be taken to be a binary variable and the mixing proportion $\pi(x, \xi) \triangleq p(Z = 1 \mid x, \xi)$ can be modeled using logistic regression. The logistic curve shown in the figure thus models the input-dependent probability associated with the two mixture components. For negative values of X, the logistic curve assigns small probability to $Z = 1$, thus essentially choosing the regression curve labeled $Z = 0$. For positive values of X, the logistic curve assigns large probability to $Z = 1$, thus essentially choosing the regression curve labeled $Z = 1$. The point at which the logistic function is equal to 0.5 can be viewed as the "split" point.

Let us now consider the geometric interpretation of the posterior probability. As shown in Figure 17.16, conditioning on a specific value of X leaves us with two possible conditional expectations of Y—the conditional expectations associated with each of the two regressions. We have Gaussian distributions around each of these conditional expectations, with variances σ_1^2 and σ_1^2, respectively. Thus the conditional distribution of Y given X is bimodal—we have a mixture distribution in the output space for each point in the input space. Consider now the point (x, y') in the figure. For this value of x, the prior $\pi(x, \xi)$ is large, corresponding to a choice of the regression curve labeled $Z = 1$. Moreover, the value y' has high probability under this regression model and the corresponding posterior $\tau(x, y', \xi)$ is therefore large. Consider, on the other hand, the point (x, y''). The prior for this point is the same as before. The value y'', however, has low probability under the regression model labeled $Z = 1$ and high probability under the regression model labeled $Z = 0$. The posterior $\tau(x, y', \xi)$ is therefore small, corresponding to a posterior choice of the regression model labeled $Z = 0$.

We see that the posterior probability has much the same interpretation as in the unconditional mixture model setting. That is, we can interpret the posterior probability as a "soft assignment," but here the assignment process reflects both the prior partitioning of the input space into regions, modeled by $\pi_i(x, \xi)$, and the ability of each of the component regressions to accomodate the observed output y at that given value of x, modeled by $p(y \mid Z^i = 1, x, \theta_i)$.

Mixtures of logistic regressions

We can readily extend the model of the previous section to mixtures of generalized linear models, thereby accommodating a wide variety of data types. An example is the mixture of logistic regressions:

$$p(y \mid x, \theta) = \sum_i \pi_i(x, \xi) \mu(\theta_i^T x)^y (1 - \mu(\theta_i^T x))^{1-y}, \tag{10.45}$$

where $\mu(\theta_i^T x)$ is the logistic function: $\mu(\theta_i^T x) = 1/(1 + e^{-\theta^T x})$. This model allows us to bring the divide-and-conquer approach to bear on classification problems.

The interpretation of the prior and posterior probabilities is identical in this model to the linear regression case, with the likelihood being a Bernouilli distribution rather than a Gaussian distribution.

10.2.2 Parameter estimation via the EM algorithm

At this point we have parameterized the graphical model in Figure 17.16, and the problem of maximum likelihood parameter estimation can be handled straightforwardly using the tools that we have developed in Section 10.1.3 and Section 10.1.5. Indeed the model is a good exercise of our skills. In this section we write down the log likelihood, the expected complete log likelihood, and the EM algorithm for conditional mixtures.

We assume an IID data set $\mathcal{D} = \{(x_n, y_n) : n = 1, \ldots, N\}$. The likelihood is obtained as the sum of the logarithm of the probability model in Eq. (10.40):

$$l(\theta \mid \mathcal{D}) = \sum_n \log \sum_i \pi_i(x_n, \xi) p(y_n \mid Z^i = 1, x_n, \theta_i). \tag{10.46}$$

Note the presence of the logarithm outside of the summation; as in the unconditional case, the likelihood is a marginal probability.

The "complete data" is the data set $\mathcal{D}_c = \{(x_n, z_n, y_n) : n = 1, \ldots, N\}$, where as before we imagine that we can observe the latent variable Z. The likelihood for N complete observations of the model is:

$$l(\theta \mid \mathcal{D}_c) = \prod_n \prod_i \left[\pi_i(x_n, \xi) p(y_n \mid Z^i = 1, x_n, \theta_i)\right]^{z_n^i} \tag{10.47}$$

and taking the logarithm yields the complete log likelihood:

$$l_c(\theta \mid \mathcal{D}_c) = \sum_n \sum_i z_n^i \log \left[\pi_i(x_n, \xi) p(y_n \mid Z^i = 1, x_n, \theta_i)\right], \tag{10.48}$$

where we now see the logarithm inside the summation.

We take the expectation of the complete log likelihood, where we now treat the variables Z_n^i as random variables. The expectation is taken with respect to the conditional probability distribution $p(z \mid x, y, \theta^{(t)})$. Given that the complete log likelihood is linear in z_n^i, we see that can obtain the expectation by simply computing:

$$\begin{aligned} \langle Z_n^i \rangle_{\theta^{(t)}} &= p(Z_n^i = 1 \mid x_n, y_n, \theta^{(t)}) && (10.49) \\ &= \tau^i(x_n, y_n, \theta^{(t)}). && (10.50) \end{aligned}$$

Thus we see that the E step of the EM algorithm amounts to computing the posterior probabilities $\tau^i(x_n, y_n, \theta^{(t)})$. These can be viewed as our "best guess" of the values of the latent variables Z_n, conditioned on the observed values x_n and y_n, and evaluated at the current value of the parameter vector $\theta^{(t)}$.

To summarize, the expected complete log likelihood takes the following form:

$$l_c(\theta \,|\, \mathcal{D}) = \sum_n \sum_i \tau_n^i(t) \log \left[\pi_i(x_n, \xi) p(y_n \,|\, Z^i = 1, x_n, \theta_i) \right], \tag{10.51}$$

where we write $\tau_n^{i(t)}$ for the posterior probability $\tau^i(x_n, y_n, \theta^{(t)})$, in order to simplify notation.

With the expected complete log likelihood in hand, we can now turn to the M step. Let us first considering maximizing l_c with respect to the parameters ξ. Collecting together the terms that depend on ξ, and referring to the result as $J(\xi)$, we have:

$$J(\xi) = \sum_n \sum_i \tau_n^i(t) \log \pi_i(x_n, \xi). \tag{10.52}$$

This is identical to the log likelihood for the discriminative classification problem (cf. Eq. (**??**)), where the role of the class labels in that problem (the z_n^i) is now played by the posterior probabilities (the $\tau_n^{i(t)}$). The interpretation is that we "fill in" the values of the latent variables Z_n^i with our "best guess." Based on these filled-in values we treat the problem of estimating the parameters of the conditional probability $p(Z \,|\, x, \xi)$ as a discriminative classification problem. In particular we can use the IRLS algorithm to update these parameters.

It is also straightforward to derive an M step for the parameters θ_i. Collecting together the terms in the expected complete log likelihood that depend on θ_i, and referring to the result as $J(\theta_i)$, we obtain:

$$J(\theta_i) = \sum_n \tau_n^i(t) \log p(y_n \,|\, Z^i = 1, x_n, \theta_i). \tag{10.53}$$

For generalized linear models, the log probability in this expression is the logarithm of an exponential family distribution. Each data point, (x_n, y_n), has an associated "weight," the posterior probability $\tau_n^i(t)$. Thus we have a weighted maximum likelihood problem to solve. In essence, each data point is "assigned" to one of the mixture components, and the estimation of the parameters of each mixture component is carried out using the data points assigned to that component.

In the case of a mixture of linear regressions, we obtain a set of weighted least squares problems, one for each mixture component. For a mixture of logistic regressions, we have a set of weighted cross-entropies. In general we can treat all of these problems within the IRLS framework—recall our discussion of weighted IRLS in Section **??**.

In summary, the EM algorithm for conditional mixtures takes the following form:

- (E step): Calculate the posterior probabilities $\tau_n^{i(t)}$.
- (M step): Use the IRLS algorithm to update the parameters ξ, based on data pairs $(x_n, \tau_n^{i(t)})$.
- (M step): Use the weighted IRLS algorithm to update the parameters θ_i, based on data pairs (x_n, y_n), with weights $\tau_n^{i(t)}$).

These steps iterate and, as we prove in Chapter 11, climb to a local maximum of the likelihood.

10.2.3 An on-line algorithm

We can obtain some additional insight into the conditional mixture model by developing an on-line estimation algorithm. As we discussed in Chapter 6, the problem here is to derive an update for the parameters based on a single data point.

To derive these updates we take the derivative of the log likelihood with respect to the parameters and delete the summation over n; this yields the "stochastic gradient." We omit the algebra, asking the reader to supply the details in Exercise **??**.

In the equations below, we use the notation $\mu_n^i(t)$ to denote the conditional expectation of Y given $\{X = x\}$, for the ith mixture component, the nth data point, and letting the parameter vector equal $\theta^{(t)}$. We assume that the canonical link function has been chosen.

Taking the derivative with respect to θ_i, we obtain the following update equation:

$$\theta_i^{(t+1)} = \theta_i^{(t)} + \rho\tau_n^{i(t)}(y_n - \mu_n^i(t))x_n, \tag{10.54}$$

where ρ is a step size. Similarly, taking the derivative with respect to ξ_i, we obtain:

$$\xi_i^{(t+1)} = \xi_i^{(t)} + \kappa(\tau_n^{i(t)} - \pi_i(x_n, \xi^{(t)}))x_n, \tag{10.55}$$

where κ is a step size.

Both of these equations have natural interpretations. The update of θ_i in Eq. (10.54) has the form of the LMS algorithm, but with the additional feature that the step size is modulated by the posterior probability $\tau_n^{i(t)}$. Thus, if our current "best guess" is that the nth data point should be assigned to the ith mixture component, then we update the parameters in the normal way. If, on the other hand, we do not think that the nth data point should be assigned to the ith mixture component, then the step size is near zero and the parameters are not adjusted.

The update of ξ_i in Eq. (10.54) also has an appealing interpretation. The update again takes the form of the LMS algorithm, where the error is the difference between the posterior probability and the prior probability. In essence we have a classification problem in which the prior probability is a prediction of the class label associated with the mixture components, and the posterior probability is an improved estimate of that label.

Figure 17.16 shows a depiction of these update equations for the case of a mixture of linear regressions. In this case $\mu_n^i(t) = \theta_i^{(tT)}x_n$, and the error which drives the update of θ_i is simply the difference $(y_n - \theta_i^{(tT)}x_n)$. This difference is shown in the figure for both of the regressions. Although the error is larger for the $Z = 0$ regression, the posterior probability associated with this regression is vanishingly small (it is proportional to the exponential of the negative of the square of the error). Thus, as we show in Figure 17.16(b), the parameters associated with the upper ($Z = 1$) curve change significantly, while the parameters associated with the lower ($Z = 0$) curve change little.

Given that the posterior probability associated with the upper curve is near one, the error in Eq. (10.55) has the effect of shifting the logistic curve towards the left. As we show in Figure 17.16(b), this implies that on future presentations of this data point, the prior prediction will be closer to one.

10.2.4 Hierarchical conditional mixtures and decision trees

10.3 Appendix XXX

As an alternative to the EM algorithm, we can use standard nonlinear optimization algorithms such as conjugate gradients. In this appendix we discuss this approach for unconditional mixtures, focusing on the problem of implementing the probabilistic constraints on the parameters.

In evaluating the derivatives we must take account of the requirement for the mixing proportions to satisfy $0 \leq \pi_i \leq 1$ and $\sum_i \pi_i = 1$. Similarly, the covariance matrices Σ_i must remain symmetric and positive-definite.

We can allow for the constraints on the mixing proportions π_i by expressing them as a nonlinear transformation of a corresponding set of unconstrained variables η_i. Specifically, we use the softmax transformation:

$$\pi_i = \frac{\exp(\eta_i)}{\sum_j \exp(\eta_j)}, \tag{10.56}$$

which has the property that the mixing proportions will automatically satisfy the required constraints.

In order to impose the required constraints on the covariance matrix we can represent the inverse[4] covariance matrix in terms of its Cholesky decomposition:

$$\Sigma^{-1} = A^T A \tag{10.57}$$

where A is an upper diagonal matrix, so that $A_{ij} = 0$ if $i < j$. It is easily seen that there are $d(d+1)/2$ remaining independent elements in A, corresponding to the number of independent elements in Σ. In computing the expected complete-data log likelihood we need the inverse square root of the determinant of the covariance matrix, which is given by:

$$|\Sigma|^{-1/2} = \prod_{i=1}^{d} A_{ii}. \tag{10.58}$$

The covariance matrix will be positive definite provided the diagonal elements A_{ii} are positive, which can be ensured by writing them as the exponentials of real values $A_{ii} = \exp(\alpha_i)$.

In summary, if we treat the values of η_i, A_{ij} (for $j > i$), α_i and the components of μ_i as independent, unconstrained real values, the required constraints will be met. The required derivatives of the log likelihood with respect to these unconstrained variables are then easily obtained Exercise **??**.

[4]The inverse of a positive definite symmetric matrix is also positive definite and symmetric.