Chapter 15

Mixture Models and Expectation Maximization

In the previous chapter we introduced some basic clustering techniques. However, these techniques were mainly motivated via heuristic intuitions. In this chapter, we will show how clustering techniques can be derived from a likelihood-based perspective. We will also introduce the general framework—called expectation maximization (EM)—that is used to optimize likelihood-based models that contain latent variables.

15.1 Gaussian Mixture Models

In earlier chapters, we considered the task of estimating the conditional probability $P(y|\mathbf{x})$. In other words, our goal was to learn the conditional distribution of the labels given the features. However, in the unsupervised setting we no longer have target labels y. Thus, we seek to simply model the probability density $p(\mathbf{x})$. That is, our goal is to find a model that can explain the distribution of features in our dataset.

15.1.1 Estimating densities via Gaussian mixtures

Learning a model $p(\mathbf{x})$ is typically known as *density estimation*. We have actually seen a simple example of density estimation in Chapter 4, when we computed the maximum likelihood estimate of a Gaussian distribution. However, a single Gaussian can often be insufficient. For example, in Figure 15.1 we illustrate a bi-modal dataset, where a single Gaussian is clearly incapable of approximating this distribution.

[1]Image credit: `https://www.cs.toronto.edu/~urtasun/courses/CSC411_Fall16/13_mog.pdf`.

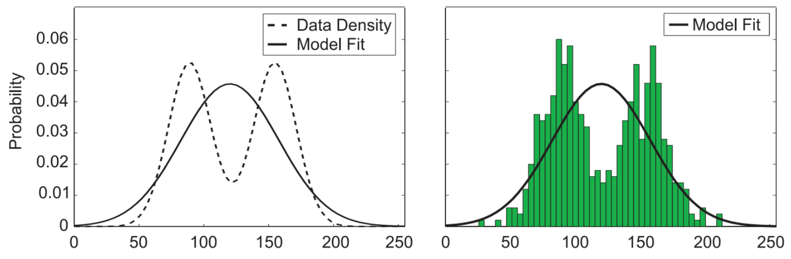

Figure 15.1: Illustration of a bi-modal distribution and an attempt to fit this distribution using a single Gaussian.[1]

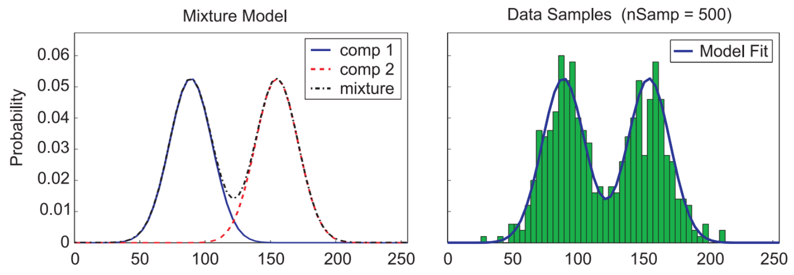

Figure 15.2: Illustration of a density fit using a mixture of two Gaussians.

Here, we will consider more complex density estimators that contain a *mixture of Gaussians*. The basic idea is that we want to model our data as follows:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \qquad \text{where} \sum_{k=1}^{K} \pi_k = 1 \tag{15.1}$$

Here, we use $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ to denote a multivariate Gaussian distribution with mean vector $\boldsymbol{\mu}_k$ and co-variance $\boldsymbol{\Sigma}_k$. The π_k terms are known as *mixing coefficients*, which determine how much each Gaussian contributes to the mixture. The parameter set for this mixture model is $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, k = 1, ..., K\}$. For example, in Figure 15.2, we show how a mixture of two Gaussians can improve upon the fit compared to the single Gaussian in Figure 15.1.

Log-likelihood of a mixture model

Given a dataset $\mathcal{D}$—which is sampled i.i.d. from the true density $p(\mathbf{x})$—the log-likelihood of this model can then be written as follows:

$$\begin{aligned}\mathcal{L}(\mathcal{D}, \Theta) &= \sum_{\mathbf{x}\in\mathcal{D}} \log\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \\ &= \sum_{\mathbf{x}\in\mathcal{D}} \log\left(\sum_{k=1}^{K} \pi_k (2\pi)^{-\frac{m}{2}} |\boldsymbol{\Sigma}_k|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right)\right).\end{aligned}$$

Note that we use $|\boldsymbol{\Sigma}_k|$ here to denote the determinant of the matrix $\boldsymbol{\Sigma}_k$. Now, one could attempt to differentiate this likelihood to optimize for the parameters $\Theta = \{\boldsymbol{\pi}_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, k = 1, ..., K\}$. However, a number of issues arise when trying to do so, and it is not actually possible to directly optimize this likelihood, even using gradient descent. For example, singularities arise when one component perfectly explains a datapoint, we need to account for the constraint that $\sum_{k=1}^{K} \pi_k = 1$, and no unique solution exists due to the arbitrariness of the cluster ordering. More generally, the summation inside the logarithm makes this a difficult function to optimize.

15.1.2 Introducing latent variables

We can make the log-likelihood of a mixture model tractable by introducing latent variables. In particular, we can assume that every point $\mathbf{x} \in \mathcal{D}$ is generated through the following process:

1. First, we sample a latent variable $z \sim \text{Categorical}(\boldsymbol{\pi})$, where $\boldsymbol{\pi} = [\pi_1, ..., \pi_K]$. In other words, we sample a mixture component $z \in \{1, ..., K\}$ for the point according to a categorical distribution with weights given by the π_k terms (i.e., $P(z = k) = \pi_k$)). We can interpret z as assigning point $\mathbf{x}$ to a particular cluster.

2. Next, we generate the point $\mathbf{x}$ by sampling from component Gaussian $\mathcal{N}(\boldsymbol{\mu}_z, \Sigma_z)$ specified by the latent variable z.

Using this approach, the likelihood of the model does not fundamentally change, as we still have

$$\mathcal{L}(\mathcal{D}, \Theta) = \sum_{\mathbf{x}\in\mathcal{D}} \log\left(\sum_{k=1}^{K} P(z=k)p(\mathbf{x}|z=k)\right) \tag{15.2}$$

$$= \sum_{\mathbf{x}\in\mathcal{D}} \log\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right). \tag{15.3}$$

However, by introducing the latent variable, we can use a trick to optimize the likelihood. The basic idea behind the trick is as follows: if we suppose what we know the cluster assignments z_i for each point $\mathbf{x}_i$, then the maximum likelihood problem becomes easy. In particular, if we actually observed latent z_i assignments, we would have that

$$\mathcal{L}(\mathcal{D}, \Theta) = \sum_{(\mathbf{x}_i, z_i) \in \mathcal{D}} \log\left(p(\mathbf{x}|z = z_i)P(z = z_i)\right) \tag{15.4}$$

$$= \sum_{(\mathbf{x}_i, z_i) \in \mathcal{D}} \log\left(\mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_{z_i}, \boldsymbol{\Sigma}_{z_i})\right) + \log(\pi_{z_i}). \tag{15.5}$$

The key thing is that if we know the cluster assignments, then we no longer have to sum over all the mixture components for each point. Instead, the likelihood only applies the mixture component for the cluster that each point belongs to. In terms of estimating the maximum likelihood parameters in this setting, if we differentiate and set to 0, we would simply obtain that

$$\boldsymbol{\mu}_k = \frac{\sum_{\mathbf{x}_i \in \mathcal{D}: z_i = k} \mathbf{x}_i}{|\mathbf{x}_i \in \mathcal{D} : z_i = k|} \tag{15.6}$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_{\mathbf{x}_i \in \mathcal{D}: z_i = k} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top}{|\mathbf{x}_i \in \mathcal{D} : z_i = k|} \tag{15.7}$$

$$\pi_k = \frac{|\mathbf{x}_i \in \mathcal{D} : z_i = k|}{|\mathcal{D}|}. \tag{15.8}$$

In other words, the maximum likelihood estimates for the parameters of each mixture component is just the empirical mean and empirical covariance of the points that belong to that cluster. Similarly, the maximum likelihood estimate for the π_k parameters is just the proportion of points that belong to cluster k.

Again, the crucial point is that *if we observe the latent cluster assignments*, then maximum likelihood estimation would be easy. In fact, if we know these cluster assignments, then the maximum likelihood estimation reduces to the basic maximum likelihood estimates for Gaussians that we discussed earlier in the course. However, in practice, we do not observe the cluster assignments z_i—i.e., they are assumed to be *latent variables*—and this is where the technique of *expectation maximization* comes into play.

15.2 Expectation Maximization

We saw in the previous section that we could easily maximize the likelihood of a Gaussian mixture model (GMM) *if we knew the latent cluster assignment for each point.* In practice, however, we never observe these latent values. Thus, we use the following iterative two-step approach:

1. First, we perform a (soft) assignment of each datapoint $\mathbf{x}_i \in \mathcal{D}$ to a cluster $z_i = k$. That is, we estimate the latent variable for each point, based on our current best guesses for the model components. This is known as the *expectation* step.

2. Next, we perform maximum likelihood optimization to refine the model, using the estimated cluster assignments z_i as observed values. This is known as the *maximization* step.

The idea of expectation maximization (EM) is that iterate these two steps until our estimates stabilize.

Expectation step for GMMs

We start by giving more detail on the expectation step. In this step, we perform a soft assignment of each datapoint to each component. In particular, our goal is to estimate a score $r(\mathbf{x}, k) = P(z = k|\mathbf{x})$, which tells us how likely it is that the point $\mathbf{x}$ belongs to cluster/component k. We can compute this score using Bayes rule:

$$\begin{aligned} r(\mathbf{x}, k) &= P(z = k|\mathbf{x}) && (15.9) \\ &= \frac{p(\mathbf{x}|z = k)P(z = k)}{p(\mathbf{x})} && (15.10) \\ &= \frac{p(\mathbf{x}|z = k)P(z = k)}{\sum_{j=1}^{K} p(\mathbf{x}|z = k)P(z = k)} && (15.11) \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} && (15.12) \end{aligned}$$

Thus, in the end, we get a score $r(\mathbf{x}, k)$ that tells us how much a point $\mathbf{x}$ belongs to each cluster k. Note that this step is very similar to the assignment step in soft K-means!

Maximization step for GMMs

The maximization step is a straightforward generalization of the maximum likelihood estimates that we derived at the end of Section 15.1.2. The key difference is that we need to generalize the maximum likelihood estimates to use soft assignments. Again, we can write the log-likelihood as

$$\mathcal{L}(\mathcal{D}, \Theta) = \sum_{\mathbf{x} \in \mathcal{D}} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right). \tag{15.13}$$

Now, say we differentiate this log-likelihood with respect to a particular cluster mean μ_k. Using the chain rule, we get

$$\frac{\partial \mathcal{L}(\mathcal{D}, \Theta)}{\partial \mu_k} = \sum_{\mathbf{x} \in \mathcal{D}} \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \frac{\partial}{\partial \mu_k} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{15.14}$$

As mentioned in Section 15.1.1, naively trying to finish this derivative and set to zero is intractable. However, we can note that the first term produced when applying the chain rule is equal to the responsibility score $r(\mathbf{x}, k)$ that we computed in the expectation step. Thus, we can treat this part of the equation as a constant, which transforms the expression as follows:

$$\frac{\partial \mathcal{L}(\mathcal{D}, \Theta)}{\partial \mu_k} = \sum_{\mathbf{x} \in \mathcal{D}} r(\mathbf{x}, k) \frac{\partial}{\partial \mu_k} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{15.15}$$

Thus, if we treat the $r(\mathbf{x}, k)$ terms as constants in our optimization, we simply get a re-weighted version of the standard derivative for the Gaussian likelihood, where the weights are determined by the scores $r(\mathbf{x}, k)$. The derivation of the optimal $\boldsymbol{\mu}_k$ parameter is thus just a re-weighted version of the standard estimate of the mean for a Gaussian distribution.

Similar arguments hold for the covariance parameters, and putting all this together, we can derive the following estimates:

$$\boldsymbol{\mu}_k = \frac{\sum_{\mathbf{x}_i \in \mathcal{D}} r(\mathbf{x}_i, k)\mathbf{x}_i}{\sum_{\mathbf{x}_i \in \mathcal{D}} r(\mathbf{x}_i, k)} \tag{15.16}$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_{\mathbf{x}_i \in \mathcal{D}} r(\mathbf{x}_i, k)(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top}{\sum_{\mathbf{x}_i \in \mathcal{D}} r(\mathbf{x}_i, k)} \tag{15.17}$$

$$\pi_k = \frac{\sum_{\mathbf{x}_i \in \mathcal{D}} r(\mathbf{x}_i, k)}{|\mathcal{D}|}. \tag{15.18}$$

In other words, we simply take weighted sums over the datapoints when computing the means and covariances for the Gaussian components, where the weight is determined by the score $r(\mathbf{x}_i, k)$.

It is important to note that the key point that made this maximization tractable is that we treated the

$$r(\mathbf{x}, k) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{15.19}$$

term as a constant during our optimization, and this is exactly the term that we estimated during the expectation step!

Testing for convergence

Typically, one tests for convergence by estimating how much the log-likelihood

$$\mathcal{L}(\mathcal{D}, \Theta) = \sum_{\mathbf{x} \in \mathcal{D}} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right). \tag{15.20}$$

changes after each EM step. If the change in the log-likelihood is less than a threshold ϵ, then we typically stop the algorithm.

15.2.1 Expectation maximization in general

Note that the EM approach is not only useful for Gaussian mixture models. In fact, the EM approach is a general strategy for optimizing models that contain latent variables. Even the K-means algorithm can be seen as an instantiation of EM. The general EM approach is characterized by a combination of an expectation step—where likely values for the latent variables are estimated based on the current model—followed by a maximization step—where the model parameters are maximized while treating the estimated latent values as fixed. In general, the EM algorithm—when properly run—is guaranteed to converge to a local minimum of the log-likelihood. However, it is not guaranteed to converge to a globally optimal model.