Machine Learning
the Fundamentals

William J. Deuschle

Harvard College
Cambridge, Massachusetts
April, 2019

Contents

Chapter 9

Mixture Models

The real world often generates observable data that falls into a combination of unseen categories. For example, I could record sound waves on a busy street that come from a combination of cars, pedestrians, and animals. If I were to try to model my data points, it would be helpful if I could group them by source, even though I didn't observe where each sound wave came from individually.

In this chapter we explore what are known as mixture models. Their purpose is to handle data generated by a combination of unobserved categories. We would like to discover the properties of these individual categories and determine how they mix together to produce the data we observe. We consider the statistical ideas underpinning mixture models, as well as how they can be used in practice.

9.1 Motivation

Mixture models are used to model data involving *latent variables*.

Definition 9.1.1 (Latent Variable): A latent variable is a piece of data that is not observed, but that influences the observed data. We often wish to create models that capture the behavior of our latent variables.

We are sometimes unable to observe all the data present in a given system. For example, if we measure the snout length of different animals but only get to see the snout measurements themselves, the latent variable would be the type of animal we are measuring for each data point. For most data generating processes, we will only have access to a portion of the data and the rest will be hidden from us. However, if we can find some way to also model the latent variables, our model will potentially be much richer, and we will also be able to probe it with more interesting questions. To build some intuition about latent variable models, we present a simple directed graphical model with a latent variable $\mathbf{z}_n$ in Figure 9.1.

One common means of modeling data involving latent variables, and the topic of this chapter, is known as a *mixture model*.

Definition 9.1.2 (Mixture Model): A mixture model captures the behavior of data coming from a combination of different distributions.

At a high level, a mixture model operates under the assumption that our data is generated by first sampling a discrete class, and then sampling a data point from within that category according

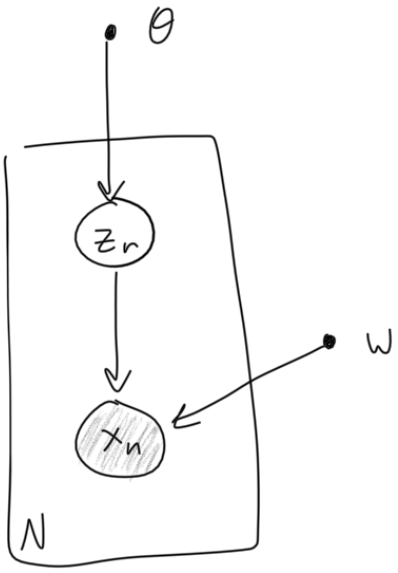

Figure 9.1: Directed graphical model with a latent variable $\mathbf{z}$.

to the distribution for that category. For the example of animal snouts, we would first sample a species of animal, and then based on the distribution of snout lengths in that species, we would sample an observation to get a complete data point.

Probabilistically, sampling a class (which is our latent variable, since we don't actually observe it) happens according to a Categorical distribution, and we typically refer to the latent variable as $\mathbf{z}$. Thus:

$$p(\mathbf{z} = C_k; \boldsymbol{\theta}) = \theta_k$$

where C_k is class k, and $\boldsymbol{\theta}$ is the parameter to the Categorical distribution that specifies the probability of drawing each class. Then, once we have a class, we have a distribution for the observed data point coming from that class:

$$p(\mathbf{x}|\mathbf{z} = C_k; \mathbf{w})$$

⋆ The distribution given by $p(\mathbf{x}|\mathbf{z} = C_k; \mathbf{w})$ is known as the *class-conditional distribution*.

This distribution depends on the type of data we are observing, and is parameterized by an arbitrary parameter $\mathbf{w}$ whose form depends on what is chosen as the class-conditional distribution. For the case of snout lengths, and many other examples, this conditional distribution will often be Gaussian, in which case our model is known as a **Gaussian Mixture Model**. We will discuss Gaussian Mixture Models in more detail later in the chapter.

If we can effectively model the distribution of our observed data points and the latent variables responsible for producing the data, we will be able to ask interesting questions of our model. For example, upon observing a new data point $\mathbf{x}'$ we will be able to produce a probability that it came from a specific class $\mathbf{z}' = C_k$ using Bayes' rule and our model parameters:

$$p(\mathbf{z}' = C_k|\mathbf{x}') = \frac{p(\mathbf{x}'|\mathbf{z}' = C_k; \mathbf{w})p(\mathbf{z}' = C_k; \boldsymbol{\theta})}{\sum_{k'} p(\mathbf{x}'|\mathbf{z}' = C_{k'}; \mathbf{w})p(\mathbf{z}' = C_{k'}; \boldsymbol{\theta})}$$

Furthermore, after modeling the generative process, we will be able to generate new data points by sampling from our categorical class distribution, and then from the class-conditional distribution for that category:

$$\mathbf{z} \sim Cat(\boldsymbol{\theta})$$
$$\mathbf{x} \sim p(\mathbf{x}|\mathbf{z} = C_k; \mathbf{w})$$

Finally, it will also be possible for us to get a sense of how many classes our data falls into, if that was not something we were aware of a priori.

ML Framework Cube: Mixture Models

While the classes of data in a mixture model will typically be discrete, the class-conditional distribution $p(\mathbf{x}|\mathbf{z} = C_k)$ can be either discrete or continuous, depending on the form of the data that we observe. Notice also that this is an unsupervised technique: while we have a data set $\mathbf{X}$ of observations, our goal is not to make predictions. Rather, we are trying to model the generative process of this data by accounting for the latent variables that generated the data points. Finally, this is a probabilistic model both for the latent variables and for our observed data.

Domain	***Training***	***Probabilistic***
Continuous/Discrete	Unsupervised	Yes

9.2 Applications

Since most of the data we observe in our world has some sort of unobserved category associated with it, there are a wide variety of applications for mixture models. Here are just a few:

1. Handwriting image recognition. The categories are given by the characters (letters, numbers, etc.) and the class-conditional is a distribution over what each of those characters might look like.

2. Modeling the topics in a document. Words or sentences in a document can be thought of as corresponding to specific topics and the class-condition for a specific topic determines a distribution over the words you might observe for that topic. More on this in the section on LDA at the end of the chapter.

3. Vehicle prices. Vehicle can be categorized a number of different ways (gas mileage, size, safety, etc.). The class-conditional for a given category determines the distribution over the price of that vehicle.

9.3 Fitting a Model

We've defined the general form of a mixture model: we have a distribution $p(\mathbf{z}; \boldsymbol{\theta})$ over our classes and a distribution $p(\mathbf{x}|\mathbf{z} = C_k; \mathbf{w})$ over our class-conditional distribution. A natural approach would be to compute the maximum likelihood values for our parameters $\boldsymbol{\theta}$ and $\mathbf{w}$. Let's consider how me might go about this for a mixture model.

9.3.1 Maximum Likelihood for Mixture Models

Our goal is to maximize the likelihood of our observed data. Because we don't actually observe the latent variables $\mathbf{z}_n$ which determine the class of each observed data point $\mathbf{x}_n$, we can simply sum over the possible classes for each of our N data points as follows:

$$p(\mathbf{X}; \boldsymbol{\theta}, \mathbf{w}) = \prod_{n=1}^{N} \sum_{k=1}^{K} p(\mathbf{x}_n, z_{n,k}; \boldsymbol{\theta}, \mathbf{w})$$

Taking the logarithm to get our log-likelihood as usual:

$$\log p(\mathbf{X}; \boldsymbol{\theta}, \mathbf{w}) = \sum_{n=1}^{N} \log \left[\sum_{k=1}^{K} p(\mathbf{x}_n, z_{n,k}; \boldsymbol{\theta}, \mathbf{w}) \right] \tag{9.1}$$

It may not be immediately obvious, but under this setup, the maximum likelihood calculation for our parameters $\boldsymbol{\theta}, \mathbf{w}$ is now intractable. The summation over the K classes of our latent variable $\mathbf{z}_n$, which is required because we don't actually observe those classes, is inside of the logarithm, which prevents us from arriving at a closed form solution (it may be helpful to try to solve this yourself, you'll realize that consolidating a summation inside of a logarithm is not possible). The rest of this chapter will deal with how we can optimize our mixture model in the face of this challenge.

9.3.2 Complete-Data Log Likelihood

We have a problem with computing the MLE for our model parameters. If we only knew which classes our data points came from, we would be able to calculate $\log p(\mathbf{x}, \mathbf{z})$ with relative ease because we would no longer require a summation inside the logarithm:

$$\log p(\mathbf{X}, \mathbf{Z}) = \sum_{n=1}^{N} \log p(\mathbf{x}_n, \mathbf{z}_n; \boldsymbol{\theta}, \mathbf{w}) \tag{9.2}$$

$$= \sum_{n=1}^{N} \log [p(\mathbf{x}_n | \mathbf{z}_n = C_k; \mathbf{w}) p(\mathbf{z}_n = C_k; \boldsymbol{\theta})] \tag{9.3}$$

$$= \sum_{n=1}^{N} \log p(\mathbf{x}_n | \mathbf{z}_n = C_k; \mathbf{w}) + \log p(\mathbf{z}_n = C_k; \boldsymbol{\theta}) \tag{9.4}$$

$$\tag{9.5}$$

Notice that because we've now observed $\mathbf{z}_n$, we don't have to marginalize over its possible values. This motivates an interesting approach that takes advantage of our ability to work with $p(\mathbf{x}, \mathbf{z})$ if we only knew $\mathbf{z}$.

The expression p($\mathbf{x}$, $\mathbf{z}$) is known as the *complete-data* because it assumes that we have both our observation $\mathbf{x}$ and the classs $\mathbf{z}$ that $\mathbf{x}$ came from. Our ability to efficiently calculate the complete-data log likelihood $\log p(\mathbf{x}, \mathbf{z})$ is the crucial piece of the algorithm we will present to optimize our mixture model parameters. This algorithm is known as **Expectation-Maximization**, or **EM** for short.

9.4 Expectation-Maximization (EM)

The motivation for the EM algorithm, as presented in the previous section, is that we do not have a closed form optimization for our mixture model parameters due to the summation inside of the

logarithm. This summation was required because we didn't observe a crucial piece of data, the class $\mathbf{z}$, and therefore we had to sum over its values.

EM uses an iterative approach to optimize our model parameters. It proposes a value for $\mathbf{z}$ using an expectation calculation, and then based on that proposed value, it maximizes our complete-data log likelihood with respect to the model parameters $\boldsymbol{\theta}$ and $\mathbf{w}$ via a standard MLE procedure.

Notice that EM is composed of two distinct steps: taking an expectation over our latent class variables and performing a maximization over our model parameters. These two steps (expectation and maximization) obviously give the algorithm its name, but more generally, this type of approach is also referred to as **coordinate ascent**. The idea behind coordinate ascent is that we can replace a hard problem (maximizing the log likelihood for our mixture model directly) with two easier problems (taking an expectation over our latent variables and maximizing our model parameters based on the complete-data, which relies on the current setting of our latent variables). We alternate between the two easier problems, executing each of them until we reach a point of convergence or decide that we've done enough.

We'll walk through the details of each of these two steps and then tie them together with the complete algorithm.

9.4.1 Expectation Step

The purpose of the expectation step (sometimes just referred to as the 'E-step') in the EM algorithm is to set the most likely values for our latent variables $\mathbf{z}_n$. In an ideal world, we would know what these classes are, and if we did, we could simply optimize our complete-data log likelihood directly as we've seen above. However, since we don't get to observe these values, one way we can get around this is just to compute the expectation of the latent variables. Let's consider what this looks like with a concrete example.

Let's say our data points $\mathbf{x}_n$ can come from one of three classes. Then, we can represent the latent variable $\mathbf{z}_n$ associated with each data point using a one-hot encoded vector. For example, if $\mathbf{z}_n$ came from class C_1, we would denote this:

$$\mathbf{z}_n = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

As we've already explained, we don't know the true value of this latent variable. Instead, we will compute its conditional expectation based on the current setting of our model parameters and our observed data $\mathbf{x}_n$. We denote the expectation of our latent variables as $\mathbf{q}_n$, and we calculate them as follows:

$$\begin{aligned} \mathbf{q}_n = \mathbb{E}[\mathbf{z}_n|\mathbf{x}_n] &= \begin{bmatrix} p(\mathbf{z}_n = C_1|\mathbf{x}_n; \boldsymbol{\theta}, \mathbf{w}) \\ p(\mathbf{z}_n = C_2|\mathbf{x}_n; \boldsymbol{\theta}, \mathbf{w}) \\ p(\mathbf{z}_n = C_3|\mathbf{x}_n; \boldsymbol{\theta}, \mathbf{w}) \end{bmatrix} \\ &\propto \begin{bmatrix} p(\mathbf{x}_n|\mathbf{z}_n = C_1; \mathbf{w})p(\mathbf{z}_n = C_1; \boldsymbol{\theta}) \\ p(\mathbf{x}_n|\mathbf{z}_n = C_2; \mathbf{w})p(\mathbf{z}_n = C_2; \boldsymbol{\theta}) \\ p(\mathbf{x}_n|\mathbf{z}_n = C_3; \mathbf{w})p(\mathbf{z}_n = C_3; \boldsymbol{\theta}) \end{bmatrix} \end{aligned}$$

Notice that we can switch from proportionality in our $\mathbf{q}_n$ values to actually probabilities by simply taking a softmax over the unnormalized values. Then, our $\mathbf{q}_n$ values will look something like the

following, where a larger number indicates a stronger belief that the data point $\mathbf{x}_n$ came from that class:

$$\mathbf{q}_n = \begin{bmatrix} 0.8 \\ 0.1 \\ 0.1 \end{bmatrix}$$

There are two important things to note about the expectation step. First, the model parameters $\boldsymbol{\theta}$ and $\mathbf{w}$ are held fixed. We're computing the expectation of our latent variables based on the current setting of those model parameters. Those parameters are randomly initialized if this is our first time running the expectation step.

Second, notice that we have a value of $\mathbf{q}_n$ for every data point $\mathbf{x}_n$ in our data set. As a result, $\mathbf{q}_n$ are sometimes called *local* variables, since there is one assigned to each data point. This is in contrast to our model parameters $\boldsymbol{\theta}$ and $\mathbf{w}$, which are considered *global* parameters. The size of the global model parameters doesn't fluctuate based on the size of our data set.

After performing the E-step, we now have an expectation for our latent variables, given by $\mathbf{q}_n$. In the maximization step, which we describe next, we use these $\mathbf{q}_n$ values to optimize our model parameters.

9.4.2 Maximization Step

After the expectation step, we have a $\mathbf{q}_n \in \mathbb{R}^K$ associated with each data point $\mathbf{x}_n$, which describes our belief that the data point came from each class C_k. Now that we have these expected 'class assignments', it's possible for us to update our model parameters $\boldsymbol{\theta}$ and $\mathbf{w}$.

Recall that optimizing our parameters using the complete-data log likelihood is tractable because we avoid summing over the classes inside of the logarithm. Although we do not have the actual complete-data (we don't know the true $\mathbf{z}_n$ values), we have an estimation of the complete-data through our $\mathbf{q}_n$ values! We use these values of $\mathbf{q}_n$ to make the optimization of our model parameters tractable.

Notice that our $\mathbf{q}_n$ values are 'soft' assignments - meaning that unlike the $\mathbf{z}_n$ values, which are one-hot encodings of assignments to a class, the $\mathbf{q}_n$ values have a probability that a data point $\mathbf{x}_n$ came from each class. Fortunately, this does not affect our maximization, which starts with out complete-data log likelihood:

$$\log p(\mathbf{X}, \mathbf{Z}) = \sum_{n=1}^{N} \log p(\mathbf{x}_n | \mathbf{z}_n = C_k; \mathbf{w}) + \log p(\mathbf{z}_n = C_k; \boldsymbol{\theta})$$

Applying the expectation with respect to $\mathbf{z}_n|\mathbf{x}_n$, which will require the $\mathbf{q}_n$ computed in the expec-

tation step:

$$\mathrm{E}_{\mathbf{z}_n|\mathbf{x}_n}[\log p(\mathbf{X}, \mathbf{Z})] = \mathrm{E}_{\mathbf{z}_n|\mathbf{x}_n}\left[\sum_{n=1}^{N} \log p(\mathbf{x}_n|\mathbf{z}_n = C_k; \mathbf{w}) + \log p(\mathbf{z}_n = C_k; \boldsymbol{\theta})\right] \tag{9.6}$$

$$= \sum_{n=1}^{N} \mathrm{E}_{\mathbf{z}_n|\mathbf{x}_n}\left[\log p(\mathbf{x}_n|\mathbf{z}_n = C_k; \mathbf{w}) + \log p(\mathbf{z}_n = C_k; \boldsymbol{\theta})\right] \tag{9.7}$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{K} p(\mathbf{z}_n = C_k|\mathbf{x}_n)\big(\log p(\mathbf{x}_n|\mathbf{z}_n = C_k; w) + \log p(\mathbf{z}_n = C_k; \boldsymbol{\theta})\big) \tag{9.8}$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{K} q_{n,k}\big(\log p(\mathbf{x}_n|\mathbf{z}_n = C_k; \mathbf{w}) + \log p(\mathbf{z}_n = C_k; \boldsymbol{\theta})\big) \tag{9.9}$$

$$\tag{9.10}$$

Notice the crucial difference between this summation and that of Equation 9.1: the summation over the classes is now outside of the logarithm! Recall that using the log-likelihood directly was intractable precisely because the summation over the classes was inside of the logarithm. This maximization became possible by taking the expectation over our latent variables (using the values we computed in the E-step), which moved the summation over the classes outside of the logarithm.

We can now complete the M-step by maximizing the tractable Equation 9.6 with respect to our model parameters $\boldsymbol{\theta}$ and $\mathbf{w}$. We simply take the derivative with respect to the parameter of interest, set to 0, solve, and update the parameter with the result.

9.4.3 Full EM Algorithm

Now that we have a grasp on the purpose of the EM algorithm, as well as an understanding of the expectation and maximization steps individually, we are ready to put everything together to describe the entire EM algorithm.

1. Begin by initializing our model parameters $\mathbf{w}$ and $\boldsymbol{\theta}$, which we can do at random. Since the EM algorithm is performed over a number of iterative steps, we will denote these initial parameter values $\mathbf{w}^{(0)}$ and $\boldsymbol{\theta}^{(0)}$. We will increment those values as the algorithm proceeds.

2. E-step: compute the values of $\mathbf{q}_n$ based on the current setting of our model parameters.

$$\mathbf{q}_n = \mathbb{E}[\mathbf{z}_n|\mathbf{x}_n] = \begin{bmatrix} p(\mathbf{z}_n = C_1|\mathbf{x}_n; \boldsymbol{\theta}^{(i)}, \mathbf{w}^{(i)}) \\ \vdots \\ p(\mathbf{z}_n = C_K|\mathbf{x}_n; \boldsymbol{\theta}^{(i)}, \mathbf{w}^{(i)}) \end{bmatrix} \propto \begin{bmatrix} p(\mathbf{x}_n|\mathbf{z}_n = C_1; \mathbf{w}^{(i)})p(\mathbf{z}_n = C_1; \boldsymbol{\theta}^{(i)}) \\ \vdots \\ p(\mathbf{x}_n|\mathbf{z}_n = C_K; \mathbf{w}^{(i)})p(\mathbf{z}_n = C_K; \boldsymbol{\theta}^{(i)}) \end{bmatrix}$$

3. M-step: compute the values of $\mathbf{w}$ and $\boldsymbol{\theta}$ that maximize our expected complete-data log likelihood for the current setting of the values of $\mathbf{q}_n$.

$$\mathbf{w}^{(i+1)} = \arg\max_{\mathbf{w}} \mathbb{E}_{\mathbf{q}_n}[\log p(\mathbf{X}, \mathbf{Z})]$$

4. Return to step 2, repeating this cycle until our likelihood converges.

9.4.4 Connection to K-Means Clustering

At this point, it's worth considering the similarity between the EM algorithm and another coordinate ascent algorithm that we considered in the context of clustering: K-Means.

Recall that K-Means proceeds according to a similar iterative algorithm: we first make hard assignments of data points to existing cluster centers, and then we update the cluster centers based on the most recent data point assignments.

In fact, the main differences between K-Means clustering and the EM algorithm are that:

1. In the EM setting, we make soft cluster assignments through our $\mathbf{q}_n$ values, rather than definitively assigning each data point to only one cluster.

2. The EM algorithm is able to take advantage of arbitrary probability distributions to capture the behavior of the observed data, whereas K-Means clustering relies only on distance measurements to make assignments and update cluster centers.

In summary, K-Means is simply a restricted form of the EM algorithm.

9.4.5 Dice Example: Mixture of Multinomials

Consider the following example scenario: we have two biased dice (with 6 faces) and one biased coin (with 2 sides). Data is generated as follows: first, the biased coin is flipped. If it lands heads, Dice 1 is rolled. If it lands tails, Dice 2 is rolled. We record the result of the dice roll, but that is our only observation. For example, our observations may look like:

$$1, 5, 3, 4, 2, 2, 3, 1, 6$$

We're going to try to infer the parameters of each of the dice based on these observation.

Let's consider how this scenario fits into our idea of a mixture model. First, the latent variable $\mathbf{z}_n$ has a natural interpretation as being which dice was rolled for the n^{th} observed data point $\mathbf{x}_n$. We can represent $\mathbf{z}_n$ using a one-hot vector, so that if the n^{th} data point came from Dice 1, we'd denote that:

$$\mathbf{z}_n = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Unsurprisingly, we denote the probability vector associated with the biased coin as $\boldsymbol{\theta} \in \mathbb{R}^2$, with θ_1 being the probability of the biased coin landing heads and θ_2 being the probability of the biased coin landing tails. Furthermore, we need parameters to describe the behavior of our biased dice. We can use $\boldsymbol{\pi_1}, \boldsymbol{\pi}_2 \in \mathbb{R}^6$, where each 6-dimensional parameter vector describes the probability that each dice lands on that face. Notice that we now have our model parameters $\mathbf{w} = \{\boldsymbol{\theta}, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2\}$. Now that we have our model parameters and an understanding of how this problem fits into the mixture model paradigm, we can optimize our model parameters using the EM algorithm. We start by initializing our parameters $\mathbf{w}^{(0)}$. Next, we need to compute our $\mathbf{q}_n$ values (the expectation step).

Recall that the formula for this is given by:

$$\mathbf{q}_n = \begin{bmatrix} p(\mathbf{z}_n = C_1|\mathbf{x}_n; \boldsymbol{\theta}^{(i)}, \mathbf{w}^{(i)}) \\ p(\mathbf{z}_n = C_2|\mathbf{x}_n; \boldsymbol{\theta}^{(i)}, \mathbf{w}^{(i)}) \end{bmatrix} \tag{9.11}$$

$$\propto \begin{bmatrix} p(\mathbf{x}_n|\mathbf{z}_n = C_1; \mathbf{w}^{(i)}) p(\mathbf{z}_n = C_1; \boldsymbol{\theta}^{(i)}) \\ p(\mathbf{x}_n|\mathbf{z}_n = C_2; \mathbf{w}^{(i)}) p(\mathbf{z}_n = C_2; \boldsymbol{\theta}^{(i)}) \end{bmatrix} \tag{9.12}$$

$$\propto \begin{bmatrix} (\pi_{11})^{x_{n,1}} (\pi_{12})^{x_{n,2}} (\pi_{13})^{x_{n,3}} (\pi_{14})^{x_{n,4}} (\pi_{15})^{x_{n,5}} (\pi_{16})^{x_{n,6}} \theta_1 \\ (\pi_{21})^{x_{n,1}} (\pi_{22})^{x_{n,2}} (\pi_{23})^{x_{n,3}} (\pi_{24})^{x_{n,4}} (\pi_{25})^{x_{n,5}} (\pi_{26})^{x_{n,6}} \theta_2 \end{bmatrix} \tag{9.13}$$

$$\tag{9.14}$$

where $x_{n,1}, ..., x_{n,6}$ denotes what was rolled for the data point $\mathbf{x}_n$. After computing the values of $\mathbf{q}_n$, we are ready to perform the maximization step for our model parameters. Recall that we are maximizing the expected complete-data log likelihood, which takes the form:

$$\mathbb{E}_{\mathbf{q}_n}[\log p(\mathbf{X}, \mathbf{Z})] = \mathbb{E}_{\mathbf{q}_n}\left[\sum_{n=1}^{N} \log p(\mathbf{z}_n) + \log p(\mathbf{x}_n|\mathbf{z}_n) \right] \tag{9.15}$$

$$= \sum_{n=1}^{N} \mathbb{E}_{\mathbf{q}_n}\left[\log p(\mathbf{z}_n) + \log p(\mathbf{x}_n|\mathbf{z}_n) \right] \tag{9.16}$$

$$= \sum_{n=1}^{N} \sum_{k=1}^{2} q_{n,k} \left(\log \theta_k + \sum_{j=1}^{6} x_{n,j} \log \pi_{k,j} \right) \tag{9.17}$$

$$\tag{9.18}$$

Note that to maximize the expected complete-data log likelihood, it's necessary to introduce Lagrange multipliers to enforce the constraints $\sum_k \theta_k = 1$ and $\sum_j \pi_{k,j} = 1$. After doing this, we recover the following update equations for our model parameters:

$$\theta_k^{(i)} \leftarrow \frac{\sum_{n=1}^{N} q_{n,k}}{N}$$

$$\pi_k^{(i)} \leftarrow \frac{\sum_{n=1}^{N} q_{n,k} \mathbf{x}_n}{\sum_{n=1}^{N} \sum_{j=1}^{6} q_{n,k} x_{n,j}}$$

We now have everything we need to perform EM for this setup. After initializing our parameters $\mathbf{w}^{(0)}$, we perform the E-step by evaluating 9.11. After calculating our values of $\mathbf{q}_n$ in the E-step, we update our parameters $\mathbf{w} = \{\boldsymbol{\theta}, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2\}$ in the M-step by maximizing 9.15 with respect to $\boldsymbol{\theta}, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2$. We perform these two steps iteratively, until convergence of our parameters.

9.5 Gaussian Mixture Models (GMM)

Our previous example was a simple but somewhat restricted application of the EM algorithm to solving a latent variable problem. We now turn to a more practical example, used widely in different contexts, called a Gaussian Mixture Model (GMM). As you might expect, a GMM consists of a combination of multiple Gaussian distributions. Among other things, it is useful for modeling scenarios where the observed data is continuous.

Let's go over a more rigorous formulation of the GMM setup. First, we have observed continous data $\mathbf{x}_n \in \mathbb{R}^m$ and latent variables $\mathbf{z}_n$ which indicate which Gaussian 'cluster' our observed data point was drawn from. In other words:

$$p(\mathbf{x}_n|\mathbf{z}_n = C_k) = \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

where $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ are the mean and covariance parameters respectively for the k^{th} cluster center.

The data generation process works as follows: we first sample a cluster center from a Categorical distribution parameterized by $\boldsymbol{\theta} \in \mathrm{R}^K$. Then, based on the sampled cluster center, we sample a data point $\mathbf{x}_n \in \mathrm{R}^m$, which is the only piece of data that we actually observe. As usual for a mixture model, it is our goal to use the observed data to determine the cluster means and covariances, as well as the parameters of the Categorical distribution that selects the cluster centers.

Fortunately, this problem setup is perfectly suited for EM. We can apply the same machinery we've discussed throughout the chapter and used in the previous example.

1. First, we randomly initialize our parameters $\boldsymbol{\theta}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$.

2. [E-Step] Calculate the posterior distribution over $\mathbf{z}_n$ given by $\mathbf{q}_n$:

$$\begin{aligned}\mathbf{q}_n = \mathrm{E}[\mathbf{z}_n|\mathbf{x}_n] &= \begin{bmatrix} p(\mathbf{z}_n = C_1|\mathbf{x}_n; \theta_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \\ \vdots \\ p(\mathbf{z}_n = C_K|\mathbf{x}_n; \theta_K, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K) \end{bmatrix} \\ &\propto \begin{bmatrix} \theta_1 \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \\ \vdots \\ \theta_K \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K) \end{bmatrix}\end{aligned}$$

   This is the current expectation for our latent variables $\mathbf{z}_n$ given our data $\mathbf{x}_n$ and the current setting of our model parameters $\boldsymbol{\theta}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$.

3. [M-Step] Using our values of $\mathbf{q}_n$, calculate the expected complete-data log likelihood, and then use that term to optimize our model parameters:

$$\begin{aligned}\mathbb{E}_{\mathbf{q}_n}[\log p(\mathbf{X}, \mathbf{Z})] &= \mathbb{E}_{\mathbf{q}_n}\left[\sum_{n=1}^N \ln(p(\mathbf{x}_n, \mathbf{z}_n; \boldsymbol{\theta}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K))\right] \\ &= \sum_{n=1}^N \sum_{k=1}^K q_{n,k} \ln \theta_k + q_{n,k} \ln \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\end{aligned}$$

   We can then use this expected complete-data log likelihood to optimize our model parameters $\boldsymbol{\theta}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ by computing the MLE as usual. Using a Lagrange multiplier to enforce

$\sum_{k=1}^{K} \theta_k = 1$, we recover the update equations:

$$\theta_k^{(i+1)} \leftarrow \frac{\sum_{n=1}^{N} q_{n,k}}{N}$$
$$\boldsymbol{\mu}_k^{(i+1)} \leftarrow \frac{\sum_{n=1}^{N} q_{n,k}\mathbf{x}_n}{\sum_{n=1}^{N} q_{n,k}}$$
$$\boldsymbol{\Sigma}_k^{(i+1)} \leftarrow \frac{\sum_{n=1}^{N} q_{n,k}(\mathbf{x}_n - \boldsymbol{\mu}_k^{(i+1)})(\mathbf{x}_n - \boldsymbol{\mu}_k^{(i+1)})^T}{\sum_{n=1}^{N} q_{n,k}}$$

4. Return to step 2. Repeat until convergence.

Finally, it's worth considering the similarity between GMMs and K-Means clustering. First, as discussed previously, GMMs rely on soft assignments rather than hard assignments. Additionally, GMMs define a normal distribution over our clusters, as opposed to a fixed point in K-Means. Intuitively, K-Means is comparable to a GMM where we only allow hard assignments and the distribution over our clusters is infinitely peaked (meaning our clusters have variance that approaches 0). In other words, K-Means is just a special case of GMMs where we've imposed more rigorous constaints than are ordinarily required by a GMM.

9.6 Admixture Models: Latent Dirichlet Allocation (LDA)

With a grasp on mixture models, it is not too difficult to understand admixture models. In a sentence: an admixture model is a mixture of mixture models. Latent Dirichlet Allocation (LDA) is a common form of admixture models, and it is sometimes also referred to as *topic modeling*, for reasons that will become apparent shortly. Describing LDA using an example will hopefully make the idea of an admixture model more concrete.

9.6.1 LDA for Topic Modeling

Consider the following data generating process for a set of text documents. We have a Dirichlet distribution $\boldsymbol{\theta} \sim Dir(\boldsymbol{\alpha})$ over the possible topics a document can take on.

⋆ If you haven't seen the Dirichlet before, it is a distribution over an n-dimensional vector whose components sum to 1. For example, a sample from a dirichlet distribution in 3-dimensions could produce a sample that is the vector

$$\begin{bmatrix} 0.2 \\ 0.5 \\ 0.3 \end{bmatrix}$$

.

We sample from that Dirichlet distribution to determine the mixture of topics $\boldsymbol{\theta}_n$ in our document D_n:

$$\boldsymbol{\theta}_n \sim Dir(\boldsymbol{\alpha})$$

Then, for each possible topic, we sample from a Dirichlet distribution to determine the mixture of words $\boldsymbol{\phi}_k$ in that topic:

$$\boldsymbol{\phi}_k \sim Dir(\boldsymbol{\beta})$$

Then, for each word $\mathbf{w}_{n,j}$ in the document D_n, we first sample from a Categorical parameterized by the topic mixture $\boldsymbol{\theta}_n$ to determine which topic that word will come from:

$$\mathbf{z}_{n,j} \sim Cat(\boldsymbol{\theta}_n)$$

Then, now that we have a topic given by $\mathbf{z}_{n,j}$ for this word $\mathbf{w}_{n,j}$, we sample from a Categorical parameterized by that topic's mixture over words given by $\boldsymbol{\phi}_{\mathbf{z}_{n,j}}$:

$$\mathbf{w}_{n,j} \sim Cat(\boldsymbol{\phi}_{\mathbf{z}_{n,j}})$$

Notice the mixture of mixtures at play here: we have a mixture model over the topics to produce each document in our corpus, and then for every word in a given document, we have a mixture over the topics to generate each individual word.

The indexing is particularly confusing because there are several layers of mixtures here, but to clarify: $n \in 1..N$ indexes each document D_n in our corpus, $k \in 1..K$ indexes each possible topic, and $j \in 1..J$ indexes each word $\mathbf{w}_{n,j}$ in document D_n, and $e \in 1..E$ indexes each word in our dictionary (note that $\mathbf{w}_{n,j} \in \mathbb{R}^E$).

$\boldsymbol{\theta}_n$ specifies the distribution over topics in document D_n, and $\boldsymbol{\alpha}$ is the hyperparameter for the distribution that produces $\boldsymbol{\theta}_n$. Similarly, $\boldsymbol{\phi}_k$ specifies the distribution over words for the k^{th} topic, and $\boldsymbol{\beta}$ is the hyperparameter for the distribution that produces $\boldsymbol{\phi}_k$.

9.6.2 Applying EM to LDA

Now that the problem setup and notation are taken care of, let's consider how we can apply EM to optimize the parameters $\boldsymbol{\theta}_n$ (the mixture over topics in a document) and $\boldsymbol{\phi}_k$ (the mixture over words for a topic). Note that we can simplify the problem slightly by considering $\boldsymbol{\theta}_n$ and $\boldsymbol{\phi}_k$ to be deterministic parameters for optimization (rather than random variables parameterized by $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$). Then, EM proceeds as follows:

1. First, we randomly initialize our parameters $\{\boldsymbol{\theta}_n\}_{n=1}^N, \{\boldsymbol{\phi}_k\}_{k=1}^K$.

2. [E-Step] Fix the topic distribution of the document given by $\boldsymbol{\theta}_n$ and the word distribution under a topic given by $\boldsymbol{\phi}_k$. Calculate the posterior distribution $\mathbf{q}_{n,j} = p(\mathbf{z}_{n,j}|\mathbf{w}_{n,j})$, and note that this is the distribution over the possible topics of a word:

$$\begin{aligned}
\mathbf{q}_{n,j} = \mathrm{E}[\mathbf{z}_{n,j}|\mathbf{w}_{n,j}] &= \begin{bmatrix} p(\mathbf{z}_{n,j} = C_1|\mathbf{w}_{n,j}; \boldsymbol{\theta}_n, \boldsymbol{\phi}_1) \\ \vdots \\ p(\mathbf{z}_{n,j} = C_K|\mathbf{w}_{n,j}; \boldsymbol{\theta}_n, \boldsymbol{\phi}_K) \end{bmatrix} \\
&\propto \begin{bmatrix} p(\mathbf{w}_{n,j}|\mathbf{z}_{n,j} = C_1; \boldsymbol{\phi}_1)p(\mathbf{z}_{n,j} = C_1; \boldsymbol{\theta}_n) \\ \vdots \\ p(\mathbf{w}_{n,j}|\mathbf{z}_{n,j} = C_K; \boldsymbol{\phi}_K)p(\mathbf{z}_{n,j} = C_K; \boldsymbol{\theta}_n) \end{bmatrix} \\
&= \begin{bmatrix} \phi_{1,\mathbf{w}_{n,j}} \cdot \theta_{n,1} \\ \vdots \\ \phi_{K,\mathbf{w}_{n,j}} \cdot \theta_{n,K} \end{bmatrix}
\end{aligned}$$

3. [M-Step] Using our values of $\mathbf{q}_n$, calculate the expected complete-data log likelihood (which marginalizes over the unknown hidden variables $\mathbf{z}_{n,j}$), and then use that expression to optimize our model parameters $\boldsymbol{\theta}_n$ and $\boldsymbol{\phi}_k$:

$$\begin{aligned}\mathbb{E}_{\mathbf{q}_n}[\log p(\mathbf{W}, \mathbf{Z})] &= \mathbb{E}_{\mathbf{q}_n}\left[\sum_{n=1}^{N}\sum_{j=1}^{J} \ln(p(\mathbf{w}_{n,j}, \mathbf{z}_{n,j}; \{\boldsymbol{\theta}_n\}_{n=1}^{N}, \{\boldsymbol{\phi}_k\}_{k=1}^{K}\right] \\ &= \sum_{n=1}^{N}\sum_{j=1}^{J}\sum_{k=1}^{K} q_{n,j,k} \ln \theta_{n,k} + q_{n,j,k} \ln \phi_{k,\mathbf{w}_{n,j}}\end{aligned}$$

We can then use this expected complete-data log likelihood to optimize our model parameters $\{\boldsymbol{\theta}_n\}_{n=1}^{N}, \{\boldsymbol{\phi}_k\}_{k=1}^{K}$ by computing the MLE as usual. Using Lagrange multipliers to enforce $\forall n \sum_{k=1}^{K} \theta_{n,k} = 1$ and $\forall k \sum_{e=1}^{E} \phi_{k,e} = 1$ (where e indexes each word in our dictionary), we recover the update equations:

$$\begin{aligned}\theta_{n,k}^{(i+1)} &\leftarrow \frac{\sum_{j=1}^{J} q_{n,j,k}}{J} \\ \phi_{k,d}^{(i+1)} &\leftarrow \frac{\sum_{n=1}^{N}\sum_{j=1}^{J} q_{n,j,k} w_{n,j,d}}{\sum_{n=1}^{N}\sum_{j=1}^{J} q_{n,j,k}}\end{aligned}$$

4. Return to step 2. Repeat until convergence.

The largest headache for applying the EM algorithm to LDA is keeping all of the indices in order, and this is the result of working with a mixture of mixtures. Once the bookkeeping is sorted out, the actual updates are straightforward.

9.7 Conclusion

Mixture models are one common way of handling data that we believe is generated through a combination of unobserved, latent variables. We've seen that training these models directly is intractable (due to the marginalization over the latent variables), and so we turned to a coordinate ascent based algorithm known as Expectation-Maximization to get around this difficulty. We then explored a couple of common mixture models, including a multinomial mixture, Gaussian Mixture Model, and an admixture model known as Latent Dirichlet Allocation. Mixture models are a subset of a broader range of models known as latent variable models, and the examples seen in this chapter are just a taste of the many different mixture models available to us. Furthermore, EM is just a single algorithm for optimizing these models. A good grasp on the fundamentals of mixture models and the EM algorithm will be useful background for expanding to more complicated, expressive latent variable models.