ELEC 548 Mixture Models and Expectation Maximization

A. Gaussian Mixtures

In general, a *mixture model* can be defined for arbitrary probability distributions. However as is almost always the case, the Gaussian distribution makes everything work nicely. Mixture models are a key foundation for machine learning based on probabilistic generative models.

We will use a *Mixture of Guassians* as a probabilistic generative model for clustering. A good reference for this topic is Chapter 9 of Bishop's *Pattern Recognition and Machine Learning*. The first step to generate a data point is to start with the cluster identity.

$$\text{Let } z \in \{1, \dots, K\} \text{ be a discrete random variable.}$$

We can define the probability of drawing a point from an individual mixture component (i.e., cluster) as

$$\Pr(z = k) = \pi_k, \text{ where } k = 1, \dots, K. \tag{1}$$

Note that for $\Pr(z)$ to be a valid probability distribution, $0 \leq \pi_k \leq 1$ and $\sum_{k=1}^{K} \pi_k = 1$.

Aside: We have begun our discussion of mixture models implicitly assuming that the number of clusters, K, is specified. Non-parametric models, for which K would not be prespefied, have been an area of active development in the last decade. A good introduction to this topic is Chapter 25 of Murphy *Machine Learning: A Probabilistic Perspective*.

Once we have our *prior probability*, $\Pr(z)$, we can define the distribution of the data point from the selected mixture component

$$\Pr(\mathbf{x} \mid z = k) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2}$$

Combining the prior probability of the cluster and the distribution of data points within the cluster we get the marginal probability of an observed data point:

$$\Pr(\mathbf{x}) = \sum_{z} \Pr(\mathbf{x} \mid z) \Pr(z) = \sum_{k=1}^{K} \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\pi_k. \tag{3}$$

A peak ahead: We will use the $\Pr(\mathbf{x})$ density with training data to do **maximum likelihood parameter estimation**. Then, with optimized parameters we can use the *a posteriori* density, $\Pr(z \mid \mathbf{x})$, to assign data points to clusters either with a ""hard decision" (choosing the class which maximizes the density) or with a "soft decision" (keeping track of how likely the datapoint is to have come from each cluster).

A.1 Maximum likelihood parameter estimation

Goal: Fit $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k$ to training data $\mathbf{x}_1, \dots, \mathbf{x}_N$.

We will use the notational shorthand $\{\mathbf{x}\}$ for the training data $\mathbf{x}_1, \ldots \mathbf{x}_N$, and the shorthand θ for all the parameters of the model $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k\}_{k=1,\ldots,K}$. Then, we can write down the likelihood of our training data

$$\begin{aligned}
\Pr(\{\mathbf{x}\} \mid \theta) &= \prod_{n=1}^{N} \Pr(\mathbf{x}_n) \\
&= \prod_{n=1}^{N} \sum_{k=1}^{K} \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\pi_k \\
\underbrace{\log \Pr(\{\mathbf{x}\} \mid \theta)}_{\text{call this } \mathcal{L}} &= \sum_{n=1}^{N} \log \left[\sum_{k=1}^{K} \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\pi_k \right].
\end{aligned}$$

(i). Find $\boldsymbol{\mu}_k$:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} = \sum_{n=1}^{N} \frac{1}{\sum_{j=1}^{K} \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \cdot \pi_j} \cdot \pi_k \cdot \left(\frac{\partial}{\partial \boldsymbol{\mu}_k} \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right) \tag{4}$$

Looking at our handy table of matrix derivatives, we find that for a symmetric $\mathbf{A}$, $\frac{\partial}{\partial \mathbf{x}} (\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x}) = 2\mathbf{A}\mathbf{x}$. Then, applying the chain rule, we have

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\mu}} \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \frac{\partial}{\partial \boldsymbol{\mu}} \left(\frac{1}{(2\pi)^{\frac{D}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T} \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})} \right) \\
&= \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \cdot \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}).
\end{aligned}$$

Plugging this into (4), we have

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} &= \sum_{n=1}^{N} \underbrace{\frac{\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \cdot \pi_k}{\sum_{j=1}^{K} \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \cdot \pi_j}}_{\text{call this } \gamma_{nk}} \cdot \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}) \\
&= \boldsymbol{\Sigma}_k^{-1} \sum_{n=1}^{N} \gamma_{nk} (\mathbf{x} - \boldsymbol{\mu}) \\
&= 0
\end{aligned}$$

Defining $N_k = \sum_{n=1}^{N} \gamma_{nk}$, we the result

$$\boxed{\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} \mathbf{x}_n} \tag{5}$$

Notice that this result is very similar to the cluster update for K-means, with γ_{nk} replacing r_{nk} as tge "responsibility" that cluster k takes in explaining the observation $\mathbf{x}_n$. Notice that $0 \leq \gamma_{nk} \leq 1$ and $\sum_{k=1}^{K} \gamma_{nk} = 1$. Let's think about this by looking at what γ_{nk} is:

$$\begin{aligned}\gamma_{nk} &= \mathsf{Pr}(z_n = k \mid \mathbf{x}_n) \\ &= \frac{\mathsf{Pr}(\mathbf{x}_n \mid z_n = k)\,\mathsf{Pr}(z_n = k)}{\mathsf{Pr}(\mathbf{x}_n)} \\ &= \frac{\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \cdot \pi_k}{\sum_{j=1}^{K} \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \cdot \pi_j}\end{aligned}$$

So, while π_k is the prior probability that $z_n = k$, γ_{nk} is the posterior probability that $z_n = k$ after we have observed $\mathbf{x}_n$.

Unlike in the case of K-means, where training data points are assigned to a particular cluster and N_k is the number of points in that cluster, for the mixture model, we can think of N_k as the "effective number" of points in cluster k.

Thus, in equation (5), $\boldsymbol{\mu}_k$ is a weighted mean of the training data, where the weights are given by the responsibilities, γ_{nk}. A small value of γ_{nk} means that cluster k bears little responsiblity for data point $\mathbf{x}_n$ and a large value means that it is very repsonsible for $\mathbf{x}_n$. (5) is very similar to the cluster update in K-means, exect that rather than hard assignments to each cluster – $r_{nk} = 0$ or 1 – each data point has a soft assignment based on γ_{nk}.

(ii). Find $\boldsymbol{\Sigma}_k$:

Looking at our handy table of matrix derivatives, we find that for a symmetric $\frac{\partial}{\partial \mathbf{X}} \mathsf{Tr}\left(\mathbf{X}^{-1}\mathbf{A}\right) = -\mathbf{X}^{-1}\mathbf{A}\mathbf{X}^{-1}$. Then, applying product and chain rules, we have

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\Sigma}} \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \frac{\partial}{\partial \boldsymbol{\Sigma}} \left(\frac{1}{(2\pi)^{\frac{D}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})} \right) \\ &= \frac{1}{(2\pi)^{\frac{D}{2}}} \left(e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})} \frac{\partial}{\partial \boldsymbol{\Sigma}} \frac{1}{|\boldsymbol{\Sigma}|^{\frac{1}{2}}} + \frac{1}{|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \frac{\partial}{\partial \boldsymbol{\Sigma}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})} \right) \\ &= \frac{1}{(2\pi)^{\frac{D}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})} \left[\left(-\frac{1}{2} \frac{|\boldsymbol{\Sigma}|\,\boldsymbol{\Sigma}^{-1}}{|\boldsymbol{\Sigma}|^{\frac{3}{2}}} \right) \right. \\ &\qquad \left. + \frac{1}{|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \left(-\frac{1}{2} \frac{\partial}{\partial \boldsymbol{\Sigma}} \mathsf{Tr}\left(\boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}} \right) \right) \right] \\ &= -\frac{1}{2} \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \left(\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} \right)\end{aligned}$$

Thus, we have

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_k} &= \sum_{n=1}^{N} \frac{1}{\sum_{j=1}^{K} \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \cdot \pi_j} \cdot \pi_k \cdot \left(\frac{\partial}{\partial \boldsymbol{\Sigma}_k} \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right) \\ &= -\frac{1}{2} \sum_{n=1}^{N} \gamma_{nk} \left(\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} \right)\end{aligned}$$

$$= 0$$

Rearranging,

$$\boldsymbol{\Sigma}^{-1} \sum_{n=1}^{N} \gamma_{nk} = \sum_{n=1}^{N} \gamma_{nk} (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^{\mathsf{T}} \boldsymbol{\Sigma}^{-1}$$

$$\boldsymbol{\Sigma}^{-1} N_k = \boldsymbol{\Sigma}^{-1} \left(\sum_{n=1}^{N} \gamma_{nk} (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^{\mathsf{T}} \right) \boldsymbol{\Sigma}^{-1}$$

Front and back multiplying by the covariance matrix, $\boldsymbol{\Sigma}$, we have our result

$$\boxed{\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} (\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})^{\mathsf{T}}} \quad . \tag{6}$$

As with the means, this is simply a weighted sample covariance of the training data.

(iii). Find π_k:

Finding a solution for π_k, the prior probilities of mixture component k is slightly more complicated, because of the constraint that the prior probabilities must sum to 1. Rather than a set-the-derivative-equal-to-zero maximization, this is a *constrained* maximization. This can be done with the Lagrange multiplier technique. Instead of maxizing $\mathcal{L}$, we maximize

$$\mathcal{L}' = \mathcal{L} + \lambda \left(\sum_{k=1}^{K} \pi_k \right),$$

where λ is the Lagrange multiplier who's value we'll discover by enforcing the constraint on the solution.

$$\begin{aligned} \frac{\partial \mathcal{L}'}{\partial \pi_k} &= \frac{\partial \mathcal{L}}{\partial \pi_k} + \lambda \\ &= \sum_{n=1}^{N} \frac{1}{\sum_{j=1}^{K} \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \cdot \pi_j} \cdot \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \lambda \\ &= \sum_{n=1}^{N} \frac{\gamma_{nk}}{\pi_k} + \lambda = 0 \end{aligned}$$

Rearranging and multiplying both sides by π_k,

$$\sum_{n=1}^{N} \gamma_{nk} = -\lambda \cdot \pi_k$$

$$\implies \pi_k = \frac{-N_k}{\lambda}$$

Now, we can enforce the constraint that $\sum_{k=1}^{K} \pi_k = 1$.

$$-\frac{\sum_{k=1}^{K} N_k}{\lambda} = 1$$

$$\implies \lambda = -N$$

Thus,

$$\boxed{\pi_k = \frac{N_k}{N}}. \tag{7}$$

In other words, the prior probability of each mixture component is the effective fraction of training data for which the component is given responsibility. If we were doing hard assignment, this would just be the number of points assigned to the component.

Wait a second! The maximum likelihood parameters given by (5), (6), and (7) are not actually in closed form! The resposibilities, γnk, appear in the right hand sides of the equations, but depend on the values of the parameters. This suggests an iterative solution – using initial guesses of the parameters to estimate the responsibilities, then recalculating the parameters. This turns out to be an instance of the **Expectation Maximization (EM) Algorithm**.

The EM algorithm is a powerful and general method for optimizing the parameters for models with **latent variables**. A latent variable is a random variable defined as part of generative model that represents something that is not actually observed. In the clustering mixture model, the $\{z_n\}$ are latent variables – unlike the $\{\mathbf{x}_n\}$, they are never directly observed.

A.2 EM Algorithm for Gaussian Mixture Model

1. Initialize parameters $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k \forall k \in 1, \ldots, K$. (Also need to decide on K!)

2. E-step: Evaluate responsibilities given current parameter values

$$\gamma_{nk} = \frac{\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

3. M-step: Re-estimate parameters using the current values of the responsibilities

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} \mathbf{x}_n$$

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

$$\pi_k^{\text{new}} = \frac{N_k}{N}$$

where $N_k = \sum_{n=1}^{N} \gamma_{nk}$.

4. Evaluate the log likelihood of training data:

$$\log \Pr(\{\mathbf{x}_n\} \mid \theta) = \sum_{n=1}^{N} \log \left[\sum_{k=1}^{K} \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \cdot \pi_k \right]$$

in order to see whether it has converged. (Alternatively, one can track the convergence of the parameters after step **3**.) If the convergence criteria are not satisfied go to step **2**.

After each iteration of the EM algorithm the log likelihood of the training data, $\log \Pr(\{\mathbf{x}_n\} \mid \theta)$, increases (meaning that the parameter estimates are improving).

Mixtures of Exponential Family Distributions: We have calculated the M-step parameter updates for clustering using a mixture of Gaussians model. The solution ends up being weighted versions of the typical maximum-likelihood parameter estimates for $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$. Here's a conjecture

Conjecture. *For a mixture model defined using an distribution from the exponential family, the parameter updates in the M-step of the EM algorithm will take the form of weighted averages of the sample estimates of these parameters.*

Here's the outline of a proof.

Proof. If we consider a distribution in the very general exponential family, we can define

$$\Pr(\mathbf{x} \mid z = k) = f(\mathbf{x} \mid \boldsymbol{\eta}_k) = h(\mathbf{x}) \exp\left(\boldsymbol{\eta}_k^{\mathsf{T}} T(\mathbf{x}) - A(\boldsymbol{\eta}_k)\right) \tag{8}$$

where $\boldsymbol{\eta}_k$ are the so-called "natural parameters" of the distribution $f(\mathbf{x})$ for component k. The typical parameters that we think of may not be the natural ones, but can be simply derived from them. The functions $h()$, $T()$, and $A()$ are determined by the type of distribution. If we have some data $\{\mathbf{x}_n\}$, for a general exponential family distribution, the maximum likelihood estimates for the natural parameters are the solution to the equation

$$\frac{\partial}{\partial \boldsymbol{\eta}} A(\boldsymbol{\eta}) = \frac{\sum_{n=1}^{N} T(\mathbf{x}_n)}{N}.$$

So now let's consider our mixture model.

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\eta}_k} = \sum_{n=1}^{N} \frac{1}{\sum_{j=1}^{K} f(\mathbf{x} \mid \boldsymbol{\eta}_j)} \cdot \pi_k \cdot \left(\frac{\partial}{\partial \boldsymbol{\eta}_k} f(\mathbf{x} \mid \boldsymbol{\eta}_k)\right) \tag{9}$$

From (8), we can see that

$$\frac{\partial}{\partial \boldsymbol{\eta}} f(\mathbf{x} \mid \boldsymbol{\eta}) = f(\mathbf{x}) \left(T(\mathbf{x}) - \frac{\partial}{\partial \boldsymbol{\eta}} A(\boldsymbol{\eta})\right)$$

Plugging this in, we have

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\eta}_k} = \sum_{n=1}^{N} \gamma_{nk} \left(T(\mathbf{x}) - \frac{\partial}{\partial \boldsymbol{\eta}_k} A(\boldsymbol{\eta}_k)\right) = 0 \tag{10}$$

$$\implies \frac{\partial}{\partial \boldsymbol{\eta}_k} A(\boldsymbol{\eta}_k) = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} T(\mathbf{x}). \tag{11}$$

Notice that this has the same form as the general maximum likelihood parameter estimate, but weighted by the responsibilities γ_{nk}. □

B. Relating EM for a Guassian Mixture Model to Classification

During the training phase, the goal for both a Gaussian Mixture Model (GMM) and a Gaussian Classification Model is to estimate the model parameters from the training data.

Key Difference: In Classification, the class labels are known, whereas in a GMM (or any Clustering problem), the class labels are not known. When we solved for the maximum likelihood parameter estimates in the Classification model, we found closed-form solutions:

$$\pi_k = \frac{N_k}{N}, \text{ where } N_k = \sum_{n \in C_k} 1 \tag{12}$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n \in C_k} \mathbf{x}_n \tag{13}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n \in C_k} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^\mathsf{T} \tag{14}$$

When we do Clustering using a GMM, we don't have labels – we don't know what class training datum $\mathbf{x}_n$ is in (i.e., whether $n \in C_k$). So we have a chicken-egg problem:

— If we knew the labels for the training data, then we could calculate ML parameters using (14).

— Conversely, if we knew the model parameters, then we could calculate $\Pr(C_k \mid \mathbf{x}_n)$ and assign $\mathbf{x}_n$ to a class, using, for example, the maximum a posteriori rule (i.e., argmax $\Pr(C_k \mid \mathbf{x}_n)$).

Using the EM algorithm, we can "bootstrap" our way out of this problem:

Summary of EM Algorithm for Clustering

1. Initialize model parameters

2. E-step: Estimate the class labels given the current estimates of the model parameters.

3. M-step: Estimate the model parameters given the current estimates of the class labels (using the distribution over classes rather than hard assignments).

4. Go to step 2: until convergence.

So to conclude the comparison, in Clustering with a GMM, the training phase uses the EM algorithm in which

— The E-step reminds us of the *test phase* in Classification.

— The M-step reminds us of the *training phase* in Classification.

In Clustering with a GMM, the test phase (cluster assignment for test or validation data) will typically involve running a single E-step on the test data (i.e., without iteration). Note: As the number of clusters is a hyper parameter for a GMM, this is exactly the sort of situation where we might split our test data into two groups, one "test" for testing different values of the number of clusters, and one "validation" for measuring the final performance (e.g., for comparison to some other algorithm).

C. The EM Algorithm in General

C.1 Motivation

What is the motivation of the EM algorithm in general? Imagine that you have a model that you'd like to fit to your training data. If the model has *latent variables* (which are ubiquitous in the case of neural signal processing), the EM algorithm provides a recipe for fitting the model. Because of this generality, the EM algorithm is among the most important and widely-used tools in machine learning. We will see it again at least two more times in this course!

C.2 Decomposition of the data likelihood

Let $\mathbf{X}$ denote all the observed variables (i.e., training data observations).
$\mathbf{Z}$ denote all the latent variables.
$\boldsymbol{\theta}$ denote all the model parameters.

Goal: Maximize $\Pr(\mathbf{X} \mid \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$.

Note: Normally, when we write down a latent variable model for our data, we describe two distributions, $\Pr(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})$ and $\Pr(\mathbf{Z})$. See, for example how we started our Gaussian Mixture model for clustering with equations (1) and (2). EM uses this model description to maximize the data likelihood, $\Pr(\mathbf{X} \mid \boldsymbol{\theta})$. In principle one could directly maximize the data likelihood without reference to the latent variables, but this quickly becomes unwieldy.

Our analysis of the EM algorithm begins by considering an arbitrary distribution over the latent variables, $q(\mathbf{Z})$.For any choice of $q(z)$, we can decomponse the data likelihood, $\Pr(\mathbf{X} \mid \boldsymbol{\theta})$, into a sum of two terms:

$$\log \Pr(\mathbf{X} \mid \boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \mathrm{KL}(q \,\|\, p_{\mathbf{Z}|\mathbf{X}}) \tag{15}$$

where

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{\Pr(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})}{q(\mathbf{Z})} \tag{16}$$

$$\mathrm{KL}(q \,\|\, p_{\mathbf{Z}|\mathbf{X}}) = -\sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{\Pr(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \tag{17}$$

— $\mathcal{L}(q, \boldsymbol{\theta})$ is a scalar that depends only on the values of the parameters, $\boldsymbol{\theta}$ and the choice of $q(\mathbf{Z})$, because $\mathbf{X}$ are observations and the latent variables, $\mathbf{Z}$, are integrated out.

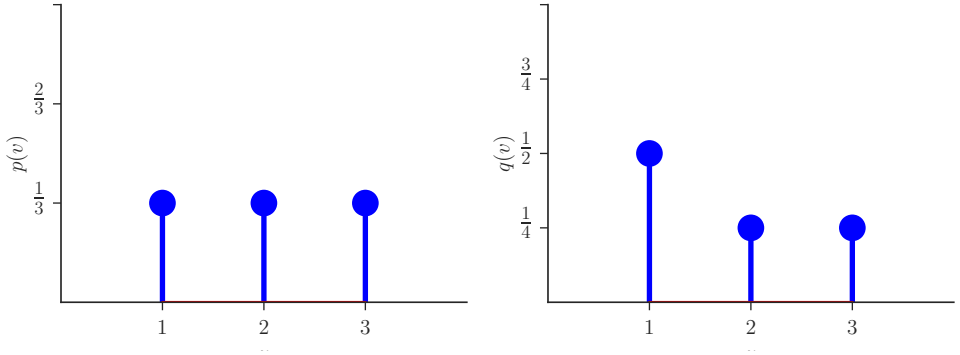
— $\mathrm{KL}(q \,\|\, p_{\mathbf{Z}|\mathbf{X}})$ is the Kullback- Leibler divergence between $q(\mathbf{Z})$ and $\Pr(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})$.

— We've written these terms as sums over the latent variables – this is shorthand for a sum or integral depending on whether the latent variables are discrete (as in the GMM) or continuous.

Aside: What is a KL divergence? The KL divergence is a scalar measure of the distance between probability distributions. In general, it's defined as

$$\mathrm{KL}(q \,\|\, p) = -\sum_{v} q(v) \log \frac{p(v)}{q(v)}$$

Let's do a simple example



$$\begin{aligned}
\mathrm{KL}(q \,\|\, p) &= -\left(q(1) \log \frac{p(1)}{q(1)} + q(2) \log \frac{p(2)}{q(2)} + q(3) \log \frac{p(3)}{q(3)} \right) \\
&= -\left(\frac{1}{2} \log \frac{\frac{1}{3}}{\frac{1}{2}} + \frac{1}{4} \log \frac{\frac{1}{3}}{\frac{1}{4}} + \frac{1}{4} \log \frac{\frac{1}{3}}{\frac{1}{4}} \right) = -\left(\frac{1}{2} \log \frac{2}{3} + \frac{1}{2} \log \frac{4}{3} \right) \\
&= -\left(\log \frac{2}{3} + \frac{1}{2} \log 2 \right) = -\left(\frac{3}{2} \log 2 - \log 3 \right) = -0.0589
\end{aligned}$$

Facts about the KL divergence

$$\mathrm{KL}(p \,\|\, p) = -\sum_v p(v) \log \frac{p(v)}{p(v)} = 0.$$

$\mathrm{KL}(q \,\|\, p) \geq 0$ for any p and q.

$\mathrm{KL}(q \,\|\, p) = 0$ if and only if $p = q$.

With the KL divergence defined, let's verify (15).

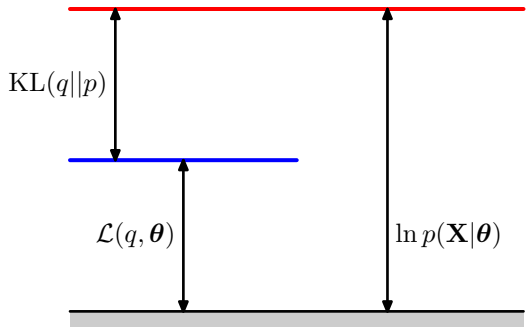
$$\begin{aligned}
\mathcal{L}(q, \boldsymbol{\theta}) &= \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{\Pr(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})}{q(\mathbf{Z})} \\
&= \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{\Pr(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}) \Pr(\mathbf{X} \mid \boldsymbol{\theta})}{q(\mathbf{Z})} \\
&= \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{\Pr(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} + \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \Pr(\mathbf{X} \mid \boldsymbol{\theta}) \\
&= -\mathrm{KL}(q \,\|\, p_{\mathbf{Z}|\mathbf{X}}) + \log \Pr(\mathbf{X} \mid \boldsymbol{\theta}) \sum_{\mathbf{Z}} q(\mathbf{Z}) \\
\implies \log \Pr(\mathbf{X} \mid \boldsymbol{\theta}) &= \mathcal{L}(q, \boldsymbol{\theta}) + \mathrm{KL}(q \,\|\, p_{\mathbf{Z}|\mathbf{X}})
\end{aligned}$$

— Notice that $\mathrm{KL}(q \,\|\, p_{\mathbf{Z}|\mathbf{X}}) \geq 0$ with equality only when our arbitrary $q(\mathbf{Z}) = \Pr(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})$.

— Thus, $\log \Pr(\mathbf{X} \mid \boldsymbol{\theta}) \geq \mathcal{L}(q, \boldsymbol{\theta})$, meaning that $\mathcal{L}(q, \boldsymbol{\theta})$ is a lower bound for $\log \Pr(\mathbf{X} \mid \boldsymbol{\theta})$.

— The bound is tight when $q(\mathbf{Z})$ is chosen to be $\Pr(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})$.

Here's the picture to have in mind (*PRML, Ch. 9, Figure 11*):



C.3 The EM algorithm

Goal: Maximize $\log \Pr(\mathbf{X} \mid \boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$.

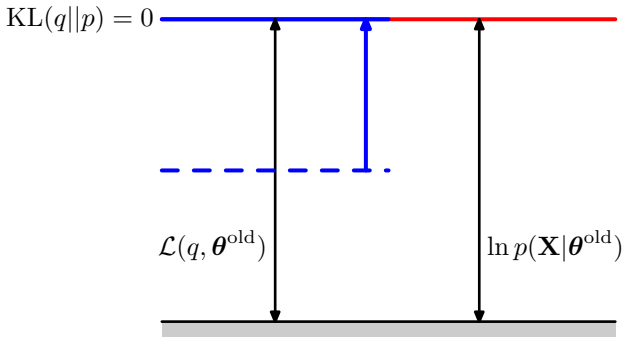
Approach: Iteratively maximize the lower bound $\mathcal{L}(q, \boldsymbol{\theta})$ w.r.t. $q()$ and $\boldsymbol{\theta}$.

E-step: Maximize $\mathcal{L}(q, \boldsymbol{\theta})$w.r.t. $q()$ while keeping $\boldsymbol{\theta}$ fixed

M-step: Maximize $\mathcal{L}(q, \boldsymbol{\theta})$w.r.t. $\boldsymbol{\theta}$ while keeping $q()$ fixed

E-step

With fixed $\boldsymbol{\theta}$, $\mathcal{L}(q, \boldsymbol{\theta})$ is maximized when $\mathrm{KL}(q \,\|\, p_{\mathbf{Z}|\mathbf{X}}) = 0$, corresponding to $q(\mathbf{Z}) = \Pr(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})$.



M-step

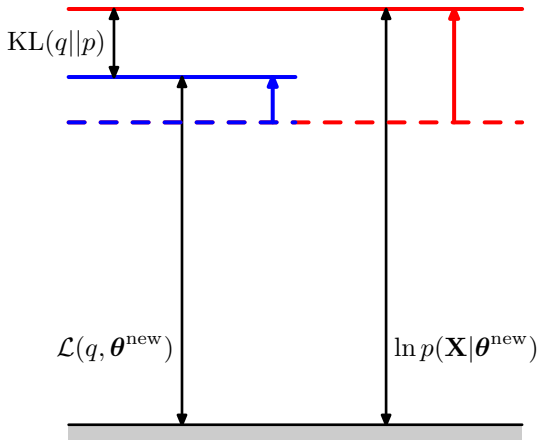
$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \Pr(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta}) - \overbrace{\sum_{\mathbf{Z}} q(\mathbf{Z}) \log q(\mathbf{Z})}^{\text{no } \boldsymbol{\theta} \text{ dependence}}$$

With fixed $q()$, maximizing $\mathcal{L}(q, \boldsymbol{\theta})$ is equivalent to maximizing

$$\sum_{\mathbf{Z}} q(\mathbf{Z}) \log \Pr(\mathbf{X}, \mathbf{X} \mid \boldsymbol{\theta}) \equiv \mathrm{E}_q[\log \Pr(\mathbf{X}, \mathbf{X} \mid \boldsymbol{\theta})].$$

We refer to $\mathrm{E}_q[\log \Pr(\mathbf{X}, \mathbf{X} \mid \boldsymbol{\theta})]$ as the "expected log joint distribution".

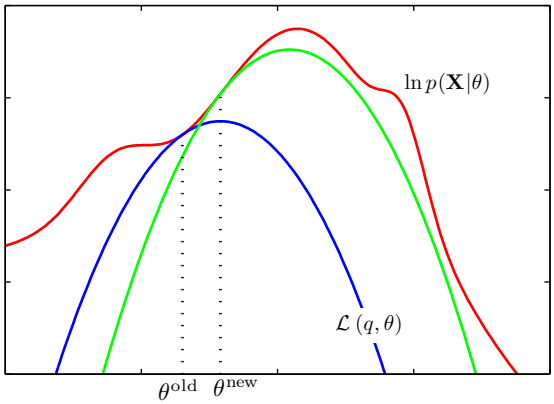
The picture now is that we increase our total data likelihood by picking the best parameters.



— Because $\mathcal{L}(q, \boldsymbol{\theta})$ is a lower bound on log Pr($\mathbf{X} \mid \boldsymbol{\theta}$), log Pr($\mathbf{X} \mid \boldsymbol{\theta}$) must increase by at least as much as $\mathcal{L}(q, \boldsymbol{\theta})$.

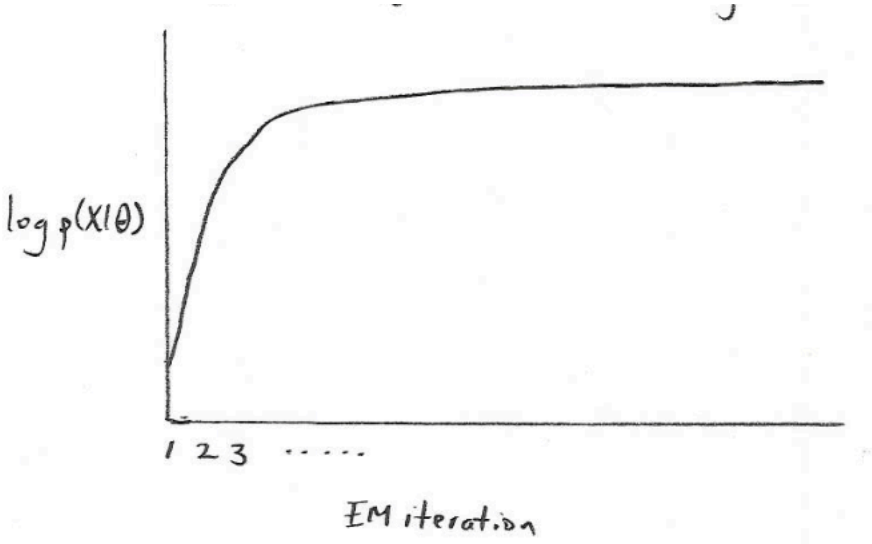
— Its "as least as much" because now there will be a nonzero $\mathrm{KL}(q \,\|\, p)$ because $p = \Pr(\mathbf{Z} \mid \mathbf{x}, \boldsymbol{\theta})$ changes if the parameters change.

Looking at the last two pictures, we see that log Pr($\mathbf{X} \mid \boldsymbol{\theta}$) is guaranteed not to decrease each EM iteration. Thus, the EM algorithm is guaranteed to converge to a local maximum. Here is a schematic of what the E and M steps of the iteration might look like as a function of a parameter:



The first E-step (blue) makes the lower bound tight, and then the first M-step finds the peak of this distribution. The next E-step (green) makes the new bound tight, etc.

When we run the EM iteration, we need to check for convergence after each step we can plot Pr($\mathbf{x} \mid \boldsymbol{\theta}$) at each time step to yield a "learning curve", which will be non-decreasing each timestep.


log p(X|θ)
1 2 3 .....
EM iteration