ECE C143A/C243A - Neural signal processing and machine learning
Prof. Jonathan Kao

Expectation maximization

Required reading for this lecture is PRML p. 423-430, 450-455. Due to time, we won't cover Gaussian mixture models in this class, but walking through the example in PRML (p. 430-450) may be helpful for gaining additional intuition about the EM algorithm.

1 General motivation

Expectation maximization (EM) is an important technique in machine learning when your model has latent (unobserved) variables. EM provides a way to learn the parameters of the model in these instances. Beacuse it is very general, it is used widely in machine learning.

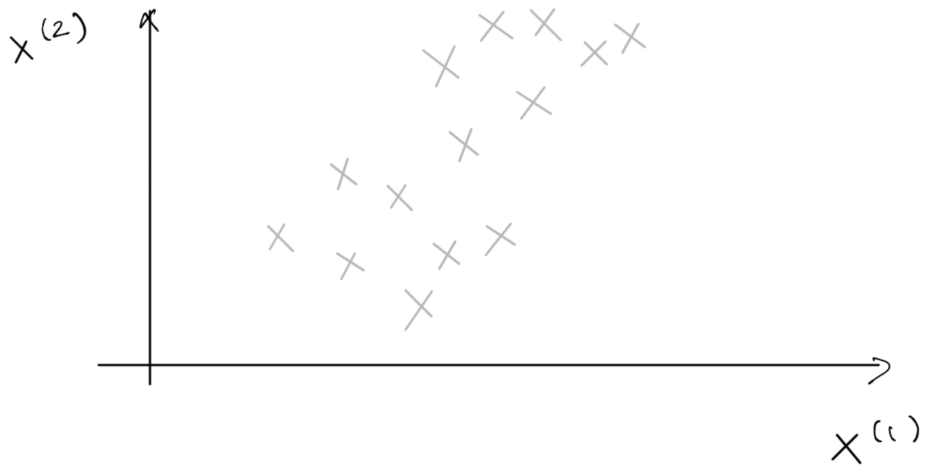
For this class, the EM algorithm will be especially useful when we consider probabilistic dimensionality reduction (where the latent variables are not observed). However, the EM algorithm has wide applicability in other scenarios. For example, it could also be used to infer the parameters of a linear dynamical system where the latent variables are not observed. It could also be used to infer the class labels for classification when the labels are unobserved. In general, if some variable in your model is unobserved (or the data is missing), EM provides a general way to still learn the parameters of your model.

The EM algorithm is composed of two steps – the "E" step and the "M" step – which are repeatedly performed. We could teach EM by presenting equations of exactly what the "E" and "M" steps do, but we'll begin with a (simple) motivating example. This example will provide the intuition behind EM; after this example we'll derive the more general EM algorithm, which we will use for probabilistic dimensionality reduction.

2 A motivating example: k-means clustering

Imagine our previous scenario of discrete classification, but instead the data now comes unlabeled. That is, we observe data, but don't know what class that data came from. Concretely, we have data $\mathbf{x}_k \in \mathbf{R}^N$ for $k = 1, \ldots, K$. We want to partition the data into C clusters (assume C is given).

As a concrete example, consider the example below (which is useful to have in-mind). How would you cluster the data if you were told that $C = 2$?



An intuitive definition of a cluster is that it is a group of data points whose inter-point distances are small compared to points outside the cluster. Let's formalize this notion.

Let $\mu_c \in \mathbf{R}^N$, where $c = 1, \ldots, C$ be the "centroid" associated with the c^{th} cluster. We define:

$$r_{kc} = \begin{cases} 1 & \text{if } \mathbf{x}_k \text{ belongs to the } c^{\text{th}} \text{ cluster} \\ 0 & \text{else} \end{cases}$$

Then an objective function quantifying our intuition is:

$$J = \sum_{k=1}^{K} \sum_{c=1}^{C} r_{kc} \left\| \mathbf{x}_k - \mu_c \right\|^2$$

Our goal would be to then find the $\{r_{kc}\}$ and μ_c such that J is minimized. One reasonable approach is to minimize J by iterating between minimizing J with respect to r_{kc} and μ_c. This is reasonable because each minimization step will decrease J. Concretely, this would look like:

- Minimize J with respect to r_{kc}, keeping μ_c fixed.

- MInimize J with respect to μ_c, keeping r_{kc} fixed.

The idea here is that we initialize some random cluster centroids. Then, we assign each point (r_{kc}) to these cluster centers such that J is minimized (i.e., each point should be assigned to the cluster closest to it). After this, we calculate new cluster centroids given the assignments r_{kc}. Then we repeat. We'll show 2 figures here from PRML to give you intuition about this.

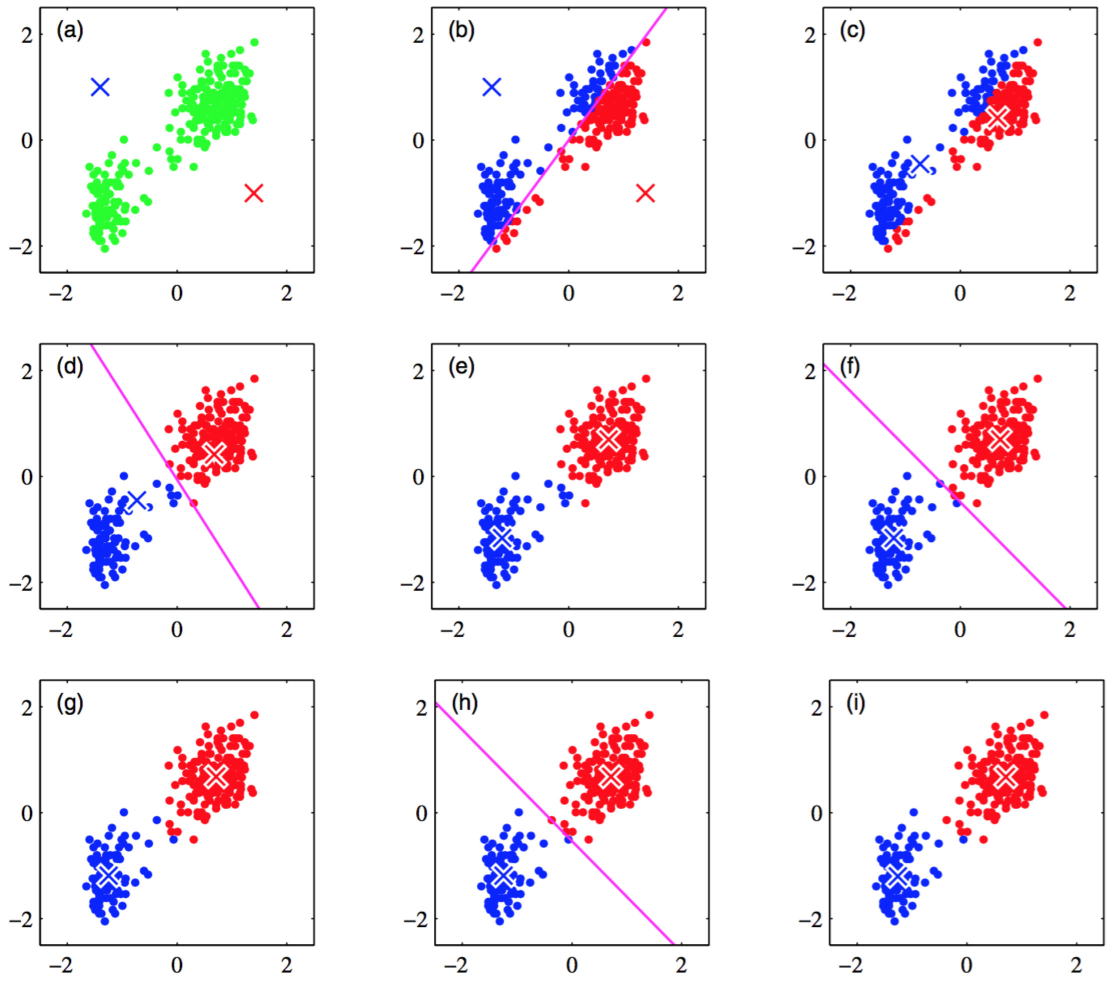


Figure 9.1 Illustration of the K-means algorithm using the re-scaled Old Faithful data set. (a) Green points denote the data set in a two-dimensional Euclidean space. The initial choices for centres $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ are shown by the red and blue crosses, respectively. (b) In the initial E step, each data point is assigned either to the red cluster or to the blue cluster, according to which cluster centre is nearer. This is equivalent to classifying the points according to which side of the perpendicular bisector of the two cluster centres, shown by the magenta line, they lie on. (c) In the subsequent M step, each cluster centre is re-computed to be the mean of the points assigned to the corresponding cluster. (d)–(i) show successive E and M steps through to final convergence of the algorithm.

Our approach in doing this was to minimize J iteratively. If we knew what the class labels r_{kc} were, we could find the cluster centroids μ_c by ML estimation (as in Discrete Classification with generative models). However, we don't know these class labels. So one way to approach the problem is to "fill-in" the values of the class labels by asking what the class labels ought to be given our μ_c (i.e., as if you were testing new data and asking what its label ought to be). After this, we can re-calculate μ_c given these new class labels. This idea is exactly the EM algorithm; we fill in the values of the latent or missing data, and then optimize our parameters as if that missing data took on these values. Then, we use these new parameters to fill in new values of the missing data, and then re-optimize our parameters to these newly-filled in values. This is the EM algorithm; the "E" step (or expectation step) corresponds to filling in the missing data (here, finding the r_{kc}) while the "M" step (or maximization step) corresponds to solving for the ML parameters with the filled-in missing data values. It's not immediately obvious that this intuitive approach ought work in practice or converge to something reasonable, but we will derive why it does.

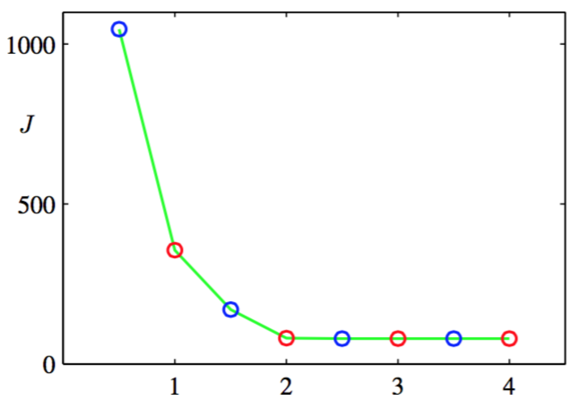


Figure 9.2 Plot of the cost function J given by (9.1) after each E step (blue points) and M step (red points) of the K-means algorithm for the example shown in Figure 9.1. The algorithm has converged after the third M step, and the final EM cycle produces no changes in either the assignments or the prototype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

2.1 E-step for k-means clustering

In assigning the labels $\{r_{k1}, \ldots, r_{kC}\}$, our constraint is that for a given data point, $\mathbf{x}_k$, this is a set of $C-1$ zeros and a single 1 (at the index of the assigned cluster).

To optimize J with respect to the r_{kc} for each k separately, we assign:

$$r_{kc} = \begin{cases} 1 & \text{if } c = \operatorname{argmin}_j \|\mathbf{x}_k - \mu_j\|^2 \\ 0 & \text{else} \end{cases}$$

In other words, we assign each data point to the closest cluster center.

2.2 M-step for k-means clustering

The M-step for k-means clustering is to optimize J with respect to μ_c. This is given by:

$$\begin{aligned}
\frac{\partial J}{\partial \mu_c} &= \frac{\partial}{\partial \mu_c} \sum_{k=1}^{K} r_{kc}(\mathbf{x}_k - \mu_c)^T(\mathbf{x}_k - \mu_c) \\
&= 2\sum_{k=1}^{K} r_{kc}(\mathbf{x}_k - \mu_c) \\
&[=] \; 0
\end{aligned}$$

which results in:

$$\mu_c = \frac{\sum_{k=1}^{K} r_{kc}\mathbf{x}_k}{\sum_{k=1}^{K} r_{kc}}$$

In other words, we assign μ_c to be equal to the mean of all data points assigned to the cluster c.

3 The EM algorithm in general

Assume we have data $\mathbf{x}_k$ that we observe, and let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K]$ denote all the observed variables. Let the variables $\mathbf{z}_k$ denote the latent variables that we don't observe, and let $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_K]$ denote all the latent variables. Let θ denote all the model parameters.

Our goal is to maximize $p(\mathbf{X}|\theta)$ with respect to θ.

Now, a common scenario is that $p(\mathbf{X}|\theta)$ may be difficult to optimize, but the quantity $p(\mathbf{X}, \mathbf{Z}|\theta)$ is straightforward to optimize (e.g., $\mathbf{Z}$ could be the discrete labels of some data; if we knew them, we could do ML estimation). Hence, it would be a good idea to write the log-likelihood in a form that has $p(\mathbf{X}, \mathbf{Z}|\theta)$.

The following decomposition of the log-likelihood is in terms of a convenient term called the Kullback-Leibler (KL) divergence. This divergence has some nice properties that make the EM algorithm intuitive. To start, we let $q(\mathbf{Z})$ be some distribution over the latent variables. For any choice of $q(\mathbf{Z})$, this decomposition $\log p(\mathbf{X}|\theta)$ is a sum of two terms:

$$\log p(\mathbf{X}|\theta) = \mathcal{L}(q, \theta) + \text{KL}(q||p)$$

where

$$
\begin{aligned}
\mathcal{L}(q,\theta) &= \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{p(\mathbf{X},\mathbf{Z}|\theta)}{q(\mathbf{Z})} && (1)\\
\mathrm{KL}(q||p) &= -\sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{p(\mathbf{Z}|\mathbf{X},\theta)}{q(\mathbf{Z})}. && (2)
\end{aligned}
$$

(On your own, you can verify that this decomposition is valid.) We call $p(\mathbf{X},\mathbf{Z}|\theta)$ the "joint distribution" and $p(\mathbf{Z}|\mathbf{X},\theta)$ the "posterior distribution." Note that $\mathcal{L}(q,\theta)$ is a scalar that depends only on $q(\cdot)$ and θ because $\mathbf{X}$ is observed and $\mathbf{Z}$ is summed out. The term $\mathrm{KL}(q||p)$ is the Kullback-Leibler divergence between $q(\mathbf{Z})$ and $p(\mathbf{Z}|\mathbf{X},\theta)$. Let's talk a bit about $\mathrm{KL}(q||p)$.

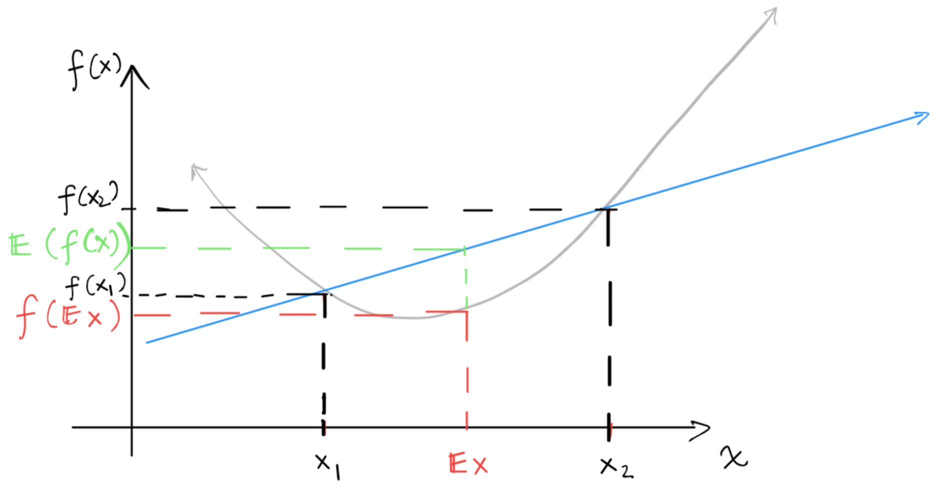
Aside: The KL divergence is a scalar that measures the "divergence" between probability distributions. Colloquially speaking, this is a measure of how different the distributions q and p are. Note however, that we should not call the KL divergence a "distance" because the KL divergence is not symmetric, i.e., $\mathrm{KL}(q||p) \neq \mathrm{KL}(p||q)$. When using KL divergence, $q(\cdot)$ is typically a "desired" distribution you would like to approximate with $p(\cdot)$. In general, KL divergence is written as:

$$
\begin{aligned}
\mathrm{KL}(q||p) &= \sum_{v} q(v) \log \frac{q(v)}{p(v)}\\
&= \sum_{v} q(v) \left(\log q(v) - \log p(v)\right).
\end{aligned}
$$

In words, I can say that the KL divergence is therefore the expected difference (under the "desired" distribution) of the log probabilities of the "desired" distribution and the approximating distribution.

Here are a few useful facts about KL divergence.

1. $\mathrm{KL}(q||p) \geq 0$ for any p and q. How do we show this? We use Jensen's inequality, which states that $\mathbb{E}f(X) \geq f(\mathbb{E}(X))$ if f is convex. A simple way to remember Jensen's inequalty is with the following diagram:



Due to Jensen's inequality:

$$\begin{aligned}\sum_v q(v)\left[-\log\frac{p(v)}{q(v)}\right] &\geq -\log\sum_v q(v)\frac{p(v)}{q(v)}\\ &= -\log\sum_v p(v)\\ &= 0\end{aligned}$$

2. $\mathrm{KL}(q||p) = 0 \iff q = p$. Showing $q = p \implies \mathrm{KL}(q||p) = 0$ is straightforward. To show the other direction, $\mathrm{KL}(q||p) = 0 \implies p = q$, note that this is the minimum of $\mathrm{KL}(q||p)$ by Jensen's inequality. Thus, we can differentiate with respect to $q(v)$ (subject to the constraint that $\sum_v q(v) = 1$) to find the $q(v)$ that achieves this minimum. We take the partial derivatives of the following expression (with respect to $q(v)$ and λ)

$$\left(\sum_v q(v)\log q(v) + \sum_v q(v)\log p(v) + \lambda\left(\sum_v q(v) - 1\right)\right)$$

and set these partial derivatives to equal 0. This gives that

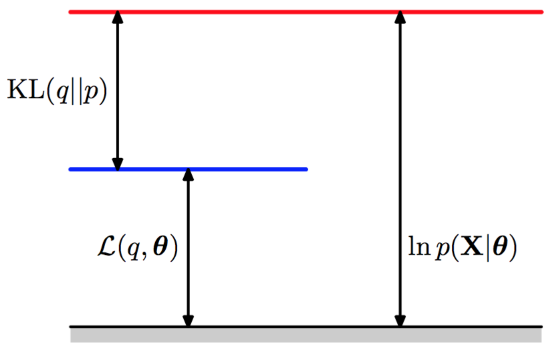
$$\begin{aligned}\log q(v) + \log p(v) + \lambda + 1 = 0 &\implies q(v) = p(v)e^{\lambda+1}\\ \left(\sum_v q(v)\right) - 1 = 0 &\implies \sum_v q(v) = 1\end{aligned}$$

Together, these terms tell us that $\mathrm{KL}(q||p) = 0$ when $q(v) = p(v)$ for all v, i.e., $q = p$.

3. In general, $\text{KL}(q||p) \neq \text{KL}(p||q)$.

In equation (2), $\text{KL}(q||p) \geq 0$, with equality if and only if $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \theta)$. Therefore, $\log p(\mathbf{X}|\theta) \geq \mathcal{L}(q, \theta)$ and $\mathcal{L}(q, \theta)$ is a lower bound for $\log p(\mathbf{X}|\theta)$. The lower bound will be tight when $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \theta)$. The picture to have in mind when we consider this decomposition is the following:

Figure 9.11 Illustration of the decomposition given by (9.70), which holds for any choice of distribution $q(\mathbf{Z})$. Because the Kullback-Leibler divergence satisfies $\text{KL}(q\|p) \geqslant 0$, we see that the quantity $\mathcal{L}(q, \boldsymbol{\theta})$ is a lower bound on the log likelihood function $\ln p(\mathbf{X}|\boldsymbol{\theta})$.


$\text{KL}(q||p)$
$\mathcal{L}(q, \boldsymbol{\theta})$
$\ln p(\mathbf{X}|\boldsymbol{\theta})$

3.1 E- and M-steps of the EM algorithm

As before, our goal is to maximize $\log p(\mathbf{X}|\theta)$ with respect to θ. To do so, we will iteratively maximize the lower bound $\mathcal{L}(q, \theta)$ with respect to q and θ. Maximizing $\mathcal{L}(q, \theta)$ is easier because it contains the term $p(\mathbf{X}, \mathbf{Z}|\theta)$. Maximizing $\mathcal{L}(q, \theta)$ with respect to q corresponds to the E-step and maximization with respect to θ corresponds to the M-step. That is:

- E-step: maximize $\mathcal{L}(q, \theta)$ with respect to q while keeping θ fixed. [Here, we are choosing the distribution $q(\cdot)$ which maximizes $\mathcal{L}(q, \theta)$. By knowing $q(\mathbf{Z})$, we can calculate the expected latent variables. This is like "filling in the values" of the latent variable; in the case of k-means, this was filling in the class labels.]

- M-step: maximize $\mathcal{L}(q, \theta)$ with respect to θ while keeping q fixed. [Here, we are choosing the optimal parameters θ assuming the expected latent variables filled in from the E-step. This is performing ML estimation with the joint data $\mathbf{X}, \mathbf{Z}$; in the case of k-means, this was re-assigning the class centroids.]

3.1.1 E-step

With fixed θ, we wish to optimize $\mathcal{L}(q, \theta)$. We have that

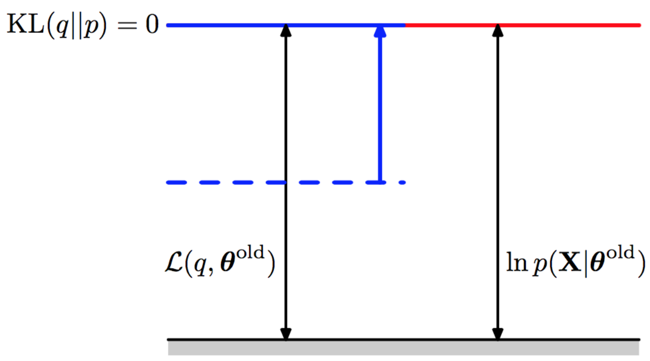
$$\mathcal{L}(q, \theta) = \log p(\mathbf{X}|\theta) - \text{KL}(q||p),$$

and hence we see that $\mathcal{L}(q, \theta)$ is maximized when $\text{KL}(q||p) = 0$. This corresponds to setting:

$$q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \theta)$$

By setting $\text{KL}(q||p) = 0$, our image of the log-likelihood looks like the following:

Figure 9.12 Illustration of the E step of the EM algorithm. The q distribution is set equal to the posterior distribution for the current parameter values $\boldsymbol{\theta}^{\text{old}}$, causing the lower bound to move up to the same value as the log likelihood function, with the KL divergence vanishing.


$\text{KL}(q||p) = 0$
$\mathcal{L}(q, \boldsymbol{\theta}^{\text{old}})$
$\ln p(\mathbf{X}|\boldsymbol{\theta}^{\text{old}})$

3.1.2 M-step

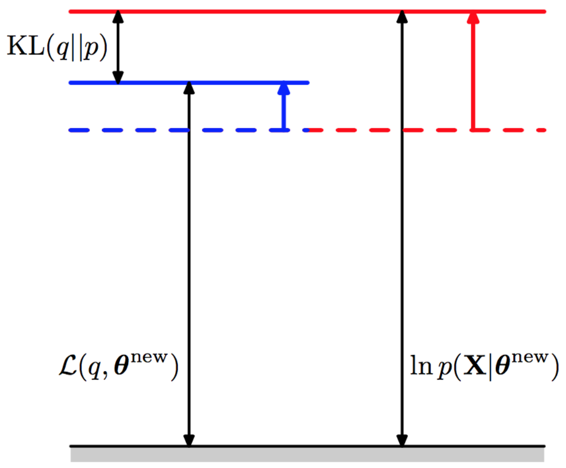
We have that

$$\mathcal{L}(q, \theta) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \log p(\mathbf{X}, \mathbf{Z}|\theta) - \sum_{\mathbf{Z}} q(\mathbf{Z}) \log q(\mathbf{Z})$$

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

$$\mathbb{E}_q \left[\log p(\mathbf{X}, \mathbf{Z}|\theta)\right] = \sum_{\mathbf{Z}} q(\mathbf{Z}) \log p(\mathbf{X}, \mathbf{Z}|\theta).$$

We would maximize this expression by calculating it, and then setting its derivative with respect to θ equal to zero (as in all previous ML estimation). Because $\mathcal{L}(q, \theta)$ is a lower bound on $\log p(\mathbf{X}|\theta)$, then $\log p(\mathbf{X}|\theta)$ must increase at least as much as $\mathcal{L}(q, \theta)$ does. Further, there is now a nonzero $\text{KL}(q||p)$ because we changed $p(\mathbf{Z}|\mathbf{X}, \theta)$.

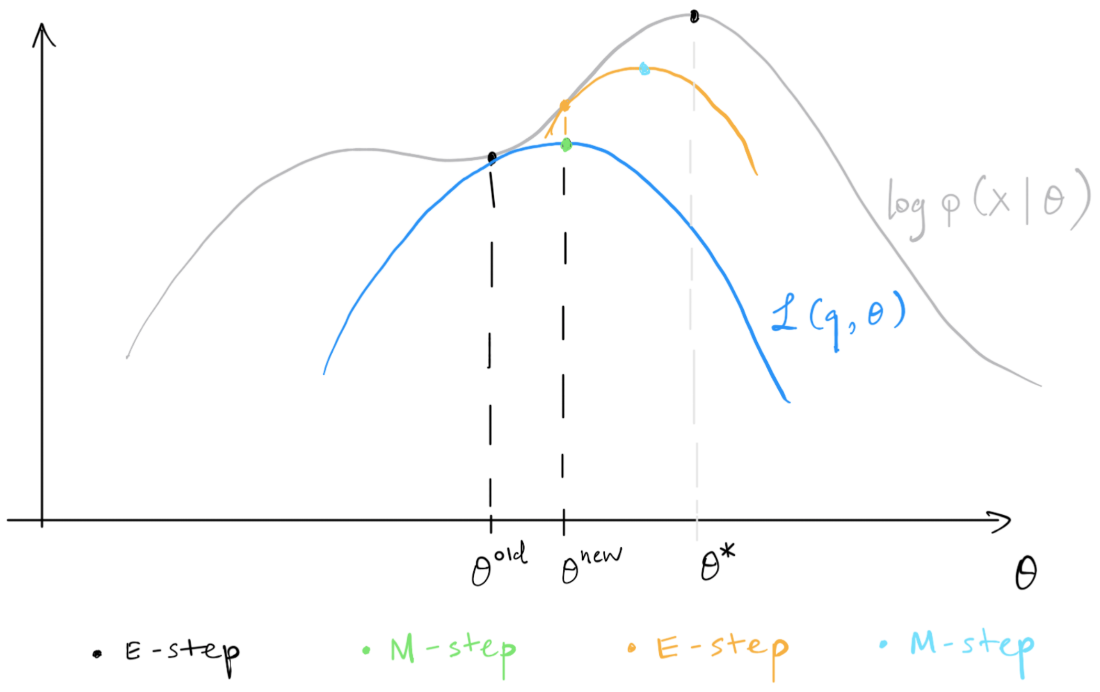


$\mathrm{KL}(q||p)$
$\mathcal{L}(q,\boldsymbol{\theta}^{\text{new}})$
$\ln p(\mathbf{X}|\boldsymbol{\theta}^{\text{new}})$

Figure 9.13 Illustration of the M step of the EM algorithm. The distribution $q(\mathbf{Z})$ is held fixed and the lower bound $\mathcal{L}(q,\boldsymbol{\theta})$ is maximized with respect to the parameter vector $\boldsymbol{\theta}$ to give a revised value $\boldsymbol{\theta}^{\text{new}}$. Because the KL divergence is nonnegative, this causes the log likelihood $\ln p(\mathbf{X}|\boldsymbol{\theta})$ to increase by at least as much as the lower bound does.

Hence, we see that $\log p(\mathbf{X}|\theta)$ is guaranteed to be non-decreasing from one EM iteration to the next. Thus, the EM algorithm is guaranteed to converge to a local optimum.

This is the picture of the EM algorithm we should have in mind:



3.2 Summary of the EM algorithm

In the EM algorithm, we:

1. Initialize parameters θ^{old}, and compute $\log p(\mathbf{X}|\theta^{\text{old}})$.
2. E-step: evaluate
$$p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}}).$$
3. M-step: find
$$\theta^{\text{new}} = \underset{\theta}{\operatorname{argmax}} \left(\sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}}) \log p(\mathbf{X}, \mathbf{Z}|\theta) \right).$$
4. Compute $\log p(\mathbf{X}, \theta^{\text{new}})$. Compare to $\log p(\mathbf{X}|\theta^{\text{old}})$.
5. If not converged, assign:
$$\theta^{\text{old}} \leftarrow \theta^{\text{new}}$$
   and go to step (2).