Mixture models and EM algorithms

October 4, 2020

Abstract

Topics: K-means clustering; Gaussian mixture models; the EM algorithm and its theory.
Readings: Chapter 9 of PRML.

1 Unsupervised learning

Unsupervised learning refers to the learning scenarios where the labels of the training examples are not given, with the goal of learning a model that describes the distribution of the given data. Clustering is one way to describe data distribution, where one groups the unlabeled examples into clusters. This is useful when one tries to reveal inherent structures of a dataset, such as customer segmentation in business intelligence, or finding similar genes in bioinformatics. In contrast, supervised learning aims to find a hypothesis that describes the relationship between the input feature vectors and output labels. As another way to describe your data, later on you will see that mixture models can describe data that can hardly be generated from a single typical distribution, such as Gaussian distributions, but easily generated from multiple distributions.

2 K-means

K-means is probably the simplest clustering algorithm. Formally, let $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ be m unlabeled training examples in $\mathbb{R}^n$ and we want to assign each $\mathbf{x}^i$ to one of K clusters $\{1, \dots, K\}$. Let the binary variable r_{ik} denote the assignment of $\mathbf{x}^{(i)}$ to the k-th cluster

$$r_{ik} = \begin{cases} 1 & \text{if } \mathbf{x}^{(i)} \text{ is assigned to cluster } k. \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

For each cluster k, there is a mean vector (or center) $\boldsymbol{\mu}_k \in \mathbb{R}^n$, which is a prototype representing the cluster (e.g., customers who are rich and old). Then K-means selects the assignment indicators and the K prototypes to minimize the following so-called "distortion" measure

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} r_{ik} \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_k\|^2. \tag{2}$$

Note that J is a function of the binary variables r_{ik}, $i = 1, \dots, m$, $k = 1, \dots, K$, and $\boldsymbol{\mu}_k$, $k = 1, \dots, K$.

First the K centers are initialized to be random vectors or certain K distinct training examples. Then K-means alternatively minimizes J with respect to the set of variables r_{ik} and the centers $\boldsymbol{\mu}_k$. In phase one, fixing $\boldsymbol{\mu}_k, k = 1, \dots, K$, for each i, the relevant terms in Eq. (2) are the sum of $\sum_{k=1}^{K} r_{ik} \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_k\|^2$. To minimize J, r_{ik^*} is assigned 1 if $\boldsymbol{\mu}_{k^*}$ is the closest centers to $\mathbf{x}^{(i)}$ and other r_{ik} are set to 0. In phase two, fixing the set of variables r_{ik}, for each k, $\boldsymbol{\mu}_k$ is updated to minimize the relevant terms $\sum_{i=1}^{m} r_{ik} \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_k\|^2$, which is a convex function in $\boldsymbol{\mu}_k$. Taking gradient of this summation with respect to $\boldsymbol{\mu}_k$ and set it to zero, we obtain

$$\sum_{i=1}^{m} r_{ik} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k) = \mathbf{0}. \tag{3}$$

Solving for $\boldsymbol{\mu}_k$ we obtain the update equation for $\boldsymbol{\mu}_k$:

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^m r_{ik}\mathbf{x}^{(i)}}{\sum_{i=1}^m r_{ik}}. \tag{4}$$

Since $r_{ik} \in \{0, 1\}$, the denominator is the number of training examples assigned to cluster k according to r_{ik}, and the numerator is the sum of all training examples that are assigned to cluster k. An intuitive interpretation is that $\boldsymbol{\mu}_k$ is updated to be the mean of examples assigned to cluster k, so the name K-means. The two phases continues until the assignments r_{ik} do not change.

Note:

- K-means will always converge since the two phases always reduce the distortion measure, which is non-negative. However, the converged solution may not be a global optimum since the objective function is not convex in both sets of variables. Typically, multiple runs of K-means can be conducted with multiple random initializations of the centers and one can select the clustering with the lowest distortion.
- Singularity where only one data point is assigned to one cluster can happen. For example, if there is an outlier that is far away from the remaining data that form a coherent cluster. In this case, the cluster with a single point will not be meaningful as it does not model the distribution of a significant portion of the data.
- Identifiability can be an issue: there is a total of $K!$ permutations of the K centers and the corresponding cluster assignment, so that these permutations has the same quality of the clusterings. There is no way to resolve the ambiguity without further knowledge of the data. In practice, each cluster needs to be assigned some meaningful label(s).

3 Gaussian Mixture Models (GMM)

3.1 Generative story of mixture models

In unsupervised learning, we are given unlabeled data $\{\mathbf{x}^{(i)}\}_{i=1}^m$ and we want to learn about how the data are generated as a description of the data. Let's assume that $\mathbf{x}^{(i)}$ are continuous vectors in $\mathbb{R}^n$ so that Gaussian distributions will be suitable to describe the generation of $\mathbf{x}^{(i)}$. However, a single Gaussian is not sufficient since the data overall may not look like Gaussian distributed. Instead, assume that the data are generated from K Gaussians

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k), k = 1, \ldots, K, \tag{5}$$

where $\boldsymbol{\mu}_k \in \mathbb{R}^n$ and $\Sigma \in \mathbb{R}^{n\times n}$ are the mean and covariance matrix of the k-th Gaussian. The K Gaussians are mixed together with a certain proportion to generate the observed training data $\{\mathbf{x}^{(i)}\}_{i=1}^m$. Let the prior probability of a data point coming the k-th Gaussian be ϕ_k. Then the probability of $\mathbf{x}^{(i)}$ is

$$\Pr(\mathbf{x}^{(i)}) = \sum_{k=1}^K \phi_k \mathcal{N}(\mathbf{x}^{(i)}|\boldsymbol{\mu}_k, \Sigma_k). \tag{6}$$

Let latent binary random variable $z_{ik} = 1$ if and only if $\mathbf{x}^{(i)}$ is sampled from the k-th Gaussian. The z_{ik} is called "latent" since they are not observed in the training data $\{\mathbf{x}^{(i)}\}_{i=1}^m$. Since each $\mathbf{x}^{(i)}$ can be from one and only one Gaussian, the random vector $\mathbf{z}_i = [z_{i1}, \ldots, z_{iK}]^\top$ is distributed according to a multinomial distribution with parameters $(\phi_1, \ldots, \phi_K)$, so that with probability ϕ_k, the k-th element of $\mathbf{z}_i$ (z_{ik}) takes value 1 and the remaining elements of $\mathbf{z}_i$ take value 0. Eq. (6) can be re-written as

$$\Pr(\mathbf{x}^{(i)}) = \sum_{\mathbf{z}_i} \Pr(\mathbf{z}_i)\Pr(\mathbf{x}^{(i)}|\mathbf{z}_i). \tag{7}$$

where

$$\Pr(\mathbf{z}_i) = \prod_{k=1}^K \phi_k^{z_{ik}}, \tag{8}$$

$$\Pr(\mathbf{x}^{(i)}|\mathbf{z}_i) = \prod_{k=1}^K \mathcal{N}(\mathbf{x}^{(i)}|\boldsymbol{\mu}_k, \Sigma_k)^{z_{ik}}. \tag{9}$$

The summation over all possible $\mathbf{z}^{(i)}$ in Eq. (7) is to marginalize out the latent variables and to obtain the probability $\Pr(\mathbf{x}^{(i)})$.

Given the parameters $\boldsymbol{\theta} = (\phi_1, \ldots, \phi_K, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \Sigma_1, \ldots, \Sigma_K)$, we can use the Bayes rule to calculate the probability of any data point $\mathbf{x}$ being generated from the k-th Gaussian by

$$\gamma_{z_k} \triangleq \Pr(z_k = 1|\mathbf{x}) = \frac{\Pr(z_k = 1, \mathbf{x})}{\Pr(\mathbf{x})} = \frac{\Pr(z_k = 1)\Pr(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} \Pr(z_j = 1)\Pr(\mathbf{x}|z_j = 1)}. \tag{10}$$

Using the Bayesian language, $\Pr(z_k = 1)$ is the prior probability that $\mathbf{x}$ is generated by the k-th Gaussian, and γ_{z_k} is the posterior probability that $\mathbf{x}$ is generated by the k-th Gaussian after seeing the data $\mathbf{x}$.

3.2 Likelihood of GMM

Assuming the training data are I.I.D. sampled from the above generative model, one can specify the probability of observing $\{\mathbf{x}^{(i)}\}_{i=1}^m$:

$$L(\boldsymbol{\theta}; \{\mathbf{x}^{(i)}\}_{i=1}^m) = \Pr(\{\mathbf{x}^{(i)}\}_{i=1}^m|\boldsymbol{\theta}) = \prod_{i=1}^{m} \Pr(\mathbf{x}^{(i)}) = \prod_{i=1}^{m} \sum_{\mathbf{z}_i} \Pr(\mathbf{z}_i)\Pr(\mathbf{x}^{(i)}|\mathbf{z}_i) \tag{11}$$

where the parameters $\boldsymbol{\theta} = (\phi_1, \ldots, \phi_K, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \Sigma_1, \ldots, \Sigma_K)$ are the parameters of the Gaussian mixture model.

Note: when discussing Gaussian discriminative analysis (GDA) for supervised learning, we need a generative model to generate the observed labels $y^{(i)}$ and feature vectors $\mathbf{x}^{(i)}$ so the likelihood of GDA is

$$L(\boldsymbol{\theta}; \{\mathbf{x}^{(i)}\}_{i=1}^m) = \prod_{i=1}^{m} \Pr(y^{(i)})\Pr(\mathbf{x}^{(i)}|y^{(i)}). \tag{12}$$

Observe that, as $y^{(i)}$ are given, there is no uncertainty in the class of $\mathbf{x}^{(i)}$ and there is no latent variables to be marginalized out. In contrast, for GMM, since we don't know $\mathbf{x}^{(i)}$ is generated by which Gaussian, we need the latent variables $\mathbf{z}$ to select a Gaussian for $\mathbf{x}^{(i)}$.

To estimate the parameters $\boldsymbol{\theta}$, it is tempting to directly maximize the log of the likelihood Eq. (11), by setting the gradient of the log-likelihood to zero. The summation in Eq. (11) will prevent to log to act directly on the probabilities so that closed-form solution is not possible (in contrast to the likelihood of GDA). We introduce EM next as an optimization method to estimate $\boldsymbol{\theta}$.

3.3 EM algorithm for GMM

The EM (Expectation-Maximization) algorithm was proposed in the 70's and has found a lot of applications in machine learning. Also, many seemingly distinct algorithms can be cast as EM algorithms. The algorithm is suitable for models with latent variable, such as GMM.

Take the log of the likelihood Eq. (11), we obtain

$$\ell(\boldsymbol{\theta}; \{\mathbf{x}^{(i)}\}_{i=1}^m) = \log L(\boldsymbol{\theta}; \{\mathbf{x}^{(i)}\}_{i=1}^m) = \sum_{i=1}^{m} \log \sum_{k=1}^{K} \left[\phi_k \Pr(\mathbf{x}^{(i)}|\boldsymbol{\mu}_k, \Sigma_k)\right]. \tag{13}$$

Here we use Eq. (6) for $\Pr(\mathbf{x}^{(i)})$. To find the parameters $\boldsymbol{\mu}_k$, set the partial derivative of ℓ with respect to $\boldsymbol{\mu}_k$ to 0, we have

$$\mathbf{0} = \sum_{i=1}^{m} \frac{\phi_k \mathcal{N}(\mathbf{x}^{(i)}|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{k=1}^{K} \phi_k \mathcal{N}(\mathbf{x}^{(i)}|\boldsymbol{\mu}_k, \Sigma_k)} \Sigma_k^{-1}(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k) \tag{14}$$

$$\doteq \sum_{i=1}^{m} \gamma(z_{ik})\Sigma_k^{-1}(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k). \tag{15}$$

where the last equality uses Eq. (10). Multiple the matrix Σ with both sides and solve for $\boldsymbol{\mu}_k$, we obtain

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{m} \gamma(z_{ik})\mathbf{x}^{(i)}}{\sum_{i=1}^{m} \gamma(z_{ik})} = \frac{\sum_{i=1}^{m} \gamma(z_{ik})\mathbf{x}^{(i)}}{N_k}. \tag{16}$$

where the denominator $N_k = \sum_{i=1}^m \gamma(z_{ik})$ is the proportion of the m training examples assigned to the k-th Gaussian according to the posterior $\gamma(z_{ik})$. This estimation of $\boldsymbol{\mu}_k$ can be interpreted as the weighted average of the training examples, with the weights on the examples being the posterior probability that the examples are generated from the Gaussians.

Solving for Σ_k is much more complicated and PRML gives the equation without proof:

$$\Sigma_k = \frac{1}{N_k} \sum_{i=1}^m \gamma(z_{ik})(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^\top. \tag{17}$$

The estimation Σ_k can also be interpreted as the weighted sum of covariance matrices $(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^\top$ estimated from each training example.

Lastly, the multinomial distribution parameters can be estimated using Lagrangian multiplier methods since there is a constraint that $\sum_{k=1}^K \phi_k = 1$. In particular, the Lagrangian function is

$$\sum_{i=1}^m \log \sum_{k=1}^K \left[\phi_k \Pr(\mathbf{x}^{(i)} | \boldsymbol{\mu}_k, \Sigma_k)\right] + \lambda \left(\sum_{k=1}^K \phi_k - 1\right). \tag{18}$$

The solution is

$$\phi_k = \frac{N_k}{m}. \tag{19}$$

See Section 9.2.2 of PRML for a detailed derivation.

There is a problem in the above derivation: we are treating the posterior $\gamma(z_{ik})$ as constants when solving for the parameters but $\gamma(z_{ik})$ is a function of the parameters (see Eq. (10)). This suggests the following coordinate maximization algorithm that modifies $\gamma(z_i k)$ and $(\boldsymbol{\mu}_k, \Sigma_k)$ alternatively. After initializing the parameters $\boldsymbol{\theta}$, the algorithm alternates between the E-step and the M-steps: in the E-step, find $\gamma(z_{ik})$ when fixing $\boldsymbol{\theta}$, and in the M-step, find $\boldsymbol{\theta}$ when fixing $\gamma(z_{ik})$. The algorithm terminates when change in the parameters or the log-likelihood between two consecutive iterations are small enough. The above optimization algorithm is called the EM (Expectation-Maximization) algorithm. See Section 9.2 of PRML for the details of the algorithm.

4 Theory for the EM algorithm

There is an unresolved question: under what condition will the EM algorithm for GMM converge? It turns out that we can formulate a more general EM algorithm as a framework to deal with models with latent variables and prove the convergence. First we need the Jensen inequality.

4.1 Jensen inequality

For a convex function f mapping from $\mathbb{R}$ to $\mathbb{R}$, we have the following inequality for any two points a and b and any constant $\lambda \in [0, 1]$:

$$f(\lambda a + (1 - \lambda)b) \leq \lambda f(a) + (1 - \lambda) f(b). \tag{20}$$

Figure 1 demonstrates this inequality. The equality holds if and only if $a = b$. We can generalize this inequality to involve more than two points. Given any $x_1, \ldots, x_n$ and any constants $\lambda_1, \ldots, \lambda_n$ so that $\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$, we have

$$f\left(\sum_{i=1}^n \lambda_i x_i\right) \leq \sum_{i=1}^n \lambda_i f(x_i). \tag{21}$$

That is, the function f evaluated at the λ-weighted average of $x_1, \ldots, x_n$ is no greater than the λ-weighted average of the values of f evaluated at $x_1, \ldots, x_n$. The equality holds if and only if x_i is a constant c. Eq. (21) is called the "Jensen inequality". If we interpret λ_i as the probability of selecting a value from $x_1, \ldots, x_n$ for the random variable x, then the Jensen inequality can be re-written using expectation:

$$f\left(\mathbb{E}_\lambda[x]\right) \leq \mathbb{E}_\lambda[f(x)]. \tag{22}$$

Even more generally, let $p(x)$ be a distribution over the domain of a convex function f, then

$$f\left(\mathbb{E}_p[x]\right) = f\left(\int_x p(x) x\right) dx \leq \int_x p(x) f(x) dx = \mathbb{E}_p[f(x)]. \tag{23}$$

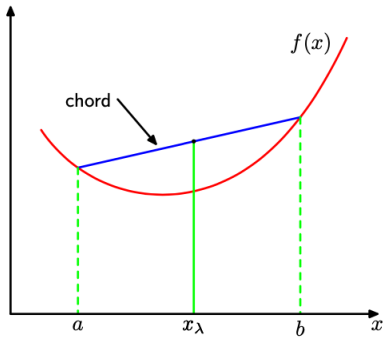

Figure 1: This figure is a copy of Figure 1.31 of PRML. The red curve shows a convex function. For any two points a and b, and a constant $0 \leq \lambda \leq 1$, the function value $f(x_\lambda) = f(\lambda a + (1-\lambda)b)$ is below the chord, which contains all the values $\lambda f(a) + (1-\lambda)f(b)$.

For a concave function f, the above inequalities will have the opposite direction. For example, Eq. (20) becomes

$$f(\lambda a + (1-\lambda)b) \geq \lambda f(a) + (1-\lambda)f(b). \tag{24}$$

For an example of using the Jensen inequality, we define the KL-divergence that computes the difference between two distributions p and q regarding the same random variable x as

$$\mathrm{KL}(p \parallel q) = \int p(x) \log \frac{p(x)}{q(x)} dx. \tag{25}$$

Since the function $f(x) = -\log x$ is convex, we have

$$\begin{aligned} \mathrm{KL}(p \parallel q) &= \int p(x) \left[-\log \frac{q(x)}{p(x)} \right] dx = \mathbb{E}_p[-\log g(x)] \\ \geq \quad & -\log \mathbb{E}_p[g(x)] = -\log \int p(x) \frac{q(x)}{p(x)} dx = -\log 1 = 0. \end{aligned}$$

The equality holds (i.e. $\mathrm{KL}(p \parallel q) = 0$) if and only if $g(x) = p(x)/q(x) = c$ for some constant c (and you can prove that $c = 1$).

4.2 The general EM algorithm

Let $\mathbf{x}$ and $\mathbf{z}$ be two sets of random variables parametrized by some parameters $\boldsymbol{\theta}$. We can for the moment think of $\mathbf{x}$ as the random variables for the data and $\mathbf{z}$ for the discrete latent variables. We call $\Pr(\mathbf{x}|\boldsymbol{\theta})$ the "incomplete-data" likelihood since $\mathbf{z}$ are missing, and $\Pr(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})$ the "complete-data" likelihood since both data and latent variables are modeled. Then the log of the incomplete-data likelihood is

$$\begin{aligned} \log \Pr(\mathbf{x}|\boldsymbol{\theta}) &= \log \sum_{\mathbf{z}} \Pr(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) = \log \sum_{\mathbf{z}} Q(\mathbf{z}) \frac{\Pr(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{Q(\mathbf{z})} \\ &\geq \sum_{\mathbf{z}} Q(\mathbf{z}) \log \frac{\Pr(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{Q(\mathbf{z})} \\ &= \mathbb{E}_Q \left[\log \frac{\Pr(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{Q(\mathbf{z})} \right] \\ &= \mathbb{E}_Q[\log \Pr(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})] - \mathbb{E}_Q \log Q(\mathbf{z}), \end{aligned} \tag{26}$$

where the inequality is due to the Jensen inequality applied to the concave logorithm function and the probability distribution $Q(\mathbf{z})$. The last term $-E_Q \log Q(\mathbf{z})$ is the entropy of the distribution Q (see Eq. (1.93) of PRML), which is a constant with respect to $\boldsymbol{\theta}$ when $Q(\mathbf{z})$ is fixed.

We see that $\log \Pr(\mathbf{x}|\boldsymbol{\theta})$ is hard to maximize with respect to $\boldsymbol{\theta}$ due to the sum after the logorithm, but the lower bound $\mathbb{E}_Q \log \frac{\Pr(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{Q(\mathbf{z})}$ is easier to maximize since the logorithm acts directly on $\Pr(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})$, which is called the "complete-data likelihood function" and can be in the exponential

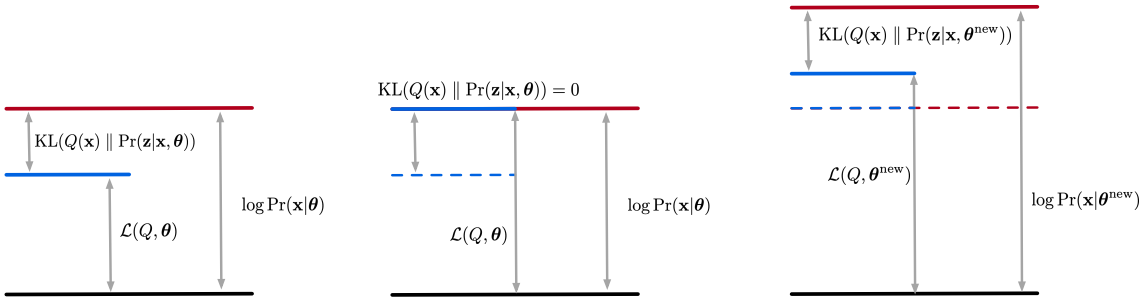

Figure 2: *Left*: Demonstration of the gap $\mathrm{KL}(Q(\mathbf{x}) \parallel \Pr(\mathbf{z}|\mathbf{x},\boldsymbol{\theta}))$ between the lower bound $\mathcal{L}(Q,\boldsymbol{\theta})$ and the incomplete-data log-likelihood $\log\Pr(\mathbf{x}|\boldsymbol{\theta})$. *Center*: The E-step chooses $Q(\mathbf{z})$ to make the gap $\mathrm{KL}(Q(\mathbf{x}) \parallel \Pr(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})) = 0$ and the lower-bound equal to $\log\Pr(\mathbf{x}|\boldsymbol{\theta})$. *Right*: The M-step updates $\boldsymbol{\theta}^{\text{old}}$ to $\boldsymbol{\theta}^{\text{new}}$ and increases the lower bound to $\mathcal{L}(Q,\boldsymbol{\theta}^{\text{new}})$. The new gap may not be zero for the $Q(\mathbf{z})$ chosen in the E-step. As a result, the new $\log\Pr(\mathbf{x}|\boldsymbol{\theta}^{\text{new}})$ may be again greater than the lower bound $\mathcal{L}(Q,\boldsymbol{\theta}^{\text{new}})$.

family. Eq. (26) gives the term "expectation" in the EM algorithm since we are taking the expectation with respect to some distribution Q. Maximizing the lower bound with respect to $\boldsymbol{\theta}$ gives the term "maximization" in the EM algorithm.

Returning to the question we raised in the beginning of this section: when will the EM algorithm converge? We show below that if the distribution Q is selected carefully, each iteration of the EM algorithm is guaranteed to increase the likelihood $\Pr(\mathbf{x}|\boldsymbol{\theta})$. Since the likelihood is upper-bounded, the EM algorithm will converge.

We analyze the gap between the lower bound in Eq. (26) and the $\log\Pr(\mathbf{x}|\boldsymbol{\theta})$. In particular,

$$\log\Pr(\mathbf{z}|\mathbf{x},\boldsymbol{\theta}) + \log\Pr(\mathbf{x}|\boldsymbol{\theta}) = \log\Pr(\mathbf{x},\mathbf{z}|\boldsymbol{\theta}), \tag{27}$$

$$\begin{aligned}
\log\Pr(\mathbf{x}|\boldsymbol{\theta}) &= \log\Pr(\mathbf{x},\mathbf{z}|\boldsymbol{\theta}) - \log\Pr(\mathbf{z}|\mathbf{x},\boldsymbol{\theta}) && (28)\\
&= \log\Pr(\mathbf{x},\mathbf{z}|\boldsymbol{\theta}) - \log Q(\mathbf{z}) + \log Q(\mathbf{z}) - \log\Pr(\mathbf{z}|\mathbf{x},\boldsymbol{\theta}) && (29)\\
&= \log\frac{\Pr(\mathbf{x},\mathbf{z}|\boldsymbol{\theta})}{Q(\mathbf{z})} - \log\frac{\Pr(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})}{Q(\mathbf{z})}. && (30)
\end{aligned}$$

By taking expectations on both sides with respect to the distribution $Q(\mathbf{z})$, we obtain

$$\begin{aligned}
\log\Pr(\mathbf{x}|\boldsymbol{\theta}) &= \mathbb{E}_Q\left[\log\frac{\Pr(\mathbf{x},\mathbf{z}|\boldsymbol{\theta})}{Q(\mathbf{z})}\right] - \mathbb{E}_Q\left[\log\frac{\Pr(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})}{Q(\mathbf{z})}\right]\\
&= \mathcal{L}(Q,\boldsymbol{\theta}) + \mathrm{KL}(Q(\mathbf{x}) \parallel \Pr(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})). && (31)
\end{aligned}$$

Note the following

- $\mathbb{E}_Q\left[\log\Pr(\mathbf{x}|\boldsymbol{\theta})\right] = \log\Pr(\mathbf{x}|\boldsymbol{\theta})$ since $\log\Pr(\mathbf{x}|\boldsymbol{\theta})$ is a constant with respect to $\mathbf{z}$.
- The first term on the right hand side is the lower bound in Eq. (26) so that the gap between the hard-to-maximize incomplete-data log-likelihood and the easier-to-maximize expected complete-data log-likelihood under Q is the KL-divergence between $Q(\mathbf{z})$ and $\Pr(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})$. Since KL-divergence is non-negative, so that we again prove that the expectation $\mathcal{L}(Q,\boldsymbol{\theta})$ is a lower bound of the incomplete-data log-likelihood $\log\Pr(\mathbf{x}|\boldsymbol{\theta})$. Eq. (31) is demonstrated in the left panel of Figure 2.

To make the gap as small as possible, we use the property that, the KL-divergence is zero if and only if the two compared distributions $Q(\mathbf{z})$ and $\Pr(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})$ are identical, that is,

$$Q(\mathbf{z}) = \Pr(\mathbf{z}|\mathbf{x},\boldsymbol{\theta}). \tag{32}$$

Under this selection of $Q(\mathbf{z})$, maximizing $\mathbb{E}_Q[\log\Pr(\mathbf{x},\mathbf{z}|\boldsymbol{\theta})]$ then guarantees that $\log\Pr(\mathbf{x}|\boldsymbol{\theta})$ is maximized and EM algorithm will converge. See the center and right panels of Figure 2. Figure 3 is a more commonly seen figure for EM algorithm.

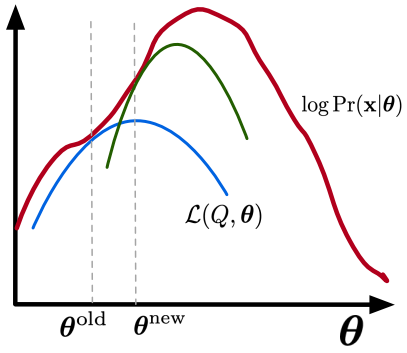

Figure 3: The incomplete-data log-likelihood (the red curve) may not be concave and is hard to maximize. The EM algorithm climbs the hill of the log-likelihood. E-step: given the current $\boldsymbol{\theta}^{\text{old}}$ choosing a $Q(\mathbf{z})$ distribution and constructing a lower bound (the blue curve) of the log-likelihood. The lower bound is a function of $\boldsymbol{\theta}$ and is a tight lower bound of the likelihood function at $\boldsymbol{\theta}^{\text{old}}$, in the sense that the bound touches the likelihood function. The lower bound function is assumed easier to maximize with respect to $\boldsymbol{\theta}$. M-step: find a new $\boldsymbol{\theta}^{\text{new}}$ that maximizes the lower bound. At $\boldsymbol{\theta}^{\text{new}}$, the previous lower bound may not touch the log-likelihood and a new lower bound needs to be constructed (the green curve).

4.3 Mixture of Bernoulli distributions

As another example of using EM to optimize models with latent variables, let's modify GMM to replace the Gaussian data generative model $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_k, \Sigma_k)$ with Bernoulli distributions $\mathbf{x} \sim \text{Bernouli}(\boldsymbol{\mu}_k)$. We are thus assuming that the observed data have binary features rather than continuous features: $\mathbf{x} \in \{0,1\}^n$. This model sometimes is used for text data clustering: each document is represented as a set of binary features, each of which indicates if a word appears in the document, and the class or topics of the documents are unknown and need to be inferred.

Similar to GMM, the mixture parameters ϕ_k, $k = 1, \ldots, K$ represent the probability of selecting one of the K Bernoulli distributions for data generation. Let the binary random vector $\mathbf{z} = [z_1, \ldots, z_K]$ be a one-hot vector ($\sum_{k=1}^K z_k = 1$) and distributed as

$$\Pr(\mathbf{z}|\boldsymbol{\phi}) = \prod_{k=1}^K \phi_k^{z_k} \tag{33}$$

The k-th Bernoulli distribution is parametrized by the mean vector $\boldsymbol{\mu_k} = [\mu_{k1}, \ldots, \mu_{kn}]$, $0 \le \mu_{kj} \le 1$ and the data are generated from the k-th Bernoulli by

$$\Pr(\mathbf{x}|\boldsymbol{\mu}_k) = \prod_{j=1}^n \mu_{kj}^{x_j}(1-\mu_{kj})^{1-x_j}. \tag{34}$$

Let all the parameters be collectively denoted by $\boldsymbol{\theta}$. Then the incomplete-data likelihood is

$$\Pr(\mathbf{x}|\boldsymbol{\theta}) = \sum_{k=1}^K \phi_k \Pr(\mathbf{x}|\boldsymbol{\mu}_k). \tag{35}$$

Given a training set of m I.I.D. examples $\{\mathbf{x}^{(i)}\}_{i=1}^m$, where each $\mathbf{x}^{(i)} \in \{0,1\}^n$, the incomplete-data likelihood is

$$\log \Pr(\{\mathbf{x}^{(i)}\}_{i=1}^m|\boldsymbol{\theta}) = \sum_{i=1}^m \log \sum_{k=1}^K \phi_k \Pr(\mathbf{x}^{(i)}|\boldsymbol{\mu}_k). \tag{36}$$

Again, we see that the log does not act on the mixture parameter and the Bernoulli distributions.

Below is the recipe of using EM to estimate parameters $\boldsymbol{\theta}$ for the mixture of Bernoulli distributions.

1. Let $Z = [\mathbf{z}_1, \ldots, \mathbf{z}_m]$ be the latent variables with $\mathbf{z}_i$ for $\mathbf{x}^{(i)}$. Form the complete-data log-likelihood

$$\log \Pr(\{\mathbf{x}^{(i)}\}_{i=1}^m, Z|\boldsymbol{\theta}) = \sum_{i=1}^m \sum_{k=1}^K z_{ik} \left\{ \log \phi_k + \sum_{j=1}^n [x_j^{(i)} \log \mu_{kj} + (1 - x_j^{(i)}) \log(1-\mu_{kj})] \right\}$$

2. For each $\mathbf{x}^{(i)}$, derive Q_i using the current estimation of $\boldsymbol{\theta}$:

$$Q_i(\mathbf{z}_i) = \Pr(\mathbf{z}_i|\mathbf{x}^{(i)}, \boldsymbol{\theta}) = \frac{\prod_{k=1}^{K}[\phi_k \Pr(\mathbf{x}^{(i)}|\boldsymbol{\mu}_k)]^{z_{ik}}}{\sum_{\mathbf{z}} \prod_{k=1}^{K}[\phi_k \Pr(\mathbf{x}^{(i)}|\boldsymbol{\mu}_k)]^{z_k}} \tag{37}$$

The numerator is the prior ϕ_k multiplied by the likelihood $\Pr(\mathbf{x}^{(i)}|\boldsymbol{\mu}_k)$ for the k where $z_{ik} = 1$. The denominator is the summation of the prior-likelihood products over all possible $\mathbf{z}$ to marginalize out $\mathbf{z}$.

3. Compute the expectation of the complete-data likelihood under the distribution $Q_i(\mathbf{z}_i) = \Pr(\mathbf{z}_i|\mathbf{x}^{(i)}, \boldsymbol{\theta})$, the posterior distribution of $\mathbf{z}_i$ after seeing the data:

$$\begin{aligned}
& \mathbb{E}_Q[\log \Pr(\{\mathbf{x}^{(i)}\}_{i=1}^m, Z|\boldsymbol{\theta})] \\
= & \sum_{i=1}^{m}\sum_{k=1}^{K} \mathbb{E}_{Q_i}[z_{ik}] \left\{ \log \phi_k + \sum_{j=1}^{n} [x_j^{(i)} \log \mu_{kj} + (1 - x_j^{(i)}) \log(1 - \mu_{kj})] \right\} \\
= & \sum_{i=1}^{m}\sum_{k=1}^{K} \gamma(z_{ik}) \left\{ \log \phi_k + \sum_{j=1}^{n} [x_j^{(i)} \log \mu_{kj} + (1 - x_j^{(i)}) \log(1 - \mu_{kj})] \right\},
\end{aligned}$$

where

$$\gamma(z_{ik}) = \mathbb{E}_{Q_i}[z_{ik}] = \sum_{\mathbf{z}_i} z_{ik} \Pr(\mathbf{z}_i|\mathbf{x}^{(i)}, \boldsymbol{\theta}) = \frac{\phi_k \Pr(\mathbf{x}^{(i)}|\boldsymbol{\mu}_k)}{\sum_{j=1}^{K} \phi_j \Pr(\mathbf{x}^{(i)}|\boldsymbol{\mu}_j)}.$$

4. Maximize $\mathbb{E}_Q[\log \Pr(\{\mathbf{x}^{(i)}\}_{i=1}^m, Z|\boldsymbol{\theta})]$ with respect to $\boldsymbol{\theta}$. By setting the partial derivative of $\mathbb{E}_Q[\log \Pr(\{\mathbf{x}^{(i)}\}_{i=1}^m, Z|\boldsymbol{\theta})]$ with respect to $\boldsymbol{\mu}_k$, we obtain

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{m} \mathbf{x}^{(i)}}{N_k}, \tag{38}$$

where $N_k = \sum_{i=1}^{m} \gamma(z_{nk})$. Using the Lagrangian multipler method to take care of the constraint that $\sum_{k=1}^{K} \phi_k = 1$, we obtain

$$\phi_k = \frac{N_k}{N}. \tag{39}$$

5. If the new estimated $\boldsymbol{\theta}$ do not differ much from the previous $\boldsymbol{\theta}$ estimation, then exit. Otherwise, go back to step 2 with the new estimation of $\boldsymbol{\theta}$.