Lecture 26: Expectation Maximization(EM algorithm)

April 12, 2016

AIM: Suppose we get only partial observations/samples from a parameterized population, then how can we perform efficient maximum likelihood parameter estimation?

Applications:

1. Machine Learning
2. Clustering (Unsupervised learning)
3. Bio-informatics, Genomics, Speech processing (Baum-Welch algorithm)

1 Estimating Mixtures of Gaussians (MoG)

The MoG model is a joint distribution on $(\boldsymbol{x}, z)$ with $\boldsymbol{x} \in \mathbb{R}^d, z \in [k]$ and z has multinomial distribution,

$$z \sim \text{Multinomial Distribution}(\boldsymbol{\phi})$$

i.e., Multinomial$\left[[\phi_1, \phi_2, ...\phi_k]^T\right]$ with $\phi_i \geq 0$; $\sum_{j=1}^k \phi_j = 1$. Given $z = j$, the random vector $\boldsymbol{x}$ is Gaussian distributed $\boldsymbol{x}|(z = j) \sim \mathcal{N}(\boldsymbol{\mu}_j, \Sigma_j)$. Here, $\boldsymbol{\phi}$ is the mixture distribution, $\{\boldsymbol{\mu}_j\}$ is the cluster center and $\{\Sigma_j\}$ is the cluster size.

Example 1.1. For $d = k = 2$, let

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \ ; \ \boldsymbol{\mu}_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix},$$

$$\Sigma_1 = \Sigma_2 = I_2, \text{ and } \boldsymbol{\phi} = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}.$$

Here, cluster concentration is uniform as seen in Fig. 1, and roughly centers of clusters are $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$.

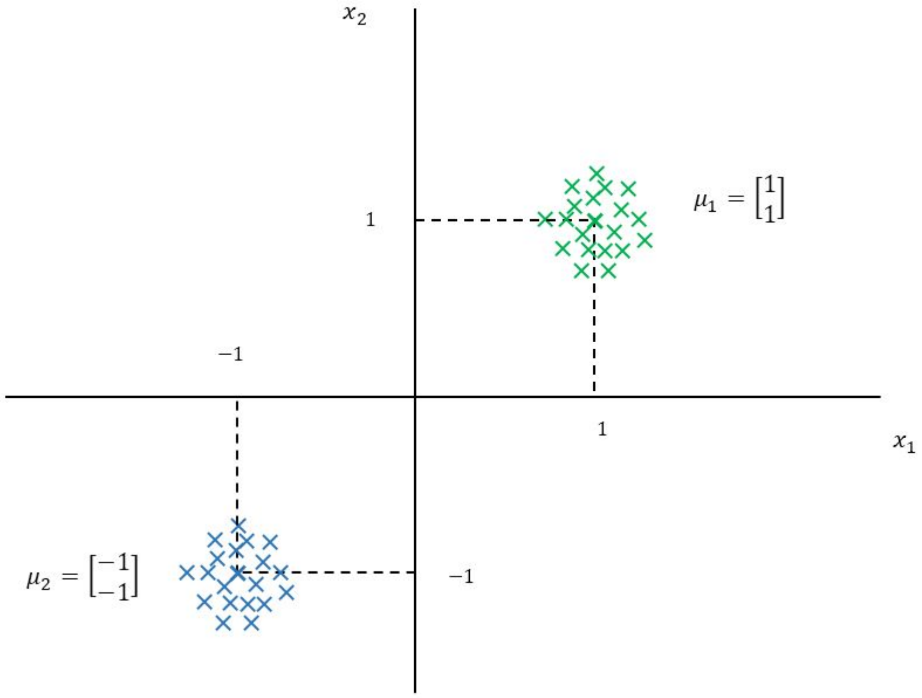

Figure 1: Example 1

Example 1.2. For $d = k = 2$, let

$$\boldsymbol{\mu}_1 = \begin{bmatrix}1\\1\end{bmatrix} \;\; ; \; \boldsymbol{\mu}_2 = \begin{bmatrix}-1\\-1\end{bmatrix},$$

$$\Sigma_1 = \Sigma_2 = I_2, \text{ and } \boldsymbol{\phi} = \begin{bmatrix}0.25 & 0.75\end{bmatrix}.$$

Since the distribution is non-uniform, cluster density is also different (see Fig. 2).

Let us define parameter

$$\theta \equiv (\boldsymbol{\phi}, \underbrace{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_k}_{\boldsymbol{\mu}}, \underbrace{\Sigma_1, \Sigma_2, ..., \Sigma_k}_{\boldsymbol{\Sigma}}). \tag{1}$$

Suppose we only observe $\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_m} \in \mathbb{R}^d$ where $(\boldsymbol{x_i}, z_i) \overset{iid}{\sim}$ mixture of Gaussians with parameter θ (here, z_i is called "latent variable"). The goal is to find a

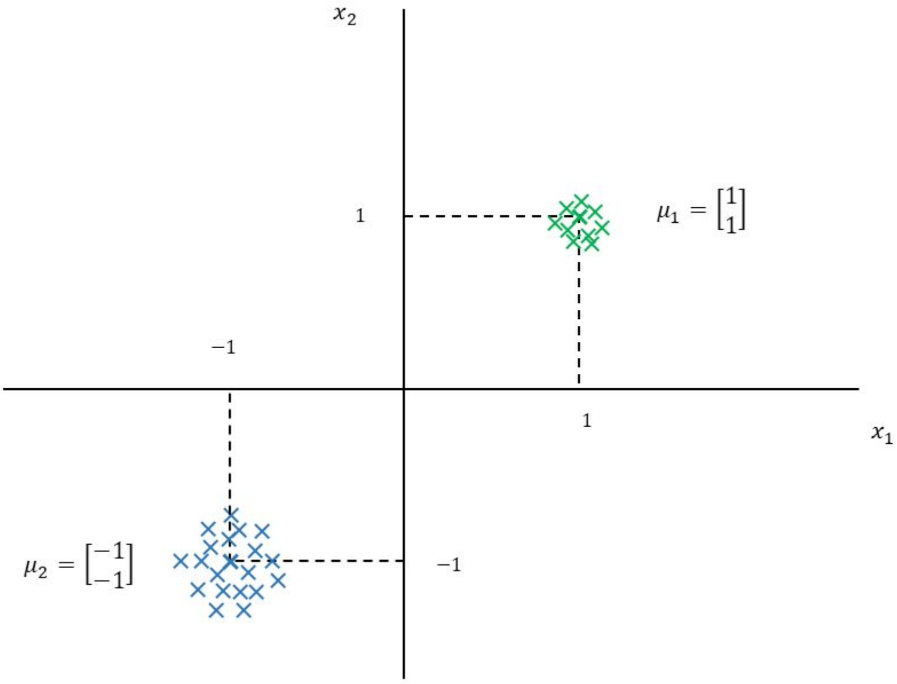

Figure 2: Example 2

"maximum likelihood" estimate of θ.

$$\theta_{\text{MLE}} = \underset{\theta \equiv \{\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}}{\arg\max} \sum_{i=1}^{m} \log p\left(\boldsymbol{x_i} | \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) \tag{2}$$

$$= \underset{\{\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}}{\arg\max} \sum_{i=1}^{m} \log \sum_{z_i \in [k]} p\left(\boldsymbol{x_i}, z_i | \boldsymbol{\phi}, \boldsymbol{\mu}, \Sigma\right) \tag{3}$$

$$= \underset{\{\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}}{\arg\max} \sum_{i=1}^{m} \log \sum_{z_i=1}^{k} \phi(z_i) f(\boldsymbol{x_i} | (z = z_i)) \tag{4}$$

where $\boldsymbol{x_i} | (z = z_i) \sim \mathcal{N}\left(\boldsymbol{\mu}_{z_i}, \Sigma_{z_i}\right)$. This optimization is impossible to solve in closed form over $\{\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$. However, MLE solution is easy if $\{z_i\}_{i=1}^{m}$ were observed.

Case: $\{z_i\}_{i=1}^m$ are observed

In this case,

$$\begin{aligned}
\tilde{\theta}_{\text{MLE}} &= \underset{\{\boldsymbol{\phi},\boldsymbol{\mu},\boldsymbol{\Sigma}\}}{\arg\max} \sum_{i=1}^{m} \log p\left(\boldsymbol{x_i}, z_i | \boldsymbol{\phi}, \boldsymbol{\mu}, \Sigma\right) && (5)\\
&= \underset{\{\boldsymbol{\phi},\boldsymbol{\mu},\boldsymbol{\Sigma}\}}{\arg\max} \sum_{i=1}^{m} \left[\log \phi(z_i) + \underbrace{\log f(\boldsymbol{x_i}|z_i)}_{\sim\mathcal{N}\left(\boldsymbol{\mu}_{z_i},\Sigma_{z_i}\right)}\right] && (6)\\
&= \underset{\{\boldsymbol{\phi},\boldsymbol{\mu},\boldsymbol{\Sigma}\}}{\arg\max} \sum_{i=1}^{m}\sum_{j=1}^{k} \mathbb{1}_{\{z_i=j\}} \left[\log \phi(j) + \underbrace{\log f(\boldsymbol{x_i}|z_i=j)}_{\sim\mathcal{N}\left(\boldsymbol{\mu}_{j},\Sigma_{j}\right)}\right] && (7)\\
&= \underset{\{\boldsymbol{\phi},\boldsymbol{\mu},\boldsymbol{\Sigma}\}}{\arg\max} \left[\sum_{j=1}^{k} \log \phi(j) \sum_{i=1}^{m} \mathbb{1}_{\{z_i=j\}} + \sum_{j=1}^{k}\sum_{i=1}^{m} \mathbb{1}_{\{z_i=j\}} \underbrace{\log f(\boldsymbol{x_i}|z_i=j)}_{\sim\mathcal{N}\left(\boldsymbol{\mu}_{j},\Sigma_{j}\right)}\right] && (8)\\
&= \{\tilde{\boldsymbol{\phi}}, \tilde{\boldsymbol{\mu}}, \tilde{\Sigma}\} && (9)
\end{aligned}$$

where,

$$\tilde{\mu}_j = \frac{\sum_{i=1}^m \mathbb{1}_{\{z_i=j\}} x_i}{\sum_{i=1}^m \mathbb{1}_{\{z_i=j\}}} \tag{10}$$

$$\tilde{\Sigma}_j = \frac{1}{\sum_{i=1}^m \mathbb{1}_{\{z_i=j\}}} \sum_{i=1}^{m} \mathbb{1}_{\{z_i=j\}} \left(x_i - \tilde{\mu}_j\right)\left(x_i - \tilde{\mu}_j\right)^T \tag{11}$$

$$\tilde{\phi}_j = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}_{\{z_i=j\}} \tag{12}$$

Thus if $z_1, z_2 ... z_m$ are observed, we have an efficient way to solve this problem. This observation leads us to an algorithm that solves the ML parameter estimation problem efficiently.

2 EM algorithm

EM algorithm is an iterative algorithm involving two steps in every iteration. In the first step which is called the “E-step”, an arbitrary value for $\theta = (\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is assumed to guess the values for the latent variables $(z_1, z_2, ..., z_m)$. In the next step which is called the M-step, the guessed values for $(z_1, z_2, ..., z_m)$ are used to find the MLE solution for $(\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ which is easy to find as seen in the previous section. The *EM-algorithm* is described in Algo. 1.

In the next section we try to answer 2 fundamental questions related EM-algorithm:

Algorithm 1 EM algorithm

Initialize $(\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ arbitrarily.
while not converged **do**
    E-step:
    $w_{ij} = \mathbb{P}[z_i = j | \boldsymbol{x}_i, \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}]$, $\forall i \in [m], j \in [k]$.
    M-step: Update
    $\forall j \in [k]$.
    $\boldsymbol{\mu}_j = \sum_{i=1}^{m} \left(\frac{1}{\sum_{i=1}^{m} w_{ij}} w_{ij} \boldsymbol{x}_i \right)$, $\boldsymbol{\Sigma}_j = \sum_{i=1}^{m} \left(\frac{1}{\sum_{i=1}^{m} w_{ij}} w_{ij} (\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T \right)$,
    $\phi_j = \frac{1}{m} \sum_{i=1}^{m} w_{ij}$.
Output: $\{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, \phi_j\}$

1. Is there a deeper principle behind EM algorithm?
2. Does it converge?

3 General EM-algorithm

Before getting into the details of the *General EM-algorithm*, lets review the Jensen's inequality which is the tool used in this algorithm.

Definition 3.1. Jensen's Inequality If X is a random variable and $f()$ is a convex function, then

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)].$$

($f()$ is a convex function if $\forall \lambda \in [0,1] f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$)

Suppose we have observations $x_1, x_2, ..., x_m$ where $(x_i, z_i) \overset{i.i.d}{\sim} f(x, z|\theta),\ \theta \in \Theta$, MLE of θ given x is,

$$\begin{aligned} \hat{\theta}_{MLE} &= \underset{\theta \in \Theta}{\arg\max}\ \log L_\theta(x) \\ &= \underset{\theta \in \Theta}{\arg\max} \sum_{i=1}^{m} \log p(x_i|\theta) \\ &= \underset{\theta \in \Theta}{\arg\max} \sum_{i=1}^{m} \log \sum_{z_i} p(x_i, z_i|\theta) \end{aligned}$$

However, the MLE is easy with observed $\mathbf{z} = (z_1, z_2 ... z_m)$, then *EM-algorithm's strategy* is to construct an "easy" uniform lower bound for $L_\theta(x)$ across $\theta \in \Theta$ and maximize it.

For each $i \in [m]$, let Q_i be some distribution for Z. Consider,

$$\begin{aligned}
\log L_\theta(x) &= \sum_{i=1}^{m} \log \sum_{z_i} p(x_i, z_i|\theta) \\
&= \sum_{i=1}^{m} \log \sum_{z_i} Q(z_i) \frac{p(x_i, z_i|\theta)}{Q(z_i)} \\
&\geq \sum_{i=1}^{m} \sum_{z_i} Q(z_i) \log \left[\frac{p(x_i, z_i|\theta)}{Q(z_i)} \right] \qquad \text{(By Jensen's inequality).}
\end{aligned}$$

This uniform lower bound for $\log L_\theta(x)$ is valid for any choice of $Q_1, Q_2, ..., Q_m$. Suppose we choose $Q_1, Q_2, ..., Q_m$ such that the lower bound is tight at some $\theta \in \Theta$. This can be achieved, if the random variable in Jensen's inequality is constant, which in turn implies,

$$\begin{aligned}
\forall i \in [m], \ \frac{p(x_i, z_i|\theta)}{Q_i(z_i)} &= C, \quad \text{(constant not depending on } z_i\text{)} \\
Q_i(z_i) &= \frac{p(x_i, z_i|\theta)}{C}, \\
Q_i(z_i) &= \frac{p(x_i, z_i|\theta)}{\sum_{z_i} p(x_i, z_i|\theta)}, \qquad \forall z_i \\
&= \frac{p(x_i, z_i|\theta)}{p(x_i|\theta)}, \\
&= p(z_i|x_i, \theta),
\end{aligned}$$

which is the posterior probability of z_i given x_i under pdf defined by θ. The *General EM-algorithm* is described in Algo. 2.

3.1 Convergence of EM-algorithm

Claim: Suppose $\theta_t \in \Theta$ and $\theta_{t+1} \in \Theta$ are parameters that are the outputs of 2 successive EM iterations. Then,

$$\log L_{\theta_t}(x) \leq \log L_{\theta_{t+1}}(x).$$

Proof. Consider starting at $\theta_t \in \Theta$. Then, E-step chooses

$$Q_i^{(t)}(z_i) = p(z_i|x_i, \theta_t).$$

Algorithm 2 General EM algorithm

Initialize $\theta \in \Theta$ arbitrarily.
while not converged **do**
    E-step:
    $Q_i(z_i) = p(z_i|x_i, \theta)$, $\forall i \in [m], \forall z_i$
    M-step:
    $\hat{\theta} = \arg\max_{\theta \in \Theta} \sum_{i=1}^{m} \sum_{z_i} Q(z_i) \log \left[\frac{p(x_i, z_i|\theta)}{Q(z_i)} \right]$
Output: $\hat{\theta}$

This makes Jensen's inequality tight at θ_t. Let

$$\log L_{\theta_t}(x) = \sum_{i=1}^{m} \sum_{z_i} Q_i^{(t)}(z_i) \log \left[\frac{p(x_i, z_i|\theta_t)}{Q_i^{(t)}(z_i)} \right] = g(\theta_t).$$

θ_{t+1} is simply the maximizer of $g()$ over $\theta \in \Theta$. Therefore, we must have

$$\log L_{\theta_{t+1}}(x) \overset{Jensen's}{\geq} \sum_{i=1}^{m} \sum_{z_i} Q_i^{(t)}(z_i) \log \left[\frac{p(x_i, z_i|\theta_{t+1})}{Q_i^{(t)}(z_i)} \right] = g(\theta_{t+1}) \geq g(\theta_t) = \log L_{\theta_t}(x).$$

■

Since $\log L_{\theta_t}(x)$ is a monotonically increasing sequence, the algorithm converges to a maximum (local) at infinity.