Machine Learning, Spring 2019
EN 2105
MWF 9-9:50am

Chao Lan
EN 4084A
clan@uwyo.edu

Gaussian Mixture Model, EM Algorithm

Gaussian Mixture Model (GMM) assumes the population is generated from a mixture of K Gaussian distributions $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K$ (Fig 1). It aims to learn these distributions as well as their contributions to the population. It treats K as a hyper-parameter. GMM can be applied for clustering by assigning different instances to different distributions and treat those assigned to one distribution as one cluster.

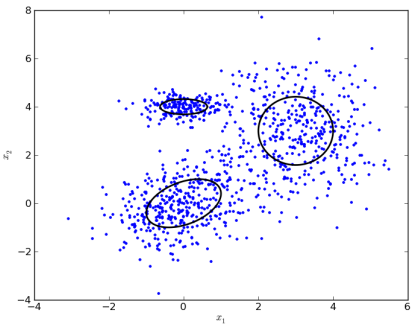

Figure 1: A Mixture of Gaussian Distributions

Specifically, GMM assumes the pdf of instance is

$$p(x) = \sum_{k=1}^{K} p(x, \mathcal{N}_k) = \sum_k p(x \in \mathcal{N}_k) \cdot p(x \mid \mathcal{N}_k) = \sum_k \pi_k \cdot \mathcal{N}(x \mid \mu_k, \Sigma_k), \tag{1}$$

where $\pi_k = p(x \in \mathcal{N}_k)$ measures how likely x is generated from $\mathcal{N}_k$ and $\mathcal{N}(x \mid \mu_k, \Sigma_k) = p(x \mid \mathcal{N}_k)$ is the pdf of x assuming it is generated from $\mathcal{N}_k$. Each $\mathcal{N}_k$ is a component. Note $p(x)$ is a convex combination of $\mathcal{N}_k$'s because

$$\sum_{k=1}^{K} \pi_k = 1. \tag{2}$$

GMM has three sets of parameters π=$\{\pi_k\}$, μ=$\{\mu_k\}$ and Σ=$\{\Sigma_k\}$. Directly optimizing them jointly is not easy as they are correlated in complicated ways. Instead, one may introduce latent variables that are connected to these parameters in simpler ways and optimize both sets of parameters alternately. A common algorithm is Expectation Maximization (EM).

Under constraint (2), the log likelihood function of unknown parameters is

$$\begin{aligned} L_n(\pi, \mu, \Sigma) &= \log p(x_1, \ldots, x_n \mid \pi, \mu, \Sigma) \\ &= \sum_{i=1}^{n} \log p(x_i \mid \pi, \mu, \Sigma) \\ &= \sum_{i=1}^{n} \log \left\{ \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x_i \mid \mu_k, \Sigma_k) \right\}. \end{aligned} \tag{3}$$

Applying the Lagrange multiplier, the augmented objective function is

$$\tilde{L}_n(\pi, \mu, \Sigma) = L_n + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right). \tag{4}$$

Optimizing (4) can be simplified by introducing a 'membership' of x_i in $\mathcal{N}_k$

$$\gamma_{ik} := \frac{\pi_k \cdot \mathcal{N}(x_i \mid \mu_k, \Sigma_k)}{\sum_{k'=1}^{K} \pi_{k'} \cdot \mathcal{N}(x_i \mid \mu_{k'}, \Sigma_{k'})} = p(x_i \in \mathcal{N}_k \mid x_i), \tag{5}$$

and two variables

$$\Gamma_k = \sum\nolimits_{i=1}^{n} \gamma_{ik} \qquad \text{and} \qquad \Gamma = \sum\nolimits_{k=1}^{K} \Gamma_k. \tag{6}$$

[*Exercise*] Verify the last equality of (5).

The connections between introduced parameters and original model parameters can be derived using the critical point method. We will show an example with μ_k and the rest are similar.

Taking derivative of $\tilde{L}_n$ w.r.t. μ_k, we have

$$\frac{\partial}{\partial \mu_k} \tilde{L}_n = \sum\nolimits_{i=1}^{n} \left(\frac{\pi_k \cdot N(x_i \mid \mu_k, \Sigma_k)}{\sum_{k=1}^{K} \pi_k \cdot N(x_i \mid \mu_k, \Sigma_k)} \right) \cdot \left(\Sigma_k^{-1} \cdot (x_i - \mu_k) \right) = \sum\nolimits_{i=1}^{n} \gamma_{ik} \, \Sigma_k^{-1} (x_i - \mu_k). \tag{7}$$

Setting above to zero and solving for μ_k gives

$$\mu_k = \Gamma_k^{-1} \sum\nolimits_{i=1}^{n} \gamma_{ik} x_i. \tag{8}$$

[*Exercise*] Derive (7) and (8).

By similar arguments, we have

$$\Sigma_k = \Gamma_k^{-1} \sum\nolimits_{i=1}^{n} \gamma_{ik} (x_i - \mu_k)(x_i - \mu_k)^T, \tag{9}$$

and

$$\pi_k = \Gamma_k / \Gamma. \tag{10}$$

[*Exercise*] Derive (9) and (10).

Now EM algorithm optimizes the two sets $\{\gamma_{ik}\}$ and $\{\pi, \mu, \Sigma\}$ alternately.

Algorithm 1 EM Algorithm for GMM

Input: a sample S, the number of components K
Initialization: (randomly) initialize π, μ and Σ
while stopping criterion is not met **do**
  E-step
  1: update γ_{ik} for each x_i by (5), using π_k, μ_k and Σ_k
  M-step
  2: update μ_k by (8), using new γ_{ik}
  3: update Σ_k by (9), using new μ_k and γ_{ik}
  4: update π_k by (10), using new γ_{ik}
end while
Output: components $\{\pi, \mu, \Sigma\}$ and membership $\{\gamma_{ik}\}$

GMM can be applied for clustering by assigning x_i to component $\mathcal{N}_k$ (or, cluster k) if

$$\gamma_{ik} \geq \gamma_{ij}, \quad \forall j \neq k. \tag{11}$$

It can also do soft-assignment, i.e., assign x_i to multiple components with different weights γ_{ik}.

EM algorithm is guaranteed to converge to local optimum. Compared to Kmeans, GMM is more flexible as it provides a natural way to do soft-assignment; but it converges more slowly, and one can initialize its centroids using Kmeans to speed up convergence.