Quick Guide to Gaussian Mixture Models & Expectation Maximization Algorithm

A GMM is an unsupervised clustering algorithm that assumes the **dataset is generated** by a mixture of K different Gaussian components

- Basically, it assumes each "cluster" in our dataset follows a unique Gaussian distribution
- We know Gaussian dist. is parametrized by two things: **mean** μ, **standard deviation** σ
- But in a GMM, we also have a π parameter for the **weights** of each Gaussian component (since GMM is a *mixture* of Gaussians)

The goal of a GMM is to fit the best set of parameters for each k'th Gaussian component $\rightarrow$ $[\mu_k \, \sigma_k \, \pi_k]$ in order to characterize our data.

One-Dimensional Example (K = 2 clusters $\rightarrow$ k_1 = A and k_2 = B) - ie. our dataset has two true clusters A & B, goal of GMM is to find the parameters for each

1. *Suppose we know* which point is an A or B $\rightarrow$ we can find the $\mu_k \; \sigma_k$ for each k
2. *Suppose we know* $\mu_k \; \sigma_k$ $\rightarrow$ we can compute which points are likely to be A or B
3. **Unsupervised Learning: UNFORTUNATELY WE KNOW NEITHER (chicken egg problem)**

Then how do we solve this problem?!! —> We use **Expectation Maximization (EM)** !!

EM Algorithm General Intuition:

For each cluster k (so for A or B in this example):

1. Randomly initialize its parameters $\mu_k \; \sigma_k$
2. Given these params $\rightarrow$ Estimate likelihood each datapoint x_i belongs to A, and likelihood each x_i belongs to B
3. Use estimates to update our new parameters
4. Repeat until convergence

EM Maths

Expectation Step

1. Random initialization of parameters for A and B
2. Using randomly initialized parameters of A, compute likelihood that x is an A using Gaussian **probability density function** (and do the same for B):

$$N(x \mid \mu_A, \sigma_A) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Translation: "Given A's parameters, what is the likelihood that x (ie. an observed datapoint) falls in it?"

3. Using $N(x \mid \mu_A, \sigma_A)$, $N(x \mid \mu_B, \sigma_B)$, compute a "responsibility value" r, that x was actually generated by A using **Bayes Theorem** (do same for B): Bayes tells us if we have some *prior* knowledge (in this case, it's the info from step 2), we can update our beliefs based on new evidence, in order to get a *posterior* probability. We divide out case A by the sum over the likelihood of *all* possible hypotheses (ie. both A and B) that could explain the evidence.

$$r_A = N(\mu_A, \sigma_A | x) = \frac{\pi_A N(x \mid \mu_A, \sigma_A)}{\pi_A N(x \mid \mu_A, \sigma_A) + \pi_B N(x \mid \mu_B, \sigma_B)}$$

Translation: "Given our observed datapoint x, what is the likelihood that it is generated by A?"

4. This r_k value is critical. It helps us get our NEW ESTIMATES for μ_k σ_k π_k

Maximization Step

1. Use our r value to update our new parameters with the following:

$$\mu'_k = \frac{\sum_{i=1}^{len(X)} r_{ik}(x_i)}{\sum_{i=1}^{len(X)} r_{ik}} \qquad \sigma'_k = \frac{\sum_{i=1}^{len(X)} r_{ik}(x_i - \mu'_k)}{\sum_{i=1}^{len(X)} r_{ik}} \qquad \pi'_k = \sum_{i=1}^{len(X)} \frac{r_{ik}}{len(X)}$$

2. Substitute these new params for the old random initializations, and repeat until "convergence" (explained below)

BUT… how can we evaluate the goodness of fit of these updated parameters? In other words, how to assess when the GMM "converges"??

→ We use the **Log Likelihood Function (Maximum Likelihood Estimation)!**

- **Goal:** Parameters of the GMM are updated to <u>maximize</u> this log-likelihood function.
- **Recall:** We now want to calculate a value L that tells us how well the current parameters can fit our *entire* dataset X (rather than a single datapoint x_i)
- When the change in this log likelihood value is below a certain threshold, we consider the algorithm to have converged → and finally return the final params of the GMM

$$L(X \mid \mu, \sigma) = \sum_{i=1}^{len(X)} log\left(\sum_{k=1}^{K} \pi_k N(x_i \mid \mu_k, \sigma_k) \right)$$

Breaking it down it says:

- $L(X \mid \mu, \sigma)$ = Fit value for our entire dataset X, given the current parameters for each cluster
- $\sum_{i=1}^{len(X)}$ = Sum starting from the first datapoint to the last
- $\sum_{k=1}^{K} \pi_k N(x_i \mid \mu_k, \sigma_k)$ = weighted sum for the probability for each datapoint belonging to each cluster

- The log part is just a useful trick that allows us to convert products into sums
- Log useful for when we need to multiply a bunch of small probabilities together, the log prevents the products from converging to 0, by instead taking the sum of the logs bc log(a*b) = log(a) + log(b)