The Expectation-Maximization Algorithm for GMMs and other Generative Probabilistic Models

Nishita Pattanayak
ai19btech11024@iith.ac.in

Tanmay Goyal
ai20btech11021@iith.ac.in

Abstract

Gaussian Mixture Models are contained within a set of algorithms commonly referred to as Generative Probabilistic Models. A common assumption made in this model is that every cluster is derived from a Gaussian with certain parameters, which in simple cases can be derived using the Maximum Likelihood Estimate. However, as the complexity increases, the EM algorithm is commonly used. It is an iterative algorithm, that guarantees convergence and also gives greater speed in computation. In general, iterative algorithms guarantee speed and computational brilliance. In this project, we shall attempt to explore the Expectation-Maximization Algorithm for Gaussian Mixture Models in greater detail. We shall also touch upon another algorithm in the Bayesian Paradigm that is similar to the EM algorithm in terms of being an iterative algorithm, the Variational Inference Algorithm, yet has different implications altogether. Finally, we shall discuss the differences between Generative and Discriminative models, touch upon the shortcomings of the GMMs, and discuss other Generative models that help us overcome the same.

1. Introduction

A *Gaussian Mixture* is simply what the name suggests! It is a mixture of various Gaussians which we can index using $k \in \{1, 2, \dots K\}$ where K is the number of clusters, or in this case, the number of Gaussians. This very simply relates to the K-Means algorithm, however, it differs from the K-Means Algorithm in the manner that it is a *soft clustering* problem, i.e each point is assigned a probability of belonging to a certain cluster. Moreover, we can show that K-means is a special case of GMMs. Each Gaussian k in the GMM consists of the following:

1. μ_k: the mean of the Gaussian
2. Σ_k: the covariance of the Gaussian
3. π_k: the contribution of the Gaussian to the mixture.

Note that $\pi_k \in [0, 1]$ and

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

Thus, our Gaussian Mixture Model shall look as follows:

$$\Pr(x|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \tag{2}$$

where $\theta = \{\mu_1, \Sigma_1, \pi_1 \dots, \mu_K, \Sigma_K, \pi_K\}$ are all the parameters. Note that the number of clusters K is not a learnable parameter and is given as an input (i.e. hyperparameter). There are various ways to determine the optimal K using techniques such as the Elbow Method and using metrics such as the AIC, BIC, and inter-cluster separation. However, that shall not be our focus here.

The task involves solving for the optimal parameters. We have various fundamental methods to do so. Let us begin with Maximum Likelihood Estimation.

If we are simply modeling the data to a single Gaussian, then we can use the Maximum Likelihood Estimator. Let the Gaussian's mean and standard deviation be $\mu; \sigma$. Likelihood is defined as the probability of seeing the given data, provided we assume the model parameters are correct. By the product rule of probability, the total likelihood is the product of the individual likelihoods of all N data points, given by:

$$\mathcal{L} = \prod_{i=1}^{N} \Pr(x_i|\mu, \sigma) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \, e^{\frac{-(x_i-\mu)^2}{2\sigma^2}} \tag{3}$$

Maximizing the likelihood is equivalent to minimizing the negative log-likelihood, which means our optimal parameters μ^*, σ^* are given by:

$$\mu^*, \sigma^* = \arg\min_{\mu,\sigma} \sum_{i=1}^{N} \frac{1}{2} \left(\ln 2\pi\sigma^2 + \frac{(x_i - \mu)^2}{2\sigma^2} \right) \tag{4}$$

$$= \arg\min_{\mu,\sigma} \frac{N}{2} \left(\ln 2\pi\sigma^2 \right) + \sum_{i=1}^{N} \frac{(x_i - \mu)^2}{2\sigma^2} \tag{5}$$

Simply setting the partial derivatives 0, we get:

$$\frac{\partial \ln \mathcal{L}}{\partial \mu} = 0 \implies \mu^*_{MLE} = \frac{1}{N}\sum_{i=1}^{N} x_i \quad (6)$$

$$\frac{\partial \ln \mathcal{L}}{\partial \sigma} = 0 \implies \sigma^*_{MLE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \quad (7)$$

Now suppose we have a mixture of Gaussians. Our likelihood shall look as follows:

$$\mathcal{L} = \prod_{i=1}^{N}\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i; \mu_k, \Sigma_k) \quad (8)$$

The log-likelihood, in this case, shall look like this:

$$\ln \mathcal{L} = \sum_{i=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i; \mu_k, \Sigma_k) \right) \quad (9)$$

We see that the problem here arises due to the summation within the logarithm. This is further illustrated by setting the partial derivative w.r.t μ_k equal to 0:

$$\frac{\partial \ln \mathcal{L}}{\partial \mu_k} = 0 \quad (10)$$

which implies

$$\sum_{i=1}^{N} \left[\frac{\pi_k \mathcal{N}(x_i; \mu_k, \Sigma_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i; \mu_k, \Sigma_k)} \times \Sigma_k^{-1}(x_i - \mu_k) \right] = 0 \quad (11)$$

where the Normal Distribution for multiple dimensions is given by:

$$\mathcal{N}(x_i; \mu_k, \Sigma_k) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_k|^{\frac{1}{2}}} e^{\left(\frac{-1}{2}(x_i - \mu_k)^T \Sigma_k^{-1}(x_i - \mu_k)\right)}$$

This is where MLE reaches it's boundaries. This can no longer be solved analytically and brings the *Expectation Maximization Algorithm* into the picture.

2. The Expectation Maximization Algorithm

The Expectation-Maximization Algorithm or the *EM Algorithm* is an iterative solution to finding the optimal parameters. It consists of two steps: the *E step* or the *Expectation step* and the *M step* or the *Maximization step*. Both these steps are repeated iteratively till there is some convergence in the parameter space.

Let us define a latent variable $\mathbf{z} = (z_1, z_2 \ldots, z_K)$, where

$$z_k = \begin{cases} 1 \text{ if data point belongs to cluster } k \\ 0 \text{ otherwise} \end{cases} \quad (12)$$

Here, $\mathbf{z}$ is a $k-$ dimensional binary random vector. We can also state the following:

$$\pi_k = \Pr(z_k = 1) \quad (13)$$

which means the probability of obtaining a data point from a particular Gaussian is equivalent to its contribution to the mixture. We have:

$$\Pr(\mathbf{z}) = \prod_{k=1}^{K} \Pr(z_k = 1)^{z_k} = \prod_{k=1}^{K} \pi_k^{z_k} \quad (14)$$

Also, the probability of obtaining a data sample x_i from a cluster k is given by:

$$\Pr(x_i | z_k) = \begin{cases} 0 \text{ if } z_k = 0 \\ \mathcal{N}(x_i; \mu_k, \Sigma_k) \text{ if } z_k = 1 \end{cases}$$

We can compute the joint probability as

$$\Pr(x_i, \mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k} \mathcal{N}(x_i; \mu_k, \Sigma_k)^{z_k} \quad (15)$$

Thus, given the latent variable $\mathbf{z} = (z_1, z_2 \ldots z_K)$, we have

$$\Pr(x_i | \mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(x_i; \mu_k, \Sigma_k)^{z_k} \quad (16)$$

This is where the latent variables become interesting. The likelihood of the data point can be found by summing over all possible states of $\mathbf{z}$. We shall find the likelihood of the total data X as the probability of seeing the data point over all possible clusters:

$$\Pr(x_i) = \sum_{\mathbf{z}} \Pr(x_i | \mathbf{z}) \Pr(\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i; \mu_k, \Sigma_k)$$

$$\implies \Pr(X) = \prod_{i=1}^{N} \Pr(x_i) = \prod_{i=1}^{N}\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i; \mu_k, \Sigma_k)$$

which is precisely the same as 8.

We have one latent vector $\vec{z_i}$ per data point x_i. Let Z be all the latent vectors. If we knew Z, we could use MLE as we would know which data point belongs to which cluster and then estimate the parameters of each cluster. However, in this case we only know X.

Note that we also have some additional information now. We can determine $\Pr(z_k = 1 | x_i)$, which is the probability of a data point x_i belonging to a certain cluster k given by:

$$\Pr(z_k = 1 | x_i) = \frac{\Pr(x_i | z_k = 1) | pr(z_k)}{\Pr(x_i)} \quad (17)$$

$$= \frac{\pi_k \mathcal{N}(x_i; \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i; \mu_j, \Sigma_j)} \quad (18)$$

$$= \gamma(z_{ik}) \quad (19)$$

Note that, we can now try taking the partial derivatives of the Likelihood wrt μ_k, and it shall be exactly equal to 11:

$$\sum_{i=1}^{N}\left[\frac{\pi_k\mathcal{N}(x_i;\mu_k,\Sigma_k)}{\sum_{k=1}^{K}\pi_k\mathcal{N}(x_i;\mu_k,\Sigma_k)}\times\Sigma_k^{-1}(x_i-\mu_k)\right]=0 \quad (20)$$

$$\implies \sum_{i=1}^{N}\gamma(z_{ik})\Sigma_k^{-1}(x_i-\mu_k)=0 \quad (21)$$

$$\implies \mu_k^*=\frac{\sum_{i=1}^{N}\gamma(z_{ik})x_i}{\sum_{i=1}^{N}\gamma(z_{ik})} \quad (22)$$

where we assume the covariance matrix Σ_k is non-singular. Similarly, we can obtain

$$\implies \Sigma_k^*=\frac{\sum_{i=1}^{N}\gamma(z_{ik})(x_i-\mu_k)(x_i-\mu_k)^T}{\sum_{i=1}^{N}\gamma(z_{ik})} \quad (23)$$

We also wish to obtain π_k. We can simply differentiate w.r.t π_k. However, we need to keep the constraint that $\sum_{k=1}^{K}\pi_k=1$. This can be done using Lagrange Multipliers and maximizing the following quantity:

$$\ln\mathcal{L}+\lambda\left(\sum_{k=1}^{K}\pi_k-1\right) \quad (24)$$

which gives

$$\sum_{i=1}^{N}\left[\frac{\mathcal{N}(x_i;\mu_k,\Sigma_k)}{\sum_{k=1}^{K}\pi_k\mathcal{N}(x_i;\mu_k,\Sigma_k)}+\lambda\right]=0 \quad (25)$$

$$\implies \pi_k\lambda=-\sum_{i=1}^{N}\gamma(z_{ik}) \quad (26)$$

$$\implies \sum_{k=1}^{K}\pi_k\lambda=-\sum_{k=1}^{K}\sum_{i=1}^{N}\gamma(z_{ik}) \quad (27)$$

$$\implies \lambda=-\sum_{k=1}^{K}\sum_{i=1}^{N}\gamma(z_{ik}) \quad (28)$$

using the constraint 1. From this, we get:

$$\pi_k^*=\frac{\sum_{i=1}^{N}\gamma(z_{ik})}{\sum_{k=1}^{K}\sum_{i=1}^{N}\gamma(z_{ik})} \quad (29)$$

Thus, in simple terms: our EM Algorithm is as given in Algorithm 1:

Algorithm 1 Expectation Maximization Algorithm

1. Initialize the means μ_k, the covariance matrix Σ_k and the contributions π_k
2. While convergence is not attained, do
   3. Calculate
   $$\gamma(z_{ik})=\frac{\pi_k\mathcal{N}(x_i;\mu_k,\Sigma_k)}{\sum_{j=1}^{K}\pi_j\mathcal{N}(x_i;\mu_j,\Sigma_j)}$$
   4. Re-evaluate μ_k, Σ_k and π_k as:
   $$\mu_k=\frac{\sum_{i=1}^{N}\gamma(z_{ik})x_i}{\sum_{i=1}^{N}\gamma(z_{ik})}$$
   $$\Sigma_k=\frac{\sum_{i=1}^{N}\gamma(z_{ik})(x_i-\mu_k)(x_i-\mu_k)^T}{\sum_{i=1}^{N}\gamma(z_{ik})}$$
   $$\pi_k=\frac{\sum_{i=1}^{N}\gamma(z_{ik})}{\sum_{k=1}^{K}\sum_{i=1}^{N}\gamma(z_{ik})}$$
   5. Compute the log-likelihood
   $$\ln\mathcal{L}=\sum_{i=1}^{N}\ln\left(\sum_{k=1}^{K}\pi_k\mathcal{N}(x_i;\mu_k,\Sigma_k)\right)$$
   and check for convergence.

There is an alternative view of the EM algorithm. Since we only have the data X, we cannot compute the complete log likelihood $\Pr(X,Z)$, however, we can compute the expectation of this log-likelihood under the posterior distribution of Z. This refers to the E-Step or the *Expectation Step*. Under the M-Step or the *Maximization Step*, we try to maximize this expectation. We define

$$Q(\theta^*,\theta)=\mathbb{E}[\ln\Pr(X,Z|\theta^*)] \quad (30)$$

$$=\sum_{Z}\Pr(Z|X,\theta)\ln\Pr(X,Z|\theta^*) \quad (31)$$

The function $Q(\theta^*,\theta)$ can be shown to be the lower bound on the gain in Likelihood over successive iterations. However, we shall skip the proof here. We can find the joint probability using 15 as:

$$\Pr(X,Z)=\prod_{i=1}^{N}\prod_{k=1}^{K}\pi_k^{z_{ik}}\mathcal{N}(x_i;\mu_k,\Sigma_k)^{z_{ik}} \quad (32)$$

Taking the logarithm, we get

$$\ln\Pr(X,Z)=\sum_{i=1}^{N}\sum_{k=1}^{K}\ln z_{ik}+\ln\left(\pi_k\mathcal{N}(x_i;\mu_k,\Sigma_k)\right) \quad (33)$$

Substituting 19 and 33 into 31, we get

$$Q(\theta^*,\theta)=\sum_{i=1}^{N}\sum_{k=1}^{K}\gamma(z_{ik})\ln\left(\pi_k\mathcal{N}(x_i;\mu_k,\Sigma_k)\right) \quad (34)$$

where we can get rid of z_{nk} using the fact that for an entire $\mathbf{z}$, it is equal to 1 only once. For the maximization step, we simply set

$$\theta^* = \arg\max_{\theta} Q(\theta^*, \theta) \tag{35}$$

On taking partial derivatives of 31, we shall obtain the same results as 22 , 23 and 29.

Thus, our updated Expectation-Maximization Algorithm shall look as given in Algorithm 2:

Algorithm 2 Expectation Maximization Algorithm

1. Initialize the initial parameters θ_{old} which consists of means μ_k, the covariance matrix Σ_k and the contributions π_k
2. While convergence is not attained, do
   3. Calculate $\Pr(Z|X, \theta_{old})$
   4. Evaluate $\theta_{new} = \arg\max_{\theta} Q(\theta, \theta_{old})$
   where $Q(\theta, \theta_{old}) = \sum_{Z} \Pr(Z|X, \theta) \ln \Pr(X, Z|\theta_{old})$
   using the following:

$$\mu_k = \frac{\sum_{i=1}^{N} \gamma(z_{ik}) x_i}{\sum_{i=1}^{N} \gamma(z_{ik})}$$

$$\Sigma_k = \frac{\sum_{i=1}^{N} \gamma(z_{ik})(x_i - \mu_k)^T (x_i - \mu_k)}{\sum_{i=1}^{N} \gamma(z_{ik})}$$

$$\pi_k = \frac{\sum_{i=1}^{N} \gamma(z_{ik})}{\sum_{k=1}^{K} \sum_{i=1}^{N} \gamma(z_{ik})}$$

   5. Compute the log-likelihood and check for any convergence of parameters. Assign $\theta_{old} = \theta_{new}$

3. Observations on Self-Implementation

We decided to code up the algorithm from scratch and decided to compare the GMM method to the K-Means method in SKlearn. The results are shown in 1 , 2 , 3:

We find that while both the algorithms produce identical results in 1 and 2, GMMs using the EM algorithm produce a more intuitive result in 3. For the complete implementation, please refer to https://github.com/tanmaygoyal258/EP4130--Data-Science-Analysis/tree/main/Final%20Project.

4. Pros and Cons of GMMs

The advantages of GMMs are as follows:

1. Flexibility to model a variety of probability distributions due to the use of Normal Distributions.
2. Can model multi-modal distributions as well
3. Easily interpretable
4. Fast and guarantee on convergence.

The disadvantages of GMMs include:

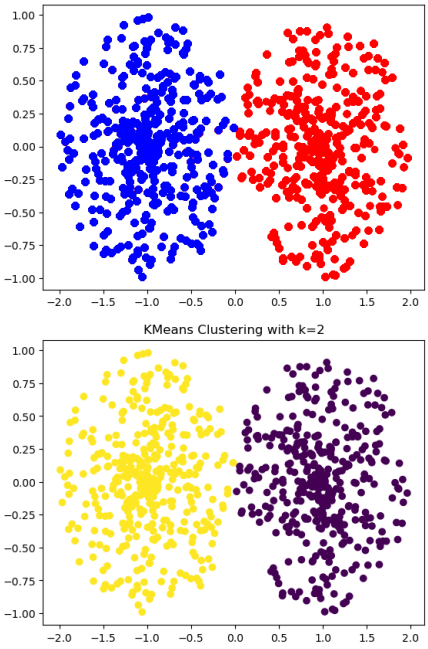

Figure 1. GMMs using EM-algorithm on the top while K-Means algorithm on the bottom

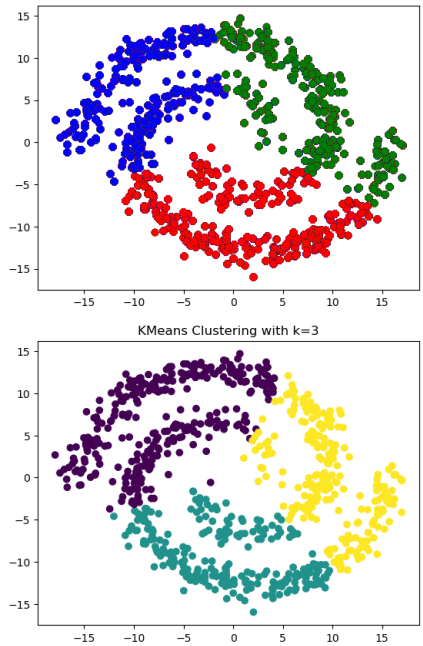

Figure 2. GMMs using EM-algorithm on the top while K-Means algorithm on the bottom

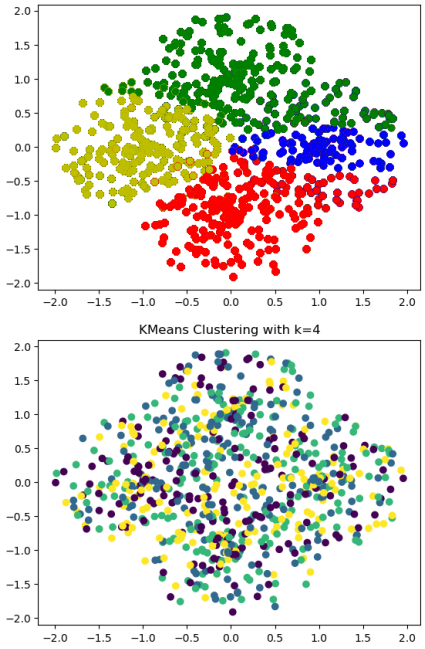

Figure 3. GMMs using EM-algorithm on the top while K-Means algorithm on the bottom

1. Assumption about normality
2. Finding the number of components (hyperparameter)
3. High dimensional data can be challenging to work with

5. Variational Inference and Mean Field Approximation

Variational Inference or Variational Bayes is a technique to compute an approximate distribution to the posterior. It is used when the posterior has hard-to-compute integrals and can not be expressed in a closed form. We then assume the distribution to be approximated has its own set of variational parameters, and by finding the best fit for these variational parameters by optimizing it, we can find the approximated distribution.

Let us assume our theoretical posterior to be found is given by $p(\theta|x)$ where θ is representational of the unknown parameters and/or latent parameters. We assume that $q(\theta)$ is the approximate distribution to p. Then, we wish to reduce the KL divergence between the two distributions.

$$\begin{aligned} KL[q||p] &= \int q(\theta) \log \left[\frac{q(\theta)}{p(\theta|x)} \right] \, d\theta \\ &= \int q(\theta) \log \left[\frac{q(\theta)}{p(\theta, x)} \right] \, d\theta + \log p(x) \end{aligned}$$

$$\implies \log p(x) = KL[q||p] + \mathbb{E}_q[p(\theta, x)] - \mathbb{E}_q[q(\theta)]$$

From here, we get that minimizing the KL divergence between $q(\theta)$ and $p(\theta, x)$ is the same as maximizing the Evidence Lower Bound, or ELBO given by:

$$ELBO(q) = \mathbb{E}_q[p(\theta, x)] - \mathbb{E}_q[q(\theta)]$$

The ELBO is named so because we see that since the KL divergence is non negative, we have:

$$\log p(x) \geq ELBO(q)$$

i.e ELBO is a lower bound on the evidence.

The mean field inference allows us to assume the approximate distribution factorizes as:

$$q(\theta) = q(\theta_1, \theta_2, \dots \theta_n) = \prod_{i=1}^{n} q_i(\theta_i)$$

Thus, our ELBO term now becomes:

$$\int_{\theta_1 \theta_2 \dots \theta_n} \prod_{i=1}^{n} q_i(\theta_i) \left[\log p(\theta, x) - \sum_{i=1} \log q_i(\theta_i) \right] \, d\theta$$

$$= \int_{\theta_j} q_j(\theta_j) \int_{\theta_{m|m \neq j}} \prod_{i \neq j} q_i(\theta_i) \left[\log p(\theta, x) - \sum_{i=1}^{n} \log q_i(\theta_i) \right] \, d\theta$$

$$= \int_{\theta_j} q_j(\theta_j) \left[\int_{\theta_{m|m \neq j}} \prod_{i \neq j} q_i(\theta_i) \log p(\theta, x) \, d\theta_1 d\theta_2 \dots d\theta_n \right] -$$

$$\int_{\theta_j} q_j(\theta_j) \left[\int_{\theta_{m|m \neq j}} \prod_{i \neq j} q_i(\theta_i) \sum_{i=1}^{n} \log q_i(\theta_i) \, d\theta_1 d\theta_2 \dots d\theta_n \right]$$

We denote

$$\mathbb{E}_{m:m \neq j}[\log p(x, \theta)]$$

$$= \int_{\theta_{m|m \neq j}} \prod_{i \neq j} q_i(\theta_i) \log p(\theta, x) d\theta_1 \dots d\theta_{j-1} d\theta_{j+1} \dots d\theta_n$$

Thus, our first term now becomes

$$\int_{\theta_j} q_j(\theta_j) \mathbb{E}_{m:m \neq j}[\log p(x, \theta)] \, d\theta_j$$

Our second term can be written as:

$$\int_{\theta_j} q_j(\theta_j) \log q_j(\theta_j) \int_{\theta_{m|m\neq j}} \prod_{i\neq j} q_i(\theta_i) d\theta_1 \dots d\theta_{j-1} d\theta_{j+1} \dots d\theta_n$$

$$- \int_{\theta_j} q_j(\theta_j) \left[\int_{\theta_{m|m\neq j}} \prod_{i\neq j} q_i(\theta_i) \sum_{i\neq j} \log q_i(\theta_i) \; d\theta_1 d\theta_2 \dots d\theta_n \right]$$

$$= \int_{\theta_j} q_j(\theta_j) \log q_j(\theta_j) d\theta_j - F(\theta_1, \dots \theta_{j-1}, \theta_{j+1} \dots \theta_n)$$

Thus, our overall ELBO term can be written as:

$$ELBO(q) = \int_{\theta_j} q_j(\theta_j) [\mathbb{E}_{m\neq j}[\log p(x,\theta)] - \log q_j(\theta_j)] + F(\theta_{-j})$$

where $\theta_{-j} = \theta_1, \dots \theta_{j-1}, \theta_{j+1}, \dots \theta_n$

Now, we wish to maximize our ELBO term. Using Lagrange Multipliers, we can write the term to be maximized as

$$l = ELBO(q) - \sum_{i=1}^{n} \lambda_i \int_{\theta_i} q_i(\theta_i) \; d\theta_i$$

where the second term involving the Lagrange Multipliers enforce the condition of the probability distributions summing to unity. On taking partial derivative wrt $q_j(\theta_j)$ we get:

$$\begin{aligned} \frac{\partial L}{\partial q_j(\theta_j)} &= q_j(\theta_j)[\mathbb{E}_{m:m\neq j}[\log p(x,\theta)] - \log q_j(\theta_j)] - \lambda_j q_j(\theta_j) \\ &= \mathbb{E}_{m:m\neq j}[\log p(x,\theta)] - \log q_j(\theta_j)] - 1 - \lambda_j \\ &= 0 \end{aligned}$$

Solving this, we get

$$q_j(\theta_j) = \frac{\exp(\mathbb{E}_{m:m\neq j}[\log p(x,\theta)])}{NC}$$

where NC is the normalizing constant.

Since this might involve terms requiring expectations of various variables, which may be unknown or latent, we may not find a closed form in each case. However, we can use an iterative algorithm such as the Expectation-Maximization algorithm to find the solution. This involves starting with an initial guess, and then iteratively updating each variable and using the updated value to update the other.

However, note that the similarity ends at both being an iterative algorithm. There are a number of differences such as the EM algorithm trying to compute the best point estimate, whereas the VI algorithm tries to approximate the posterior distribution. The EM algorithm tries to learn the optimal parameters, whereas VI algorithm tries to learn the "variational parameters" for approximating the parameters, such as priors and hyperpriors.

6. Discriminative v/s Generative Models

In an earlier section, we had mentioned that GMMs are contained within a set of algorithms referred to as *Generative Probabilistic Models*. There is another class of models reffered to as *Discrimincative Probabilistic Models*. In this section, we wish to provide a brief yet clear distinction between the two.

Suppose we are given $X = \{x_1, x_2 \dots x_N\}$ datapoints with labels $Y = \{y_1, y_2, \dots y_N\}$.

Discriminative models are like drawing a boundary separator between the two classes in space. It assumes a functional form for $\Pr(Y|X)$ and then attempts to find the best-fit parameters for this functional form. It is mainly used in the supervised classification domain. They mainly aim to separate classes and make no assumptions about the data. Examples of this kind include Support Vector Machines, Logistic Regression, etc.

Generative Models aim to model the entire distribution. They, too, ultimately return the conditional probability $\Pr(Y|X)$, but do so after assuming functional forms for $\Pr(Y)$ and $\Pr(X|Y)$. They then use Bayes' Theorem to find the conditional probability. This also allows us to generate new data points since we now know the underlying distribution of the data.

GMMs are a generative model because it assumes the data belongs to a mixture of Gaussians, and then we can generate new data points from the same mixture, with the probability of each data point belonging to a particular Gaussian equal to the contribution of the Gaussian to the mixture.

A popular example of a Generative Model is the Generative Adversarial Networks (GANs) that we shall introduce shortly.

7. Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a class of Neural networks that engage in a zero-sum game and with training, can learn to produce data following the same statistics of the training data. A GAN consists of two entities: a Generator and a Discriminator. The purpose of the Generator is to generate data from a distribution, as well as some fake data, and the purpose of the Discriminator is to tell it apart as it learns. Thus, in the zero-sum game, the Generator is trained to try to fool the Discriminator, and the Discriminator is trained to tell apart the Generator's tricks.

Using a simple cross-entropy loss between the real and fake data, the loss function can be defined to be:

$$\mathcal{L}(G, D) = \mathbb{E}_x[\log(D(x))] + \mathbb{E}_z[\log(1 - D(G(z))]$$

Here, x represents all the real data and $D(x)$ is the probability of being real assigned by the Discriminator to real

samples. $G(z)$ is the Generator's output in response to noise z and hence, $D(G(z))$ is the Discriminator's assigned probability of being real assigned to these fake samples generated by the Generator. Finally, an expectation over z includes all fake samples generated.

The Generator cannot impact the first term of the loss function. Since it wishes to fool the discriminator, it shall want the term $D(G(z))$ to be as close to 1 as possible. Thus, the Generator shall try to minimize the loss function, and since the first term is independent of G, minimizing the loss function is equivalent to minimizing $\mathbb{E}_z[\log(1 - D(G(z))]$.

The Discriminator shall try to push $D(x)$ towards one and $D(G(z))$ towards 0. Thus, it shall try to maximize the loss function.

Thus, immediately from the loss function we can see the Generator and Discriminator are trying to minimize and maximize the same quantity, thus, engaging in a zero-sum game. Moreover, these two neural networks can be trained in an alternative fashion, i.e train the Generator while holding the Discriminator constant and train the Discriminator while holding the Generator constant.

Finally, if the parameters for the Generator are represented by ϕ_G and the parameters for the Discriminator are represented by ϕ_D, then

$$\phi_G^*, \phi_D^* = \arg\min_{\phi_G} \max_{\phi_D} \mathcal{L}(G, D)$$

We decided to train a small GAN on data from $\mathcal{N}(0, 1)$ dataset, and then generate standard normal data using the same. The plot showing the true distribution versus the generated distribution is give in 4.

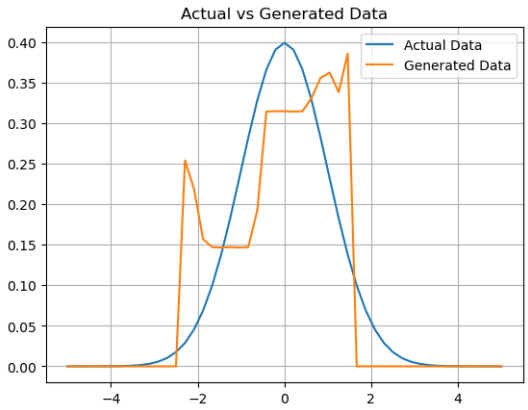

Figure 4. GMMs using EM-algorithm on the top while K-Means algorithm on the bottom

Even though the distributions do not look very similar, the trained model did well enough to generate a unimodal distribution and learned enough to know to ensure the distribution tails off in both directions. We also ran the Shapiro Wilk Test as well as the KS test for this newly generated distribution. Using the Shapiro-Wilk test, we obtained a p-value of order 10^{-8}. This is not very bad considering that the lowest p-value Scipy can provide us is of the order 10^{-33}. With more training and certain tweaks to the model, we can definitely generate a curve closer to the true curve. For the complete implementation, please refer to https://github.com/tanmaygoyal258/EP4130--Data-Science-Analysis/tree/main/Final%20Project.

8. Conclusion

This paper aims to capture one of the most popular Generative Probabilistic Models in literature- the Gaussian Mixture Models, and its associated algorithm- the Expectation-Maximization Algorithm. The paper touches upon it's details, it's advantages and disadvantages, and some observations from implementing the algorithm by ourselves. The focus then shifts to another popular algorithm- the Variational Inference method, which is similar to the EM algorithm by being iterative. However, apart from the repeated iteration, the two have multiple differences. Finally, we cover the difference between Generative and Discriminative Probabilistic models and look upon one of the more fundamental Generative models that also improves upon the shortcomings of GMMs- Generative Adversarial Networks or GANs. We also tried training a simple GAN model from scratch.

9. References

1. Christopher M. Bishop. Pattern recognition and machine learning.
2. https://en.wikipedia.org/wiki/Generative_adversarial_network
3. https://towardsdatascience.com/expectation-maximization-for-gmms-explained-5636161577ca
4. Trevor Hastie Robert Tibshirani Jerome Friedman. The Elements of Statistical Learning