Gaussian Mixture Model

Siddhant Garg

March 2018

1 Introduction

A Gaussian Mixture Model is a probabilistic model for representing the presence of sub-populations (classes) within an overall population, without requiring that the observed data is labeled, ie, we do not know the class of any data point. It is a parametric probability density function represented as a weighted sum of Gaussian component densities. GMM parameters are estimated from training data using the iterative **Expectation-Maximization (EM) algorithm** from a well-trained prior model. A Gaussian Mixture Model clusters the data into different classes and learn the parameters for the probability distributions for each class. Further the Gaussian Mixture Models can be used to classify the data into different classes. This model belongs to unsupervised classifiers category as the labels were not specified during the training.

The general Gaussian Mixture model with K classes is modelled as

$$p(\mathbf{x}) = \sum_{i=1}^{K} \pi_i \mathcal{N}(\mathbf{x}; \boldsymbol{\mu_i}, \boldsymbol{\Sigma_i})$$

$$\sum_{1=1}^{K} \pi_i = 1$$

where $\boldsymbol{\pi_i}$ is the mixing coefficient or the weight and $\boldsymbol{\mu_i}$ and $\boldsymbol{\Sigma_i}$ are the parameters of the gaussians modelling each class. There are several variants on the GMM shown in the above equation. The covariance matrices, $\boldsymbol{\Sigma_i}$, can be full rank or constrained to be diagonal. Additionally, parameters can be shared, or tied, among the Gaussian components, such as having a common covariance matrix for all components, The choice of model configuration (number of components, full or diagonal covariance matrices, and parameter tying) is often determined by the amount of data available for estimating the GMM parameters and how the GMM is used in a particular application. GMMs have been used recently for feature extraction from speech data for use in speech recognition systems. They have also been used extensively in object tracking of multiple objects, where the number of mixture components and their means predict object locations at each frame in a video sequence. The EM algorithm is used to update the component means over time as the video frames update, allowing object tracking.

One hint that data might follow a mixture model is that the data looks multimodal, i.e. there is more than one "peak" in the distribution of data. Trying to fit a multimodal distribution with a unimodal (one "peak") model will generally give a poor fit. Since many simple distributions are unimodal, an obvious way to model a multimodal distribution would be to assume that it is generated by multiple unimodal distributions. For several theoretical reasons, the most commonly used distribution in modeling real-world unimodal data is the Gaussian distribution. Thus, modeling multimodal data as a mixture of many unimodal Gaussian distributions makes intuitive sense. Furthermore, GMMs maintain many of the theoretical and computational benefits of Gaussian models, making them practical for efficiently modeling very large datasets.

2 Problem Statement

We have a Swiss-bank dataset consisting of currency features and their corresponding labels as original or fake. Each data point has 6 features, labeled as "Length", "Left", "Right", "Top", "Bottom" and "Diagonal" and its label as either (1) for fake currency and (0) for original currency, for every data point in the dataset. We will model this data using a Multivariate Gaussian Mixture model with two components and then use EM Algorithm to learn the parameters of the proposed GMM. We will then use our GMM model to classify the data points as fake or original.

3 Expectation-Maximization Algorithm for Gaussian Mixture Model

Given the Gaussian mixture model the aim is to maximize the likelihood function with respect to the parameters comprising the means and covariances of the components and the mixing coefficients.

Given N p-dimensional ($p = 6$) data points, $\boldsymbol{x_1}, \ldots, \boldsymbol{x_N}$ we want to model a Gaussian mixture model. Since the data is categorized into 2 classes only, so our mixture model will be consists of two gaussian distributions. The probability distribution of a data point $\boldsymbol{x}$ is:

$$p(\boldsymbol{x}) = \pi\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1}) + (1-\pi)\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_2}, \boldsymbol{\Sigma_2})$$

where

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} e^{-(\boldsymbol{x}-\boldsymbol{\mu})^{\boldsymbol{T}}\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}$$

To learn the parameters of the above probability distribution we introduce latent variables $\delta_1, \ldots, \delta_N$ to get the complete data $(\boldsymbol{x_1}, \delta_1), \ldots, (\boldsymbol{x_N}, \delta_N)$

$$\text{Let } P\{\Delta = \delta\} = \begin{cases} \pi & \delta = 1 \\ 1-\pi & \delta = 0 \end{cases}$$

$$P\{\boldsymbol{x}|\delta = 1\} = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1})$$

$$P\{\boldsymbol{x}|\delta = 0\} = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_2}, \boldsymbol{\Sigma_2})$$

So therefore,

$$P\{\boldsymbol{x}, \Delta = \delta\} = P\{\boldsymbol{x}|\Delta = \delta\}P\{\Delta = \delta\}$$

Let $\Theta = (\pi, \mu_1, \Sigma_1, \mu_2, \Sigma_2)$, $\theta_1 = (\mu_1, \Sigma_1)$ and $\theta_2 = (\mu_2, \Sigma_2)$
The likelihood function, denoted by $L(\boldsymbol{x}|\Theta)$ can be written as:

$$\begin{aligned} L(\boldsymbol{x}|\Theta) &= \prod_{i=1}^{N} P\{\boldsymbol{x_i}, \delta_i\} \\ &= \prod_{i=1}^{N} \big\{\pi\mathcal{N}(\boldsymbol{x_i}|\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1})\big\}^{\delta_i} \big\{(1-\pi)\mathcal{N}(\boldsymbol{x_i}|\boldsymbol{\mu_2}, \boldsymbol{\Sigma_2})\big\}^{1-\delta_i} \end{aligned}$$

Let $l(\Theta) = \ln L(\boldsymbol{x}|\Theta)$, Then

$$\begin{aligned} l(\Theta) &= \sum_{i=1}^{N} \delta_i \ln\pi + \sum_{i=1}^{N} \delta_i \ln\mathcal{N}(\boldsymbol{x_i}|\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1}) + \sum_{i=1}^{N}(1-\delta_i)\ln(1-\pi) + \sum_{i=1}^{N}(1-\delta_i)\ln\mathcal{N}(\boldsymbol{x_i}|\boldsymbol{\mu_2}, \boldsymbol{\Sigma_2})^{1-\delta_i} \\ &= \sum_{i=1}^{N} \delta_i \ln\pi + \sum_{i=1}^{N}(1-\delta_i)\ln(1-\pi) + \sum_{i=1}^{N} \delta_i \big[-(\boldsymbol{x_i} - \mu_1)^T\Sigma_1^{-1}(\boldsymbol{x_i} - \mu_1)\big] + \\ &\sum_{i=1}^{N}(1-\delta_i)\big[-(\boldsymbol{x_i} - \mu_2)^T\Sigma_2^{-1}(\boldsymbol{x_i} - \mu_2)\big] \end{aligned}$$

The maximum likelihood estimators (MLEs) of the unknown parameters can be obtained by maximizing the log-likelihood function with respect to the unknown parameters. The normal equations can be obtained by taking derivatives of $l(\Theta)$ with respect to π, μ_1, Σ_1, μ_2 and Σ_2, respectively, and equating them to 0.

Let $\hat{\Theta} = (\hat{\pi}, \hat{\mu_1}, \hat{\Sigma_1}, \hat{\mu_2}, \hat{\Sigma_2})$ be the MLEs of the log-likelihood. Based on the complete data, the MLEs can be obtained in expicit forms.

$$\hat{\pi} = \frac{\sum_{i=1}^{N} \delta_i}{N}$$

$$\hat{\mu_1} = \frac{\sum_{i=1}^{N} \delta_i \boldsymbol{x_i}}{\sum_{i=1}^{N} \delta_i} \qquad \hat{\Sigma_1} = \frac{\sum_{i=1}^{N} \delta_i (\boldsymbol{x_i} - \hat{\mu_1})(\boldsymbol{x_i} - \hat{\mu_1})^T}{\sum_{i=1}^{N} \delta_i}$$

$$\hat{\mu_2} = \frac{\sum_{i=1}^{N} (1-\delta_i) \boldsymbol{x_i}}{\sum_{i=1}^{N} (1-\delta_i)} \qquad \hat{\Sigma_2} = \frac{\sum_{i=1}^{N} (1-\delta_i)(\boldsymbol{x_i} - \hat{\mu_1})(\boldsymbol{x_i} - \hat{\mu_1})^T}{\sum_{i=1}^{N} (1-\delta_i)}$$

Conditional Expectation:

$$\begin{aligned} E[\Delta|\boldsymbol{x}] &= 1.P[\Delta = 1|\boldsymbol{x}] + 0.P[\Delta = 0|\boldsymbol{x}] \\ &= \frac{\pi \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1})}{\pi \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1}) + (1-\pi)\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_2}, \boldsymbol{\Sigma_2})} \end{aligned}$$

Based on the above observations the EM-Algorithm can be described as follows:

Step 1: For the 0^{th} iteration, Initialize the means, $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ and the covariances, $\boldsymbol{\Sigma_1}$ and $\boldsymbol{\Sigma_2}$ and the mixing coeficient $\boldsymbol{\pi}$ with random values and evaluate the log-likelihood with these parameters

Step 2: E-Step: For the k^{th} iteration, $k = 1, 2, \ldots,$ calculate the conditional expectations of the latent variables.

$$\begin{aligned} a_i^{(k)} &= E[\delta_i|\boldsymbol{x}, \Theta^{(k-1)}] \\ &= \frac{\pi^{(k-1)} \mathcal{N}(\boldsymbol{x_i}|\boldsymbol{\mu_1^{(k-1)}}, \boldsymbol{\Sigma_1^{(k-1)}})}{\pi^{(k-1)} \mathcal{N}(\boldsymbol{x_i}|\boldsymbol{\mu_1^{(k-1)}}, \boldsymbol{\Sigma_1^{(k-1)}}) + (1-\pi^{(k-1)}) \mathcal{N}(\boldsymbol{x_i}|\boldsymbol{\mu_2^{(k-1)}}, \boldsymbol{\Sigma_2^{(k-1)}})} \forall i = 1, \ldots, N \end{aligned}$$

Step 3: M-Step: For the k^{th} iteration calculate the MLEs of the log-likelihood function from the data and the expected values of the latent variables.

$$\hat{\pi}^{(k)} = \frac{\sum_{i=1}^{N} a_i^{(k)}}{N}$$

$$\hat{\mu_1}^{(k)} = \frac{\sum_{i=1}^{N} a_i^{(k)} \boldsymbol{x_i}}{\sum_{i=1}^{N} a_i^{(k)}} \qquad \hat{\Sigma_1}^{(k)} = \frac{\sum_{i=1}^{N} a_i^{(k)} (\boldsymbol{x_i} - \hat{\mu_1}^{(k)})(\boldsymbol{x_i} - \hat{\mu_1}^{(k)})^T}{\sum_{i=1}^{N} a_i^{(k)}}$$

$$\hat{\mu_2}^{(k)} = \frac{\sum_{i=1}^{N} (1-a_i^{(k)}) \boldsymbol{x_i}}{\sum_{i=1}^{N} (1-a_i^{(k)})} \qquad \hat{\Sigma_2}^{(k)} = \frac{\sum_{i=1}^{N} (1-a_i^{(k)})(\boldsymbol{x_i} - \hat{\mu_2}^{(k)})(\boldsymbol{x_i} - \hat{\mu_2}^{(k)})^T}{\sum_{i=1}^{N} (1-a_i^{(k)})}$$

Repeat the above two steps until convergence.

4 Classification Methods

In this section I, will present two classification methods. First method uses **Maximum Likelihood Estimators** of the Guassian Mixture model to classify the given data. Second method make use of the estimators that we learned using EM-Algorithm.

4.1 Gaussian Mixture Model Classifier

The dataset we used, have the labels of the corresponding data points. Because of the availability of the labels we can use our Gaussian Mixture Model as a classifier. After we have trained our mixture model, we can proceed to classification test. Guassian Mixture models can also be used as classifiers as they separate the data into different clusters. Since we have been provided with the labels of the data points, we can tell which type of currency belong to whch cluster. If we were not provided with the labels, we could not have classified the

clusters as the clusters of fake and original currencies. The only thing that we could have done was to model the data into two clusters without knowing which one of them was the cluster containing original currency and which one contains fake currency. And consequently while classification, we could have only assign a new data point to one the clusters, without making any comment on its originality or fakeness.

A feature vector $\boldsymbol{x}$ is said to belong to class C_i, $i = 1, 2$, if it maximizes $P(C_i|\boldsymbol{x}) = P(\boldsymbol{x}|C_i)P(C_i)$.

$$P(\Delta = \delta|\boldsymbol{x}) = \begin{cases} \pi\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1})/f(\boldsymbol{x}) & \delta = 1 \\ (1-\pi)\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_2}, \boldsymbol{\Sigma_2})/f(\boldsymbol{x}) & \delta = 0 \end{cases}$$

Note that the denominator $f(\boldsymbol{x})$ does not depend on the class to which the $\boldsymbol{x}$ belong to. So, we can only maximize the numerator and ignore the denominator term completely. Therefore, the class C to which the point $\boldsymbol{x}$ belongs to is given by:

$$C = \underset{C_i, i=1,2}{argmax} P(\Delta = \delta|\boldsymbol{x})$$

$$C = \underset{C_i, i=1,2}{argmax}\ \pi_i\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1})/f(\boldsymbol{x}),\ \text{where } \pi_1 = \pi,\ \pi_2 = 1 - \pi$$

Here (π, $\boldsymbol{\mu_1}$, $\boldsymbol{\Sigma_1}$, $\boldsymbol{\mu_2}$, $\boldsymbol{\Sigma_2}$) are same as ($\hat{\pi}$, $\hat{\boldsymbol{\mu_1}}$, $\hat{\boldsymbol{\Sigma_1}}$, $\hat{\boldsymbol{\mu_2}}$, $\hat{\boldsymbol{\Sigma_2}}$) as learned by the EM-Algorithm.

4.2 Maximum Likehood Estimate Classification

Maximum Likelihood Estimate is method of estimating the parameters of the likelihood function given the observations. MLEs are generally easy to compute and they agree with our intuition in most of the examples.
Definition: Given data the maximum likelihood estimate (MLE) for the parameter $\hat{\theta}$ is the value of θ that maximizes the likelihood $P(data|\theta)$. That is, the MLE is the value of θ for which the data is most likely.

In this project we have the data of form $X = \{(\boldsymbol{x_i}, y_i), i = 1, \ldots, n\}$, where $\boldsymbol{x_i}$ is a feature vector and y_i is the corresponding label. The probability of a point $(\boldsymbol{x}, y)$ is

$$P\{\boldsymbol{x}, \Delta = \delta\} = P\{\boldsymbol{x}|\Delta = \delta\}P\{\Delta = \delta\}$$

where,

$$P\{\Delta = \delta\} = \begin{cases} \pi & \delta = 1 \\ 1 - \pi & \delta = 0 \end{cases}$$

$$P\{\boldsymbol{x}|\delta = 1\} = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1})$$

$$P\{\boldsymbol{x}|\delta = 0\} = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_2}, \boldsymbol{\Sigma_2})$$

Let $\Theta = (\pi, \mu_1, \Sigma_1, \mu_2, \Sigma_2)$ The likelihood function $\mathcal{L}(\boldsymbol{X}|\Theta)$ is given by

$$\begin{aligned} \mathcal{L}(\boldsymbol{X}|\Theta) &= \prod_{i=1}^{n} P\{\boldsymbol{x_i}, \Delta = \delta\} \\ &= \prod_{i=1}^{N} \Big\{\pi\mathcal{N}(\boldsymbol{x_i}; \boldsymbol{\mu_1}, \boldsymbol{\Sigma_1})\Big\}^{y_i} \prod_{i=1}^{N} \Big\{(1-\pi)\mathcal{N}(\boldsymbol{x_i}; \boldsymbol{\mu_2}, \boldsymbol{\Sigma_2})\Big\}^{1-y_i} \end{aligned}$$

Let log-likelihood $l(\Theta) = \log \mathcal{L}(\boldsymbol{X}|\Theta)$. Therefore,

$$l(\boldsymbol{X}|\Theta) = \sum_{i=1}^{N} \log \pi y_i + \sum_{i=1}^{N} \log y_i\mathcal{N}(\boldsymbol{x_i}; \boldsymbol{\mu_1}, \boldsymbol{\Sigma_1}) + \sum_{i=1}^{N}(1-\pi)(1-y_i) + \sum_{i=1}^{N}(1-y_i)\mathcal{N}(\boldsymbol{x_i}; \boldsymbol{\mu_2}, \boldsymbol{\Sigma_2})$$

By maximizing the above log-likelihood function, we get the following expressions for the MLEs

$$\pi = \frac{\sum_{i=1}^{N} y_i}{N}$$

$$\hat{\mu_1} = \frac{\sum_{i=1}^{N} y_i \boldsymbol{x_i}}{\sum_{i=1}^{N} y_i} \qquad \hat{\Sigma_1} = \frac{\sum_{i=1}^{N} \delta_i (\boldsymbol{x_i} - \hat{\mu_1})(\boldsymbol{x_i} - \hat{\mu_1})^T}{\sum_{i=1}^{N} \delta_i}$$

$$\hat{\mu_2} = \frac{\sum_{i=1}^{N} (1 - y_i) \boldsymbol{x_i}}{\sum_{i=1}^{N} (1 - y_i)} \qquad \hat{\Sigma_2} = \frac{\sum_{i=1}^{N} (1 - y_i)(\boldsymbol{x_i} - \hat{\mu_1})(\boldsymbol{x_i} - \hat{\mu_1})^T}{\sum_{i=1}^{N} (1 - y_i)}$$

We say that a feature vector $\boldsymbol{x}$ belong to class C_i, $i = 1, 2$, if it maximizes the conditional probability $P(C_i|\boldsymbol{x}) = P(\boldsymbol{x}|C_i)P(C_i)$.

$$P(C = y|\boldsymbol{x}) = \begin{cases} \pi \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1})/f(\boldsymbol{x}) & y = 1 \\ (1 - \pi)\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_2}, \boldsymbol{\Sigma_2})/f(\boldsymbol{x}) & y = 0 \end{cases}$$

Note that the denominator is constant for both the classes. So therefore, we can maximize the numerator. The class C to which the point $\boldsymbol{x}$ belongs to is

$$\hat{C} = \underset{y=0,1}{argmax} P(C = y|\boldsymbol{x})$$

5 Data Analysis

In this section we analyze one data set to see the effectiveness of the proposed model. In this project we used Swiss-bank dataset containing 200 data points. Each data point contains a 6-dimensional feature vector for a currency and its corresponding label as fake(1) and original(0). Since our data is categorized into two classes, we model this data as the mixture model of two Gaussians. We will use EM-Algorithm to learn the weights and parameters of the two Gaussians on 180 data points (90 are fake and 90 are original) and then will test the remaining 20 data points by classifying them as fake or original by using our learned model.

First few data points are presented in the table below:

Labels	**Features**					
	Length	**Left**	**Right**	**Bottom**	**Top**	**Diagonal**
0	214.8	131	131.1	9	9.7	141
0	214.8	129.7	129.7	8.7	9.6	142.2
1	213.9	130.7	130.5	8.7	11.5	137.8
1	214.8	130.1	130	11.4	10.5	139.6

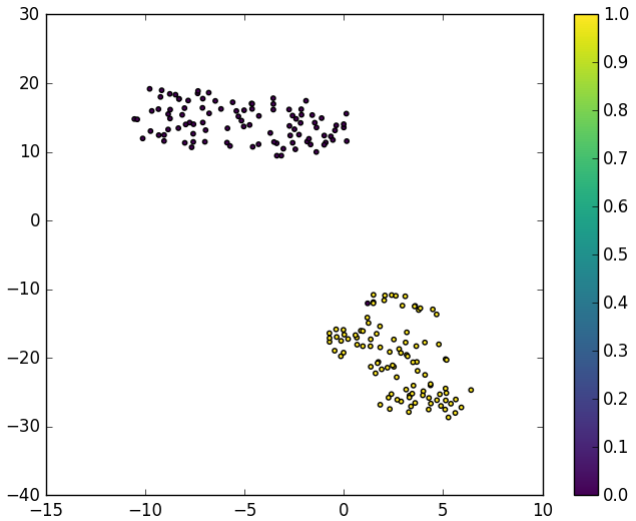

Figure 1: Data representation

It can also be seen pictorially that the given data can be clustered into two different clusters and each cluster can be modelled using separate Gaussian distribution functions for each cluster.

6 Results

In this section I have presented the results of two different classifiers, **Gaussian Mixture Model Classifier** and **Maximum Likelihood Estimate Classifier**. The training is done on the whole data to get the MLEs. During testing, I randomly took 40 data points, 20 data points from each category and compute the accuracy scores for 5 epochs. Then final accuracy score reported is the average of the five accuracy scores obtained.

6.1 Gaussian Mixture Model Classifier

1. **Initialization using the complete data**: I have equal number of data points for each type of currency. So I initialized $\pi^{(0)} = 0.5$. Similarly, I took the sample mean of type (1) data points and type (0) data points and assign them to $\boldsymbol{\mu_1}^{(0)}$ and $\boldsymbol{\mu_2}^{(0)}$ respectively. Similarly, the Σ_1 and Σ_2 are the initialized to the covariance of the type(1) data points and type(2) data points respectively.

$$\mu_1^{(0)} = \frac{\sum_{i=1}^{N} y_i \boldsymbol{x_i}}{\sum_{i=1}^{N} y_i} \qquad \mu_2^{(0)} = \frac{\sum_{i=1}^{N} (1-y_i) \boldsymbol{x_i}}{\sum_{i=1}^{N} (1-y_i)}$$

$$\Sigma_1^{(0)} = \frac{1}{\sum_{i=1}^{N} y_i - 1} \sum_{i=1}^{N} y_i (\boldsymbol{x_i} - \boldsymbol{\mu_1}^{(0)})(\boldsymbol{x_i} - \boldsymbol{\mu_1}^{(0)})^T$$

$$\Sigma_2^{(0)} = \frac{1}{\sum_{i=1}^{N} (1-y_i) - 1} \sum_{i=1}^{N} (1-y_i) (\boldsymbol{x_i} - \boldsymbol{\mu_2}^{(0)})(\boldsymbol{x_i} - \boldsymbol{\mu_2}^{(0)})^T$$

The MLEs of the Gaussian mixture model after 500 iterations of the EM-Algorithm are:

$$\hat{\pi} = 0.581448009517$$

$$\hat{\mu_1} = \begin{pmatrix} 214.98 \\ 130.02 \\ 129.79 \\ 8.35 \\ 10.34 \\ 141.11 \end{pmatrix} \quad \hat{\Sigma_1} = \begin{bmatrix} 0.17 & 0.05 & 0.0429236 & 0.06 & 0.017 & 0.0042 \\ 0.050 & 0.15 & 0.108 & 0.085 & 0.109 & -0.223 \\ 0.0429 & 0.108 & 0.147 & 0.0809 & 0.0887 & -0.195 \\ 0.0597 & 0.085 & 0.0809 & 0.483 & -0.242 & -0.149 \\ 0.0066 & 0.1091 & 0.0887 & -0.242 & 0.601 & -0.486 \\ 0.0042 & -0.2231 & -0.1951 & -0.149 & -0.486 & 1.271 \end{bmatrix}$$

$$\hat{\mu_2} = \begin{pmatrix} 214.77 \\ 130.260 \\ 130.175 \\ 10.903 \\ 11.073 \\ 139.622 \end{pmatrix} \quad \hat{\Sigma_2} = \begin{bmatrix} 0.0781 & 0.0309 & 0.0354 & -0.0525 & 0.039 & 0.0421 \\ 0.031 & 0.067 & 0.048 & 0.0321 & -0.0119 & 0.0386 \\ 0.035 & 0.0478 & 0.092 & -0.021 & 0.0187 & 0.057 \\ -0.0525 & 0.03208 & -0.02105 & 0.714 & -0.4484 & -0.0914 \\ 0.039 & -0.0119 & 0.01879 & -0.448 & 0.4063 & 0.0348 \\ 0.0427 & 0.0386 & 0.0568 & -0.0914 & 0.0348 & 0.118 \end{bmatrix}$$

The **Accuracy Score** of this classifier comes out to **98.01%**

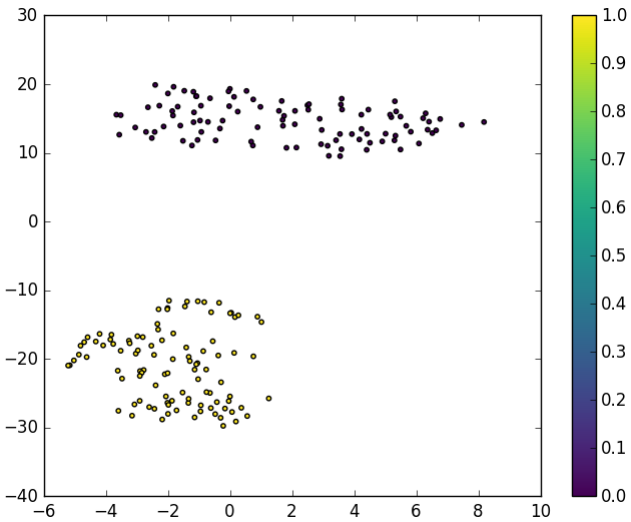

Figure 2: Clusters using the predicted values

2. **Second Initialization** In the second initialization, I introduced a gaussian noise of $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ in the initial mean values of the first initialization. Σ_1 and Σ_2 are taken to be the covariance of the given data as before. Also $pi^{(0)} = 0.5$ is taken.

Therfore,

$$\mu_1^{(0)} = \frac{\sum_{i=1}^{N} y_i \boldsymbol{x_i}}{\sum_{i=1}^{N} y_i} + \mathcal{N}(\mathbf{0}, \boldsymbol{I}_6) \qquad \mu_2^{(0)} = \frac{\sum_{i=1}^{N} (1-y_i) \boldsymbol{x_i}}{\sum_{i=1}^{N} (1-y_i)} + \mathcal{N}(\mathbf{0}, \boldsymbol{I}_6)$$

The MLEs of the Gaussian Mixture model after 500 iterations of the EM-Algorithm are:

$$\hat{\pi} = 0.505$$

$$\hat{\mu_1} = \begin{pmatrix} 214.824 \\ 130.299 \\ 130.193 \\ 10.506 \\ 11.134 \\ 139.451 \end{pmatrix} \qquad \hat{\boldsymbol{\Sigma_1}} = \begin{bmatrix} 0.122 & 0.031 & 0.024 & -0.101 & 0.019 & 0.011 \\ 0.031 & 0.064 & 0.046 & -0.021 & -0.012 & -0.005 \\ 0.024 & 0.046 & 0.087 & -0.018 & 0.000 & 0.034 \\ -0.101 & -0.021 & -0.018 & 1.319 & -0.482 & 0.230 \\ 0.019 & -0.012 & 0.000 & -0.482 & 0.396 & -0.022 \\ 0.011 & -0.005 & 0.034 & 0.230 & -0.022 & 0.305 \end{bmatrix}$$

$$\hat{\mu_2} = \begin{pmatrix} 214.970 \\ 129.941 \\ 129.715 \\ 8.308 \\ 10.158 \\ 141.535 \end{pmatrix} \qquad \hat{\boldsymbol{\Sigma_2}} = \begin{bmatrix} 0.150 & 0.058 & 0.058 & 0.057 & 0.015 & 0.004 \\ 0.058 & 0.132 & 0.085 & 0.057 & 0.046 & -0.038 \\ 0.058 & 0.085 & 0.124 & 0.060 & 0.026 & -0.014 \\ 0.057 & 0.057 & 0.060 & 0.412 & -0.260 & -0.006 \\ 0.015 & 0.046 & 0.026 & -0.260 & 0.410 & -0.055 \\ 0.004 & -0.038 & -0.014 & -0.006 & -0.055 & 0.162 \end{bmatrix}$$

The **Accuracy Score** of this classifier comes out to **99.12 %**

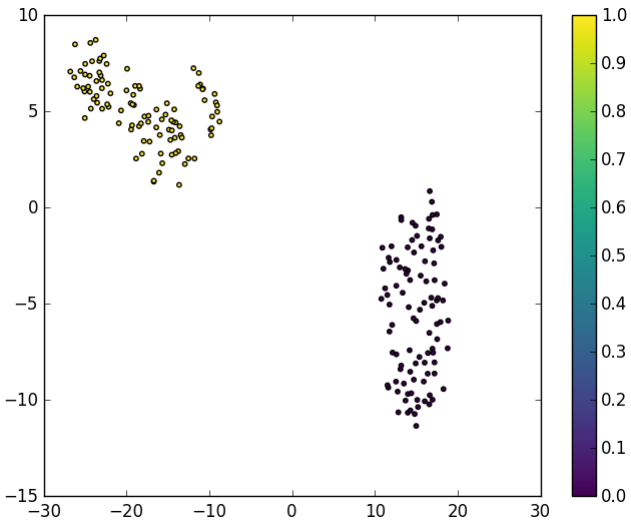

Figure 3: Clusters using the predicted values

Third Initialization : This time I initialized $\mathbf{\Sigma_1}$ and $\mathbf{\Sigma_2}$ to be 6 identity matrix. $\pi^{(0)} = 0.5$ and $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ are initialized as before.

The MLEs of the Gaussian Mixture model after the 500 iterations of the EM-Algorithm are:

$$\hat{\pi} = 0.5103$$

$$\hat{\mu_1} = \begin{pmatrix} 214.824 \\ 130.299 \\ 130.193 \\ 10.505 \\ 11.134 \\ 139.451 \end{pmatrix} \qquad \hat{\Sigma_1} = \begin{bmatrix} 0.122 & 0.031 & 0.024 & -0.101 & 0.019 & 0.011 \\ 0.031 & 0.064 & 0.046 & -0.021 & -0.012 & -0.005 \\ 0.024 & 0.046 & 0.087 & -0.018 & 0.000 & 0.034 \\ -0.101 & -0.021 & -0.018 & 1.319 & -0.482 & 0.230 \\ 0.019 & -0.012 & 0.000 & -0.482 & 0.396 & -0.022 \\ 0.011 & -0.005 & 0.034 & 0.230 & -0.022 & 0.305 \end{bmatrix}$$

$$\hat{\mu_2} = \begin{pmatrix} 214.970 \\ 129.940 \\ 129.715 \\ 8.308 \\ 10.158 \\ 141.536 \end{pmatrix} \qquad \hat{\Sigma_2} = \begin{bmatrix} 0.150 & 0.058 & 0.058 & 0.057 & 0.015 & 0.004 \\ 0.058 & 0.132 & 0.085 & 0.057 & 0.046 & -0.038 \\ 0.058 & 0.085 & 0.124 & 0.060 & 0.026 & -0.014 \\ 0.057 & 0.057 & 0.060 & 0.412 & -0.260 & -0.006 \\ 0.015 & 0.046 & 0.026 & -0.260 & 0.410 & -0.055 \\ 0.004 & -0.038 & -0.014 & -0.006 & -0.055 & 0.162 \end{bmatrix}$$

The **Accuracy Score** of this classifier comes out to **88.5%**

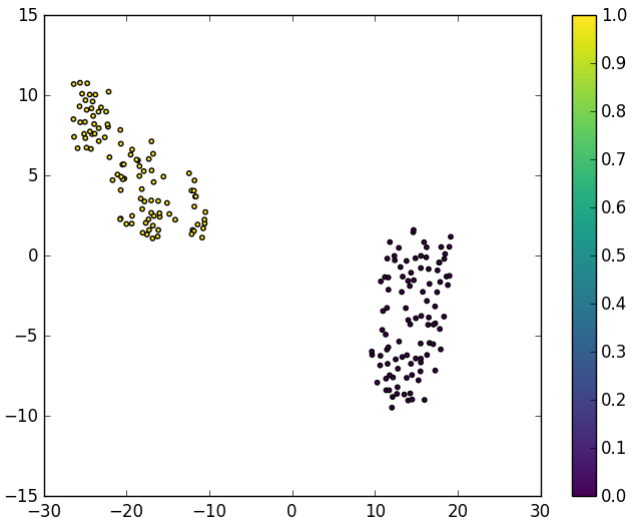

Figure 4: Clusters using the predicted values

6.2 Maximum Likelihood Estimator

The MLEs obtained are :

$$\hat{\pi} = 0.5$$

$$\hat{\mu_1} = \begin{pmatrix} 214.823 \\ 130.300 \\ 130.193 \\ 10.530 \\ 11.133 \\ 139.450 \end{pmatrix} \quad \hat{\Sigma_1} = \begin{bmatrix} 0.124 & 0.032 & 0.024 & -0.101 & 0.019 & 0.012 \\ 0.032 & 0.065 & 0.047 & -0.024 & -0.012 & -0.005 \\ 0.024 & 0.047 & 0.089 & -0.019 & 0.000 & 0.034 \\ -0.101 & -0.024 & -0.019 & 1.281 & -0.490 & 0.238 \\ 0.019 & -0.012 & 0.000 & -0.490 & 0.404 & -0.022 \\ 0.012 & -0.005 & 0.034 & 0.238 & -0.022 & 0.311 \end{bmatrix}$$

$$\hat{\mu_2} = \begin{pmatrix} 214.969 \\ 129.943 \\ 129.720 \\ 8.305 \\ 10.168 \\ 141.517 \end{pmatrix} \quad \hat{\Sigma_2} = \begin{bmatrix} 0.150 & 0.058 & 0.057 & 0.057 & 0.014 & 0.005 \\ 0.058 & 0.133 & 0.086 & 0.057 & 0.049 & -0.043 \\ 0.057 & 0.086 & 0.126 & 0.058 & 0.031 & -0.024 \\ 0.057 & 0.057 & 0.058 & 0.413 & -0.263 & -0.000 \\ 0.014 & 0.049 & 0.031 & -0.263 & 0.421 & -0.075 \\ 0.005 & -0.043 & -0.024 & -0.000 & -0.075 & 0.200 \end{bmatrix}$$

The **Accuracy Score** of this classifier comes out to **99.00%**

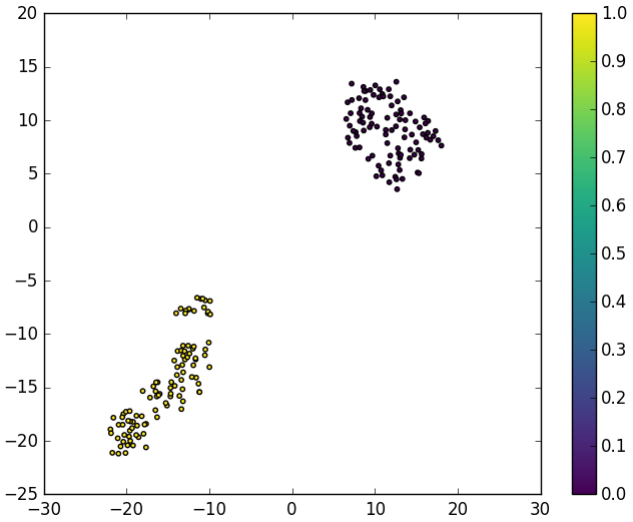

Figure 5: Clusters using the predicted values

7 Conclusions

In this project we trained a 2-component Gaussian Mixture Model and later used it as a classifier. We saw that GMMs belong to class of unsupervised classifiers category. The MLEs obtained using the EM-Algorithm are dependent on the initialization of the paramters. We saw that as the initial parameters become more and more random, the accuracy of the classifier decreases. The accuracy of the **Maximum Likelihood Classifier** is very high on the training data due to overfitting.

8 Code

```
import numpy as np
from scipy.stats import multivariate_normal as mnorm
from sklearn.datasets import make_spd_matrix
from sklearn.mixture import GaussianMixture as GMM
from sklearn.metrics import accuracy_score
import random
import math
import sys
from random import shuffle
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
from tqdm import tqdm
with open("swiss-bank.dat","r") as f:
        a = f.read().strip().split('\n')

b = []
for line in a:
        b.append(line.split(' '))
original = [] #->0
fake = []   #->1
mu1 = np.zeros(6)
mu2 = np.zeros(6)
type1m = np.zeros(6)
type2m = np.zeros(6)
for i in range(1,101):
        x = []
        for j in range(1,len(b[i])):
                x.append(float(b[i][j]))
        original.append(x)
        mu2 += np.array(x)
for i in range(101,201):
        x = []
        for j in range(1,len(b[i])):
                x.append(float(b[i][j]))
        fake.append(x)
        mu1 += np.array(x)

xtrain = fake+original

type1m = mu1/100.0
type2m = mu2/100.0

data = fake + original
data = np.array(data)
labels = [1 for i in range(100)] + [0 for i in range(100)]
X_embedded = TSNE(n_components=2).fit_transform(data)
plt.scatter(X_embedded[:, 0], X_embedded[:, 1], c=labels, s=10, cmap='viridis')
plt.colorbar()
plt.savefig('Clustering with Actual Labels',bbox_inches='tight')
# plt.show()
plt.close()
def EM(xtrain,pi,mu1,sig1,mu2,sig2):
        n = len(xtrain)
        a = np.zeros(len(xtrain))
        for iter in tqdm(range(100)):
                for j,x in enumerate(xtrain):
                        # print(len(x))
```

```
                        f1 = mnorm.pdf(x,mean=mu1,cov=sig1)
                        f2 = mnorm.pdf(x,mean=mu2,cov=sig2)
                        a[j] = pi*f1*1.0/(pi*f1 + (1-pi)*f2)
                # Mstep
                pi = np.sum(a)*1.0/n
                t1=np.zeros(6)
                t2=np.zeros(6)
                for j,x in enumerate(xtrain):
                        t1 += a[j]*np.array(x)
                        t2 += (1-a[j])*np.array(x)
                mu1 = t1*1.0/np.sum(a)
                mu2 = t2*1.0/(n-np.sum(a))

                t1 = np.zeros([6,6])
                t2 = np.zeros([6,6])
                for j,x in enumerate(xtrain):
                        y = (x-mu1).reshape(6,1)
                        z =     (x-mu2).reshape(6,1)
                        t1 += a[j]*np.dot(y,y.T)
                        t2 += (1-a[j])*np.dot(z,z.T)
                sig1 = t1*1.0/np.sum(a)
                sig2 = t2*1.0/(n-np.sum(a))
        return pi,mu1,sig1,mu2,sig2

def classifier(x,pi,mu1,sig1,mu2,sig2):
        f1 = pi*mnorm.pdf(x,mean=mu1,cov=sig1)
        f2 = (1-pi)*mnorm.pdf(x,mean=mu2,cov=sig2)

        if f1>f2:
                return 1
        else:
                return 0

def accuracy(fake,original,p,mu1,sig1,mu2,sig2,model):
        accuracy=0
        data=fake+original
        data = np.array(data)
        # print(data.shape)
        for i in range(5):
                shuffle(fake)
                shuffle(original)
                x = fake + original
                pred = np.array([classifier(y,pi,mu1,sig1,mu2,sig2) for y in x])
                # print(pred.shape)
                # print(pred)
                labels = [1 for i in range(100)]+[0 for i in range (100)]
                accuracy += accuracy_score(pred, labels)
                X_embedded = TSNE(n_components=2).fit_transform(x)
                # predicted = [str(x) for x in pred]
                plt.scatter(X_embedded[:, 0], X_embedded[:, 1], c=pred, s=10, cmap='viridis
                plt.colorbar()
                plt.savefig('Clustering with EM Predicted Labels_' + model + '_{}.png'.form
                plt.close()
                # plt.show()
        accuracy/=5.0
        return accuracy

# First Initialization
pi = 0.5
mu1 = type1m
mu2 = type2m
```

```
sig1 = np.cov(fake,rowvar=False)
sig2 = np.cov(original,rowvar=False)
print("Accuracy MLE : ",accuracy(fake,original,pi,mu1,sig1,mu2,sig2,"MLE"))
p1,mu11,sig11,mu21,sig21 = EM(xtrain,pi,mu1,sig1,mu2,sig2)
print("Accuracy 1 : ",accuracy(fake,original,pi,mu11,sig11,mu21,sig21,"EM1"))
# Second Initialization
pi = 0.5
mu1 = type1m + np.random.multivariate_normal(np.zeros(6),np.identity(6))
mu2 = type2m + np.random.multivariate_normal(np.zeros(6),np.identity(6))
sig1 = np.cov(fake,rowvar=False)
sig2 = np.cov(original,rowvar=False)
p,mu1,sig1,mu2,sig2 = EM(xtrain,pi,mu1,sig1,mu2,sig2)
print("Accuracy 2 : ",accuracy(fake,original,pi,mu1,sig1,mu2,sig2,"EM2"))

# Third Initialization
pi = 0.5
mu1 = type1m + np.random.multivariate_normal(np.zeros(6),np.identity(6))
mu2 = type2m + np.random.multivariate_normal(np.zeros(6),np.identity(6))
sig1 = np.identity(6)
sig2 = np.identity(6)
print(len(sig1))
p,mu1,sig1,mu2,sig2 = EM(xtrain,pi,mu1,sig1,mu2,sig2)
print("Accuracy 3 : ",accuracy(fake,original,pi,mu1,sig1,mu2,sig2,"EM3"))
```

9 Declaration

I, **Siddhant Garg** hereby declare that the work presented in the project report entitled "**Gaussian Mixture Model**" contains my own ideas in my own words. At places, where ideas and words are borrowed from other sources, proper references, as applicable, have been cited. To the best of our knowledge this work does not emanate or resemble to other work created by person(s) other than mentioned herein. The work was created on this 18th day of April 2018.