# ASSIGNMENT 2:- EM and Regression

CS5691:Pattern Recognition and Machine

Course Instructor : Arun Rajkumar

Pragalbh Vashishtha **MM19B012**

February 2022

## 1 Introduction

In this assignment we look over the Expectation Maximization algorithm for a Gaussian Mixture Model, EM for an exponential distribution, K-means and have compared the three.

We further saw the Regression problem and the analytical solution, the least squares solution with gradient descent, stochastic gradient descent, and Ridge Regression

## 2 Q1) EM and Kmeans

### 2.1 Synopsis

Expectation–maximization is an iterative method to find maximum a posteriori estimates of parameters for our model. It is an unsupervised algorithm. It alternates between performing Expectation step , which maximizes the modified log-likelihood evaluated using the current estimate for the parameters, and Maximization step, which computes parameters maximizing the expected log-likelihood.

### 2.2 Q1)i) Expectation Maximization for Data-set's distribution

First we look at the data-set's distribution, for this we plot the histogram of frequency v/s values of data-points. This is the graph we obtain:-

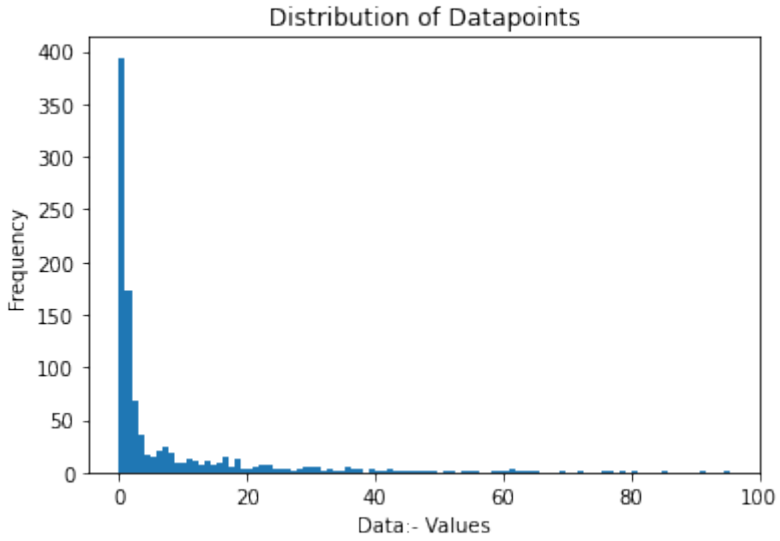


Figure 1: Histogram: Frequency v/s Values

It is evident from the graph that the distribution is akin to a **Exponential Distribution**.

We shall now derive the parameters for the EM for this distribution:-

We assume there are K independent underlying exponential distributions that generate these data points.The log likelihood of the distribution can be written as

$$logL = \sum_{i=1}^{n} ln(\sum_{k=1}^{K} \pi_k \mu_k e^{-\mu_k x_i})$$

Note that the log of this function is linear, hence it is log-concave. Hence we can apply the Jensen's inequality to derive modfied Log Likelihood for our function. Here, we have been given **K=4**.

$$logL_{mod} = \sum_{i=1}^{n} \sum_{k=1}^{K} \lambda_k^i ln(\frac{\pi_k \mu_k e^{-\mu_k x_i}}{\lambda_k^i})$$

Here $\pi_k$ is the probability that a data-point is from k-th distribution. $\lambda_k$ gives us the probability that a data-point is from the k-th distribution, given the data point.

First we fix $\theta := \mu, \pi$ and optimize over $\lambda$.

The PDF now can be treated as a constant. We can then solve using the Lagrange Multiplier method and obtain the equation for the **EXPECTATION** step:

$$\lambda_k^{i(t+1)} = \frac{\pi_k^t \mu_k^t e^{-\mu_k^t x_i}}{\sum_{k=1}^{K} \pi_k^t \mu_k^t e^{-\mu_k^t x_i}}$$

Similarly, by keeping $\lambda$ constant we can derive the parameters $\mu$ and $\pi$ by differentiation of the modified log likelihood. Thus we get the equations for our **MAXIMIZATION** step:

$$\pi_k^{t+1} = \frac{\sum_{i=1}^{n} \lambda_k^{i(t+1)}}{n}$$

Differentiating the modified log likelihood with respect to $\mu$ and setting the derivative to 0 gives us:

$$\sum_{i=1}^{n} (\frac{\lambda_k^{i(t+1)}}{\mu_k} - \lambda_k^{i(t+1)} x_i) = 0$$

$$\mu_k^{t+1} = \frac{\sum_{i=1}^{n} \lambda_k^{i(t+1)}}{\sum_{i=1}^{n} \lambda_k^{i(t+1)} x_i}$$

Note that the expression for $\mu$ is the inverse of the the means in a Gaussian mixture model.

We now perform the EM algorithm for the exponential distribution and observe the following. The algorithm was run with random initializations of the parameters $\mu$, and $\pi$, averaged over 100 iterations :-

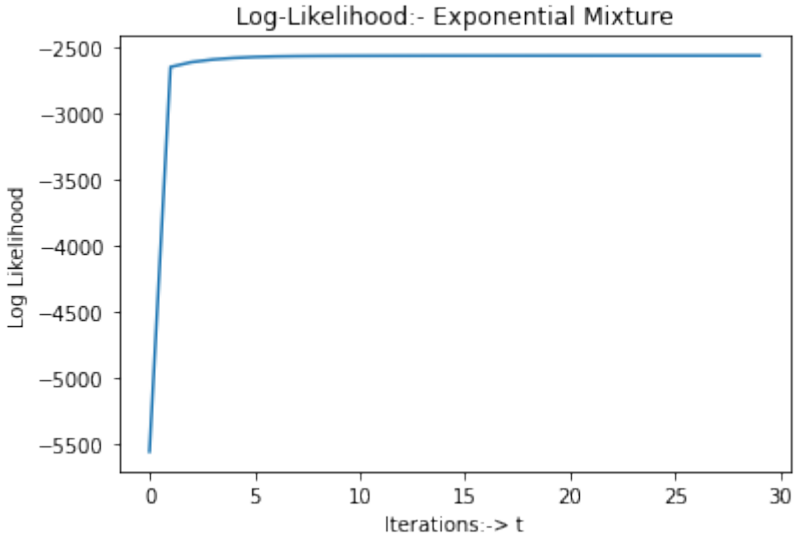


Figure 2: Log-likelihood of Exponential EM - avg 100 runs

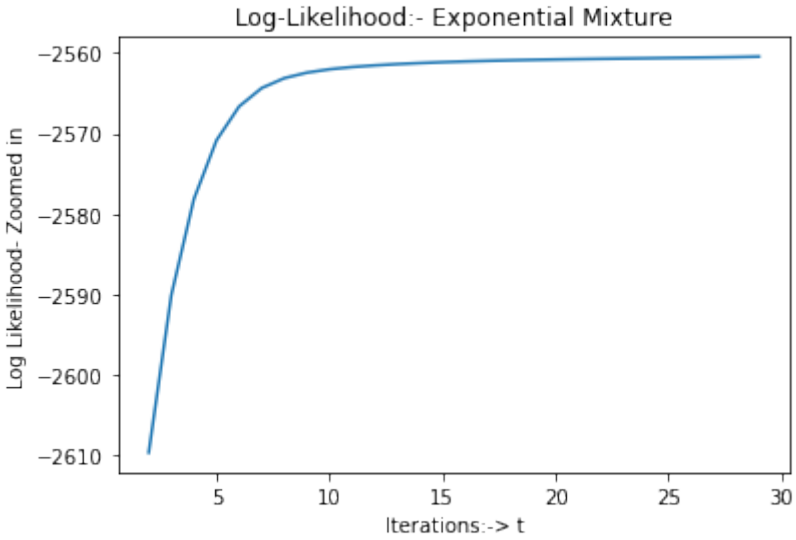


Figure 3: Zoomed:- Log-likelihood of Exponential EM -avg 100 runs

## 2.3 1)ii)

Now we shall be looking at the standard Gaussian Mixture Model. We shall now derive Modified log-likelihood. We know that the log likelihood is:-

$$logL = \sum_{i=1}^{n} ln(\sum_{k=1}^{K} \pi_k e^{\frac{-(x_i-\mu_k)^2}{\sqrt{2\pi\sigma_k}}} * \frac{1}{\sqrt{2\pi\sigma_k}})$$

We use Jensen's inequality to derive modified Log Likelihood **K=4**.

$$logL = \sum_{i=1}^{n} \lambda_k * ln(\frac{\sum_{k=1}^{K} \pi_k e^{\frac{-(x_i-\mu_k)^2}{\sqrt{2\pi\sigma_k}}} * \frac{1}{\sqrt{2\pi\sigma_k}}}{\lambda_k})$$

Again $\pi_k$ is the probability that a data-point is from k-th distribution. $\lambda_k$ is the probability that a data-point is from the k-th distribution, given the data point.

First we fix $\theta := \mu, \sigma, \pi$ and optimize over $\lambda$.

The PDF now can be treated as a constant. We can then solve using the Lagrange Multiplier method and obtain the equation for the **EXPECTATION** step:

$$\lambda_k^{i(t+1)} = \frac{\pi_k^t e^{\frac{-(x_i-\mu_k)^2}{\sqrt{2\pi\sigma_k}}} * \frac{1}{\sqrt{2\pi\sigma_k}}}{\sum_{k=1}^{K} \pi_k^t e^{\frac{-(x_i-\mu_k)^2}{\sqrt{2\pi\sigma_k}}} * \frac{1}{\sqrt{2\pi\sigma_k}}}$$

Similarly, by keeping $\lambda$ constant we can derive the parameters $\mu$, $\sigma$ and $\pi$ by differentiation of the modified log likelihood. Thus we get the equations for our **MAXIMIZATION** step:

$$\pi_k^{t+1} = \frac{\sum_{i=1}^{n} \lambda_k^{i(t+1)}}{n}$$

$$\mu_k^{t+1} = \frac{\sum_{i=1}^{n} \lambda_k^{i(t+1)} x_i}{\sum_{i=1}^{n} \lambda_k^{i(t+1)}}$$

$$\sigma_k^{t+1} = \frac{\sum_{i=1}^{n} \lambda_k^{i(t+1)} (x_i - \mu_k)^2}{\sum_{i=1}^{n} \lambda_k^{i(t+1)}}$$

When putting into practise the following graphs were observed. The algorithm was run with random initializations of the parameters $\mu$, $\sigma$ and $\pi$, averaged over 100 iterations :-

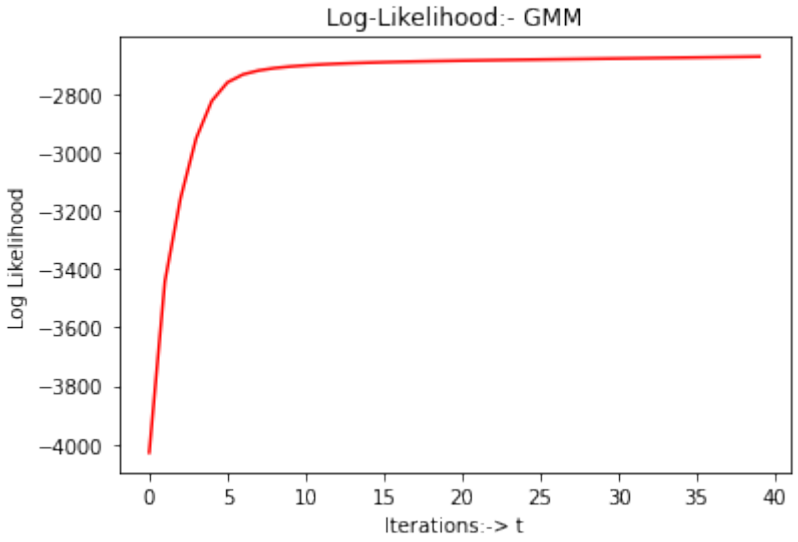


Figure 4: Log-likelihood of GMM - avg 100 iters

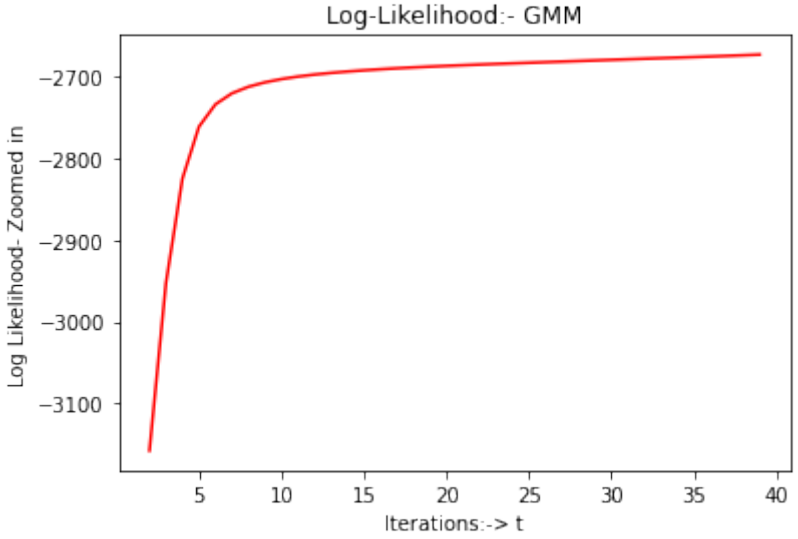


Figure 5: Zoomed:- Log-likelihood of GMM-avg 100 iters

The following observations are made:- The final value of log likelihood for Exponential mixture is higher than Gaussian mixture. Thus data-set is more likely to be generated from an exponential mixture.
The final value for EM-Exponential is -2560.434402326306 while for EM-GMM it is -2672.598883211129.

EM-GMM performs fairly well on the dataset, and presumably on any given dataset, hence it is a widely used.

## 2.4 1)iii) K-means algorithm with K = 4

We run the K-means algorithm and observe the following plots:-

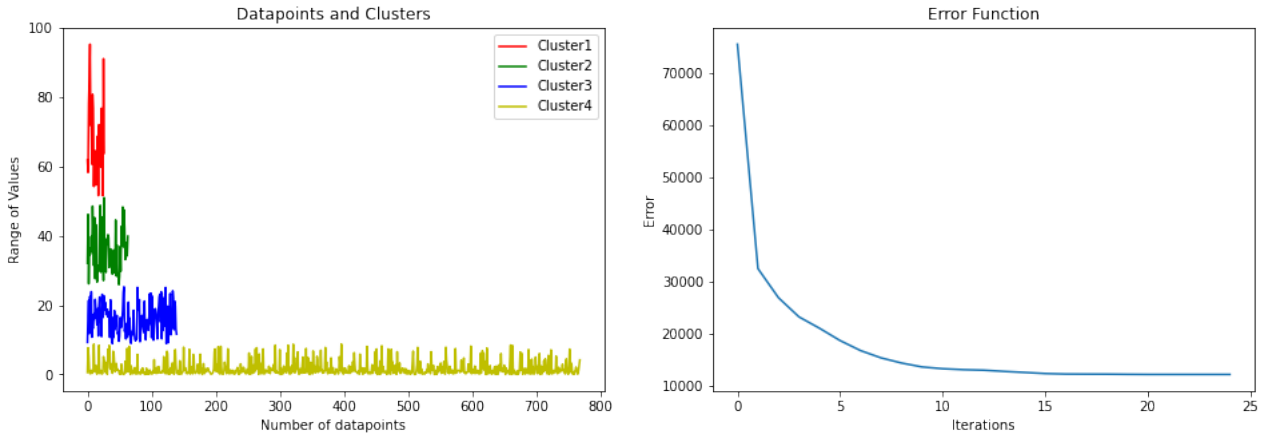


Figure 6: K-means with K=4

### 2.5 1)iv)Best Model?

Analysing the graphs between EM-GMM and EM-Exponential Mixture, we observe the following:-

EM-Exponential converges to a higher log-likelihood value of a value of -2550, while EM-GMM converges to a value of -2700 at best. Also EM-Exponential Converges faster, and hence is a better approximation of the generative model for the data-set.
Now comparing with the K-means algorithm we note that the objective functions are not directly comparable. Converting to hard clustering, k-means seems to perform the best, however the comparison is not fair. Due to the hard clustering nature of the K-means algorithm, and a lack of modelling of the variance, the error function of K-means is very high. This means that the K-means algorithm is performing poorly.

Hence out of the three, EM-Exponential is the best!

## 3 Q2)Regression

### 3.1 Synopsis

In the Regression problem the output variable y is continuous. If we assume there exists a linear relationships between the features of the data, it is converted to a linear regression problem.
Linear Regression has a closed form solution. However for large data-sets (large d values) solving used the closed form becomes tedious, hence gradient descent is used. Gradient descent sometimes is ineffective for even larger data-sets (n is of the order of 100 millions) , hence we employ stochastic gradient descent.

We also look at Ridge regression, which places a penalty on the norm of w. This shrinks the coefficients which then help in generalizing to the test set better.

### 3.2 2)i) Analytical Solution

The analytical solution to Linear Regression is:

$$w_{ML} = (XX^T)^{\dagger}Xy$$

We performed the same on the data-set and derived $w_{ML}$. The dagger symbol is the pseudo-inverse operation.
The MSE obserevd on train data is 0.03968521 while on the test data it is 0.3707273111697886

### 3.3 2)ii) Gradient Descent

For gradient descent various values of step size were tried, a value of 2*e-6 was chosen. For values slightly greater than this, the function explodes, while values smaller than this take a lot of time. A convergence criteria of $||grad(w_t)|| < 1e-10$ was used. When the gradient diminishes it means the descent algorithm has hit a local minima.
Gradient descent is run on the data-set. The gradient update step is given below:-

$$w^{t+1} = w^t - \eta^t[2(XX^T)w^t - 2(Xy)]$$

The following graph of $||w^t - w_{ml}||_2$ was observed:-

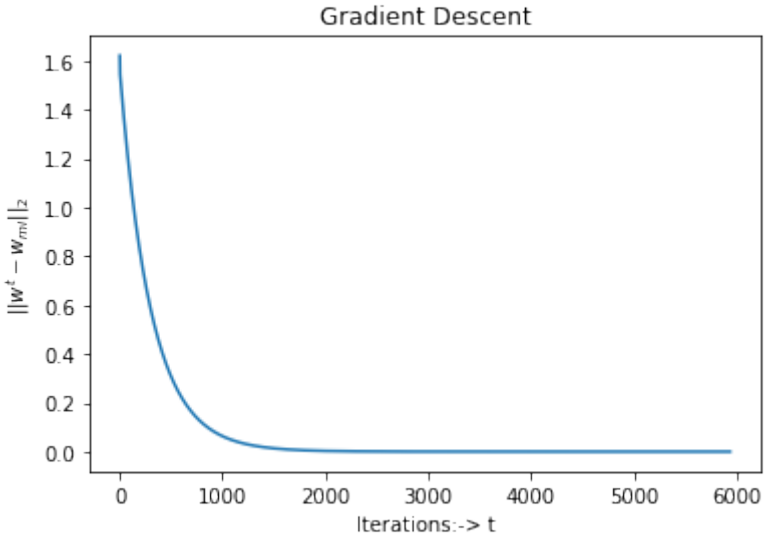


Figure 7: Gradient Descent: $||w^t - w_{ml}||_2$ vs t

From the graph we observe the following;-
The gradient falls very steeply and it slows down as we reach the optimum.
AT around 6000 iterations the value of the error reaches around 1e-9 which is basically 0.
The MSE obtained was 0.039685788 which is very close to that obtained by $w_{ml}$

## 3.4 2iii) Stochastic Gradient Descent

Stochastic Gradient descent is run on the data-set. The stochastic gradient descent algorithm is run using batch size of 100. The gradient update step is given below:-

$$w^{t+1} = w^t - \eta^t[2(\widetilde{X}\widetilde{X}^T)w^t - 2(\widetilde{X}\widetilde{y})]$$

Here $\widetilde{X}$ and $\widetilde{y}$ represent the reduced dataset.
Two methods were used. Directly returning $w_t$ and calculating $w^T = \frac{1}{n}\sum_1^n w^t$

The following graph of $||w^t - w_{ml}||_2$ was observed:-

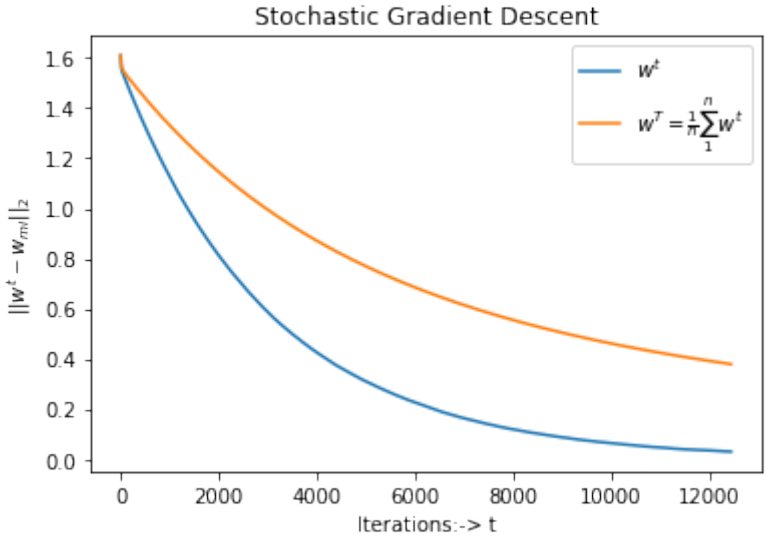


Figure 8: Gradient Descent: $||w^t - w_{ml}||_2$ vs t

The mean of all parameter vectors obtained until t-th iteration, $w_{avg}$ performs slightly worse than just looking at the t-th parameter.

The following observations can be drawn by comparing it to regular gradient descent:- Stochastic Gradient descent converges to a $||w^t - w_{ml}||_2$ of only 0.029972866856289568 after 12000 iterations, while gradient descent converges to a value of 1e-10 after 6000 iterations itself.
Although the value of $||w^t - w_{ml}||_2$ doesn't reduce monotonically, it reduces over short periods with high probability.
Stochastic gradient descent runs way faster, and is computationally more efficient than gradient descent.

### 3.5 2iv) Ridge Regression

Ridge regression introduces the L2 norm regularization. The closed form solution is:

$$w_{ML} = (XX^T + \lambda I)^\dagger Xy$$

.
We run 5-Fold Cross Validation to find the optimal $\lambda$. We plot the various values of $\lambda$ s vs the cross validation:-

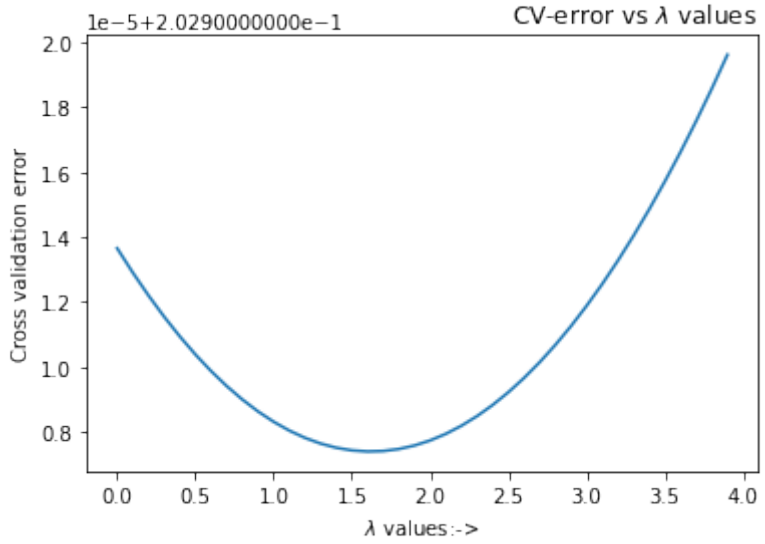


Figure 9: The error in the validation set as a function of $\lambda$

As we note from the graph, $\lambda$ **value of 1.6** is the optimal value for the training data-set.
We then run the gradient descent for ridge using the formula:-
$w^{t+1} = w^t - \eta^t[2(XX^T)w^t - 2(Xy) + 2 * \lambda w^t]$
The following is the performance of the various models on the test data-set. The definition of error is the following;- $\frac{1}{n}\sum_{i=1}^{n}(x_i^T w_i - y_i)^2$

- Analytical solution:- 0.3707273111697886
- Ridge Regression:- 0.37009291813224593
- Extremely Large Penalties ($\lambda$=100) :-0.33526834914411224.

We note that Ridge Regression performs slightly better as it only chooses features which discriminate highly and reduce the error function significantly. Thus it does not learn noise.
However when looking at the test dataset, and trying out a large value of lambda, essentially returning mean of the data, we note that it performs better than ridge. This is maybe because the data generation process is different for the test dataset