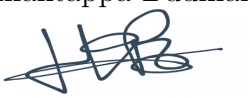
CS5691: Pattern recognition and Machine learning Assignment 2

Name and Signature
Hanumantappa Budihal



1. **Question 1 :** You are given a data-set with 1000 data points generated from a mixture of some distribution in the file A2Q1.csv.

   (a) Determine which probabilisitic mixture could have generated this data (It is not a Gaussian mixture). Derive the EM algorithm for your choice of mixture and show your calculations. Write a piece of code to implement the algorithm you derived by setting the number of mixtures $K = 4$. Plot the log-likelihood (averaged over 100 random initializations) as a function of iterations.

Solution:
The histogram of original data :

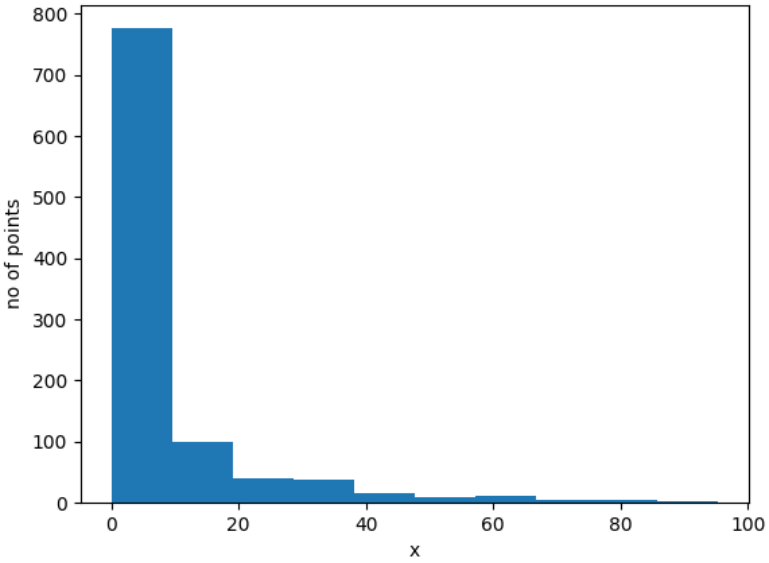
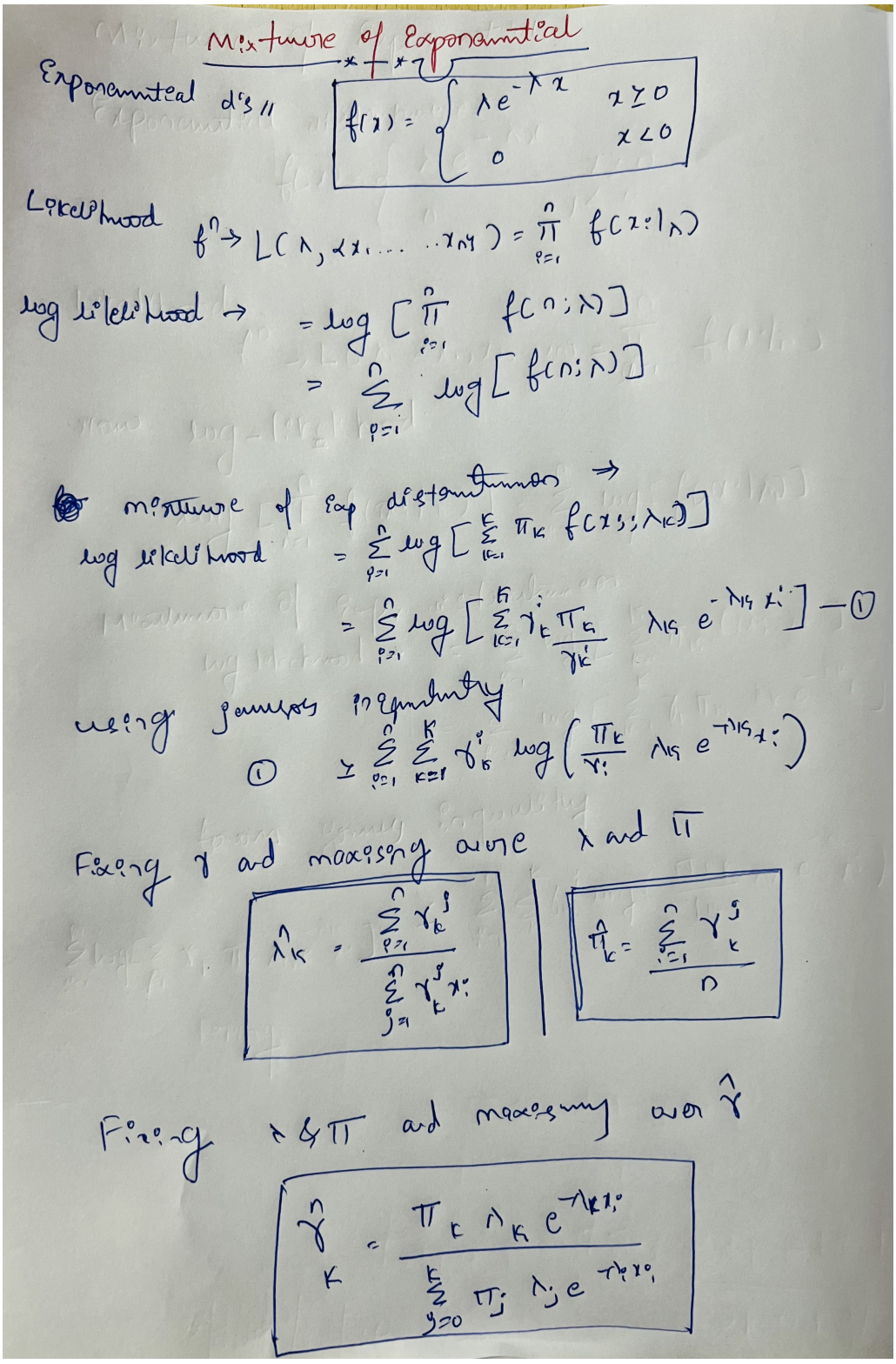


Fig 1 : Original data

Observations : data consists only positive points and the number of data pointsin the bins seems to fall rapidly.

So, data could have been generated by exponential mixture.

Mixture of exponential distribution derivation



Observations:

As given data number of the mixture is 4 (k-4). Below graph shows the likelihood as function of the iteration.

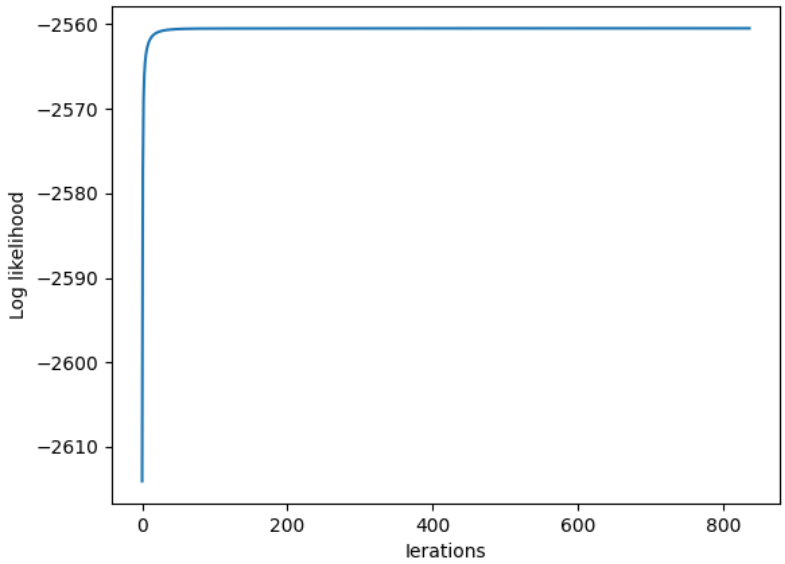


Fig 2 : Log-likelihood for Mixture of exponential

Code : Q1a.py

(b) Assume that the same data was infact generated from a mixture of Gaussians with 4 mixtures. Implement the EM algorithm and plot the log-likelihood (aver- aged over 100 random initializations of the parameters) as a function of iterations. How does the plot compare with the plot from part (i)? Provide insights that you draw from this experiment.Cm

Solution:

Gaussian Mixture Model

Below graph shows the log likelihood as the function of iteration.

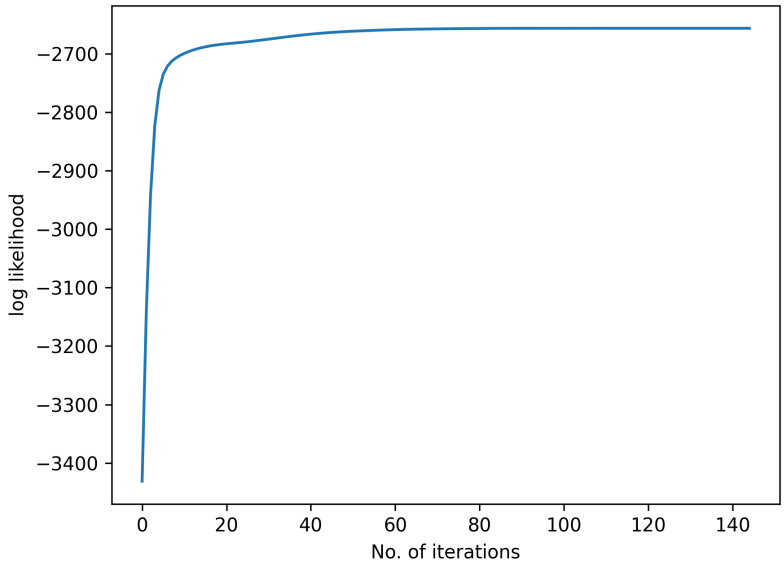


Fig 3 : Log-likelihood for Mixture of Gaussian

Observations:

Assuming data is generated by Gaussian mixture would lead to EM algorithm to converger at the local maximum instead of the global maximum.

EM algorithm is not guarantee to converge at global maximum. As we compare the likelihood of the both Exponential and Gaussian mixture of EM , Gaussian mixture of EM likelihood is converged at the local maximum instead global maximum.

For given data the mixture of exponentials is a better model than mixture of the Gaussian for the data generation.

Code : Q1b.py

(c) Run the K-means algorithm with K = 4 on the same data. Plot the objective of K means as a function of iterations.

Solution:

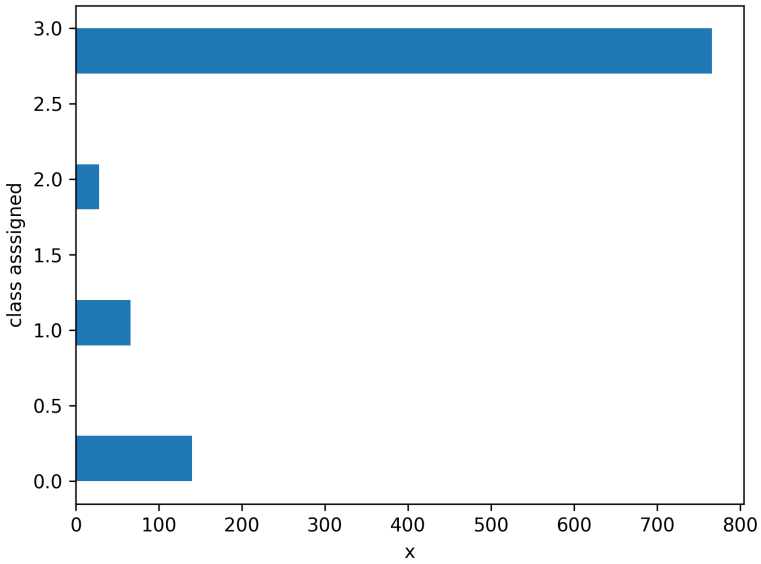


Fig 4 : Cluster assignment for X

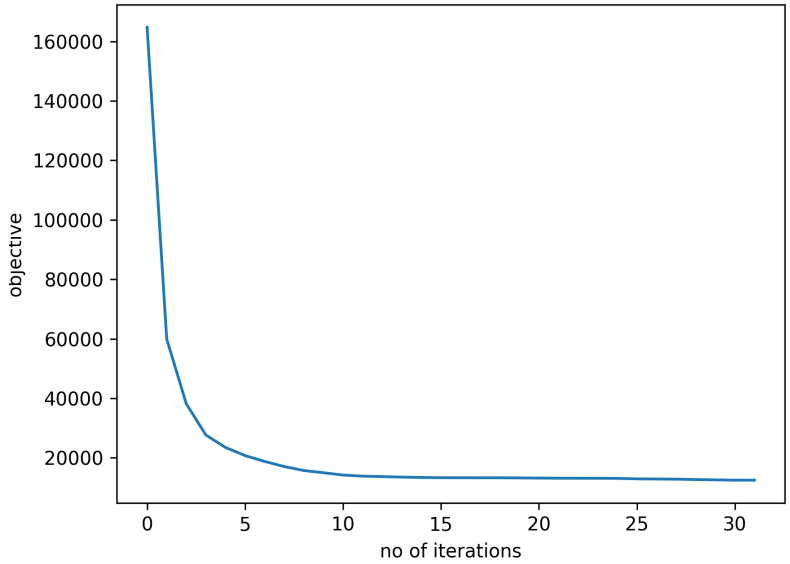


Fig 5 : Objective of the K means

Code : Q1c.py

(d) Among the three different algorithms implemented above, which do you think you would choose to for this dataset and why?

Solution: Good algorithm to generate the given data is : Mixture of Exponentials.

Based on the above experiments, it is clear that mixture of the exponential model is better than the mixture of Gaussian because Gaussian model is converge at the local maximum compare to the Exponential mixture.

Also mixture of Exponential is converge with fewer steps compare to Gaussian model, so it;s faster than exponential.

K means is not provides variances along with the means and not provides the random behavior for the given dataset as compare to exponential mixture mode.

Considering the all points , Mixture of Exponentials would be better choice.

2. . You are given a data-set in the file A2Q2Data train.csv with 10000 points in (R100,R) (Each row corresponds to a datapoint where the first 100 components are features and the last component is the associated y value).

   (a) Obtain the least squares solution w_{ML} to the regression problem using the analytical solution.

Solution: Least squares solution w_{ML} using analytical method is

$$w_{ML} = (X^T X)^{-1} X^T Y$$

Code : Q2a.py

   (b) Code the gradient descent algorithm with suitable step size to solve the least squares algorithms and plot $\|w^t\text{-}w_{ML}\|2$ as a function of t. What do you observe?.

Solution: Analytical solution require the large amount of computation if we have large no of components and have very large no of data points, as computation of $= (X^T X)^{-1}$ becomes highly expensive, while for gradient descent we do not have to compute inverse, this reduces the computation to large amount for if we have large no of dimension.

Gradient Descent algorithm is used to find the minima or slope of function of w. Consider we run a total of T iteration, and we are currently in the t^th iteration. Then:

$$w^t + 1 = w^t - \eta \nabla f(w^t)$$

The algorithm follows the above step until convergence.
In the above equation, η = Learning rate or step size. We have used $= 0.000007$ in our algorithm.

Observations:

The value of learning rate used is optimal because it is the largest step value for which gradient descent algorithm does not grow exponentially.

The algorithm converges after 1000 iterations for $= 0.000007$. For values of learning rate which are too small, the algorithm takes a much longer time to reach convergence.

For values of $\eta > 0.0001$, the algorithm behaves erratically and becomes exponential. For example, $= 0.001$ produces an exponential graph

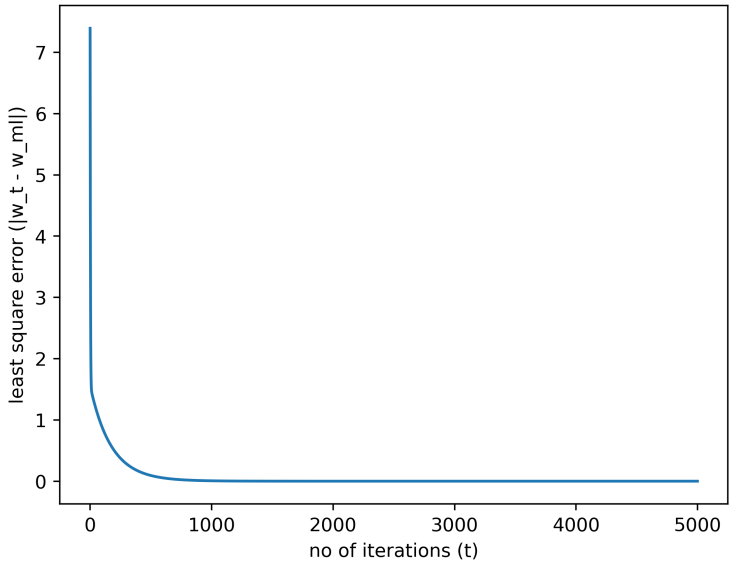


Fig 3 : least square error V/s no of iterations

The main takeaway from the gradient descent algorithm is that it is an ideal fit for datasets where number of data points is less.

We compute gradient step considering all n data points, and this is not feasible for large datasets with more than a million data points

Code : Q2b.py

(c) Code the stochastic gradient descent algorithm using batch size of 100 and plot $\|w^t\text{-}w_{ML}\|2$ as a function of t. What are your observations?

Solution: Stochastic gradient descent differs from gradient descent in the sense that it uses only a small "bunch" of the original data points, to calculate step size for next iteration. This is called a batch of the data, and for each iteration we sample the same number of data points uniformly at random.
Here size of batch N = 100.

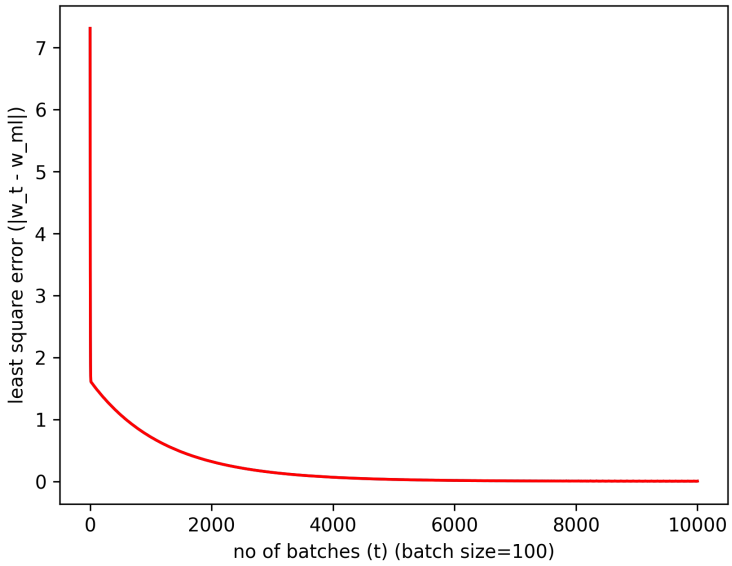


Fig 3 : Gradient Descent: least square error vs no of iterations

Observations:

Stochastic gradient descent is computationally much faster since it uses fewer data points per iteration to calculate step size.

Here, we use a value of N = 100. The faster computation comes at the cost of convergence, since stochastic gradient descent takes longer to converge than gradient descent. For = 0.0001, GD converges in 1000 iterations while SGD takes more than 9000 iterations to converge.

Therefore, it is a computationally faster but convergent slower algorithm.

SGD behaves similar to GD algorithm with respect to values. If is too small, the algorithm will take longer to converge, whereas for if is too large the algorithm may not converge at all.

Code : Q2c.py

(d) Code the gradient descent algorithm for ridge regression. Cross-validate for various choices of and plot the error in the validation set as a function of . For the best chosen, obtain wR. Compare the test error (for the test data in the file A2Q2Data test.csv) of wR with wML. Which is better and why?

Solution: plot of vs error in validation set is shown in below fig

Here we have used cross validation for training and validating set from given training set

for , $\lambda = 2.0$ we are having minimum error in validation set.

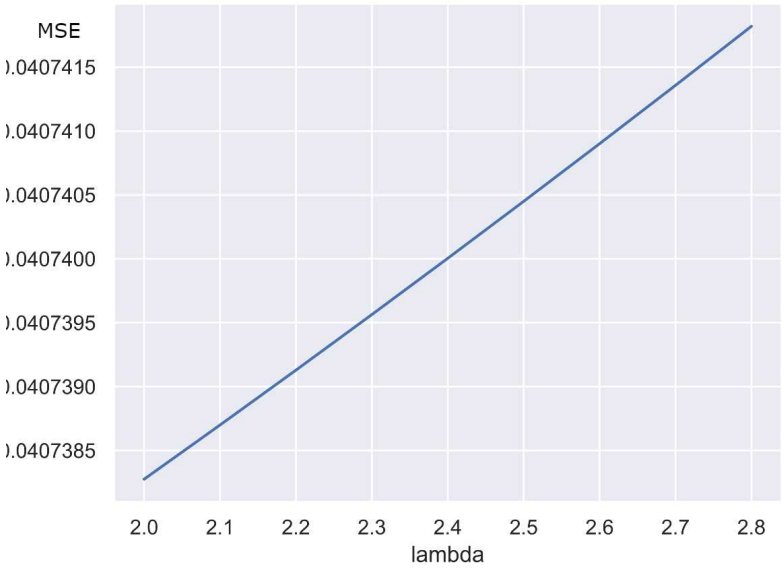


Fig 3 : Error in the validation set

For the test data, MSE error for $W_ML = 0.3707273$
For the test data, MSE error for $W_r = 0.36279045$

Hence we have less error for W_r as compared to the error w.r.t W_{ML} so ridge regression is better as compared to analytic solution because, as analytic method is unbiased and it tries to minimize the error for training set as much as possible but this may lead to overfitting issues, while ridge regression is biased and it tries to minimize the sum of bias + variance hence it tries to find a balance for bias and variance , so it avoids the overfitting of mode

Code : Q2d.py