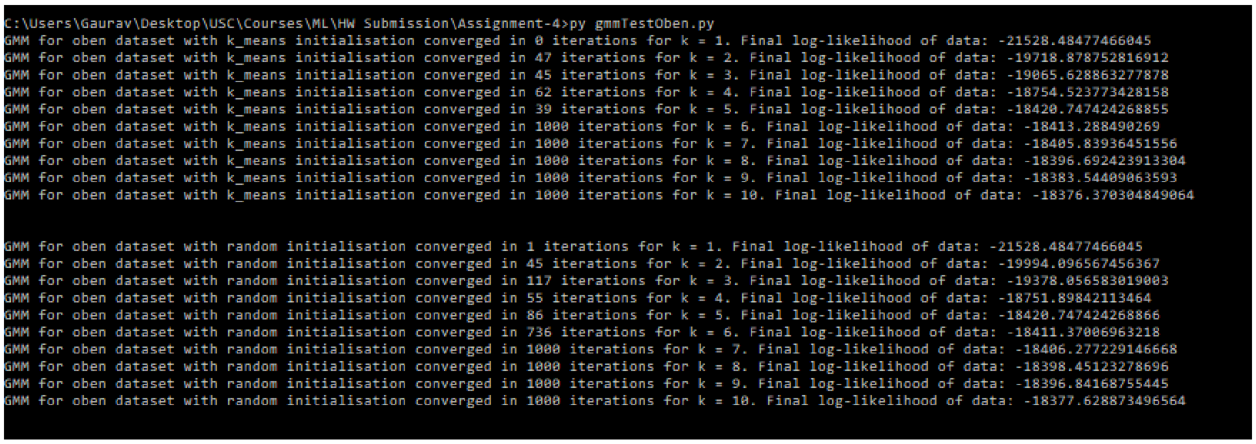
# Oben Assignment – Gaurav Bose

After Running the GMM code given in the GMM notebook, I have found that the number of clusters for the given observations is 5.

The outputs are as follows:



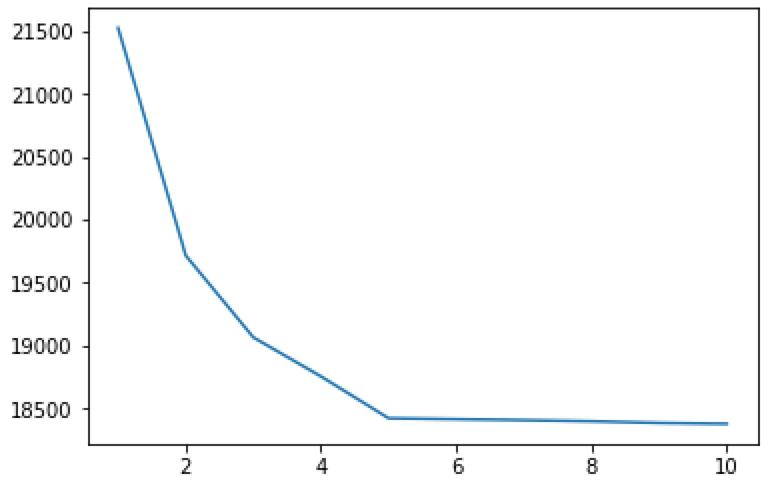
```
C:\Users\Gaurav\Desktop\USC\Courses\ML\HW Submission\Assignment-4>py gmmTestOben.py
GMM for oben dataset with k_means initialisation converged in 0 iterations for k = 1. Final log-likelihood of data: -21528.48477466045
GMM for oben dataset with k_means initialisation converged in 47 iterations for k = 2. Final log-likelihood of data: -19718.878752816912
GMM for oben dataset with k_means initialisation converged in 45 iterations for k = 3. Final log-likelihood of data: -19065.628863277878
GMM for oben dataset with k_means initialisation converged in 62 iterations for k = 4. Final log-likelihood of data: -18754.523773428158
GMM for oben dataset with k_means initialisation converged in 39 iterations for k = 5. Final log-likelihood of data: -18420.747424268855
GMM for oben dataset with k_means initialisation converged in 1000 iterations for k = 6. Final log-likelihood of data: -18413.288490269
GMM for oben dataset with k_means initialisation converged in 1000 iterations for k = 7. Final log-likelihood of data: -18405.83936451556
GMM for oben dataset with k_means initialisation converged in 1000 iterations for k = 8. Final log-likelihood of data: -18396.692423913304
GMM for oben dataset with k_means initialisation converged in 1000 iterations for k = 9. Final log-likelihood of data: -18383.54409063593
GMM for oben dataset with k_means initialisation converged in 1000 iterations for k = 10. Final log-likelihood of data: -18376.370304849064


GMM for oben dataset with random initialisation converged in 1 iterations for k = 1. Final log-likelihood of data: -21528.48477466045
GMM for oben dataset with random initialisation converged in 45 iterations for k = 2. Final log-likelihood of data: -19994.096567456367
GMM for oben dataset with random initialisation converged in 117 iterations for k = 3. Final log-likelihood of data: -19378.056583019003
GMM for oben dataset with random initialisation converged in 55 iterations for k = 4. Final log-likelihood of data: -18751.89842113464
GMM for oben dataset with random initialisation converged in 86 iterations for k = 5. Final log-likelihood of data: -18420.747424268866
GMM for oben dataset with random initialisation converged in 736 iterations for k = 6. Final log-likelihood of data: -18411.37006963218
GMM for oben dataset with random initialisation converged in 1000 iterations for k = 7. Final log-likelihood of data: -18406.277229146668
GMM for oben dataset with random initialisation converged in 1000 iterations for k = 8. Final log-likelihood of data: -18398.45123278696
GMM for oben dataset with random initialisation converged in 1000 iterations for k = 9. Final log-likelihood of data: -18396.84168755445
GMM for oben dataset with random initialisation converged in 1000 iterations for k = 10. Final log-likelihood of data: -18377.628873496564
```

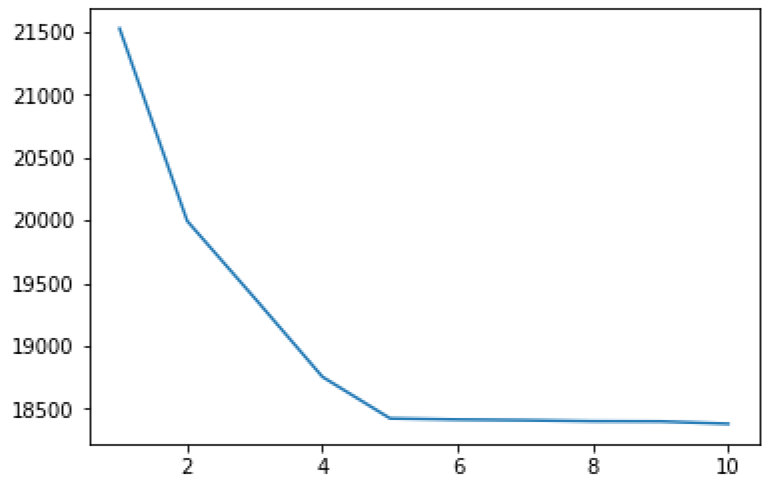
For better reading, the same has been executed in Jupyter Notebook.

The graphs plotted on Jupyter notebook show the following.

For K means initialization:



For random initialization:



In either case, the elbow method shows the significant drop and lowest point as 5.

Therefore the number of clusters is 5.

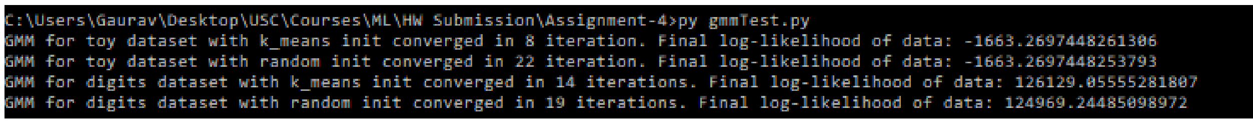
- Question 1:

  Other algorithms such as MM or ME which are Maximization-Maximization or Maximization-Expectation can be used instead of EM. MM is faster whereas ME is more accurate.
- Question 2:

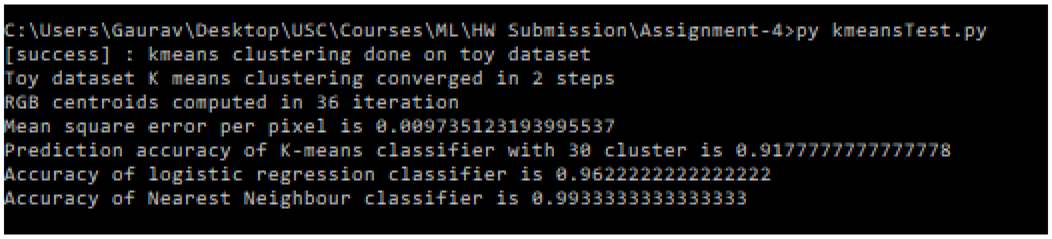
  The main limitation of GMM is that it can fail to work if the dimensionality of the problem is too high. SVM works better or perform PCA before evaluating it for GMM. Also, number of mixture models need to be determined.
- Question 3:

  K means is a narrowed down unique version of GMM. K means start by assigning all to one cluster and then depends on "hard assignment" where we are certain a data belongs to a color. GMM works with "soft assignment" where we say that the data maybe "50% red" and "20% blue "or "30% green".

  This problem has been further explored. I have attached my github link where I have compared the working for both on toy dataset and digits.



```
C:\Users\Gaurav\Desktop\USC\Courses\ML\HW Submission\Assignment-4>py gmmTest.py
GMM for toy dataset with k_means init converged in 8 iteration. Final log-likelihood of data: -1663.2697448261306
GMM for toy dataset with random init converged in 22 iteration. Final log-likelihood of data: -1663.2697448253793
GMM for digits dataset with k_means init converged in 14 iterations. Final log-likelihood of data: 126129.05555281807
GMM for digits dataset with random init converged in 19 iterations. Final log-likelihood of data: 124969.24485098972
```



```
C:\Users\Gaurav\Desktop\USC\Courses\ML\HW Submission\Assignment-4>py kmeansTest.py
[success] : kmeans clustering done on toy dataset
Toy dataset K means clustering converged in 2 steps
RGB centroids computed in 36 iteration
Mean square error per pixel is 0.009735123193995537
Prediction accuracy of K-means classifier with 30 cluster is 0.9177777777777778
Accuracy of logistic regression classifier is 0.9622222222222222
Accuracy of Nearest Neighbour classifier is 0.9933333333333333
```

I have also posted the results in the results folder where you can clearly see the difference.

- Question 4:

  In this case we can try using DBScan clustering algorithm or a two way GMM to estimate the number of clusters from intractable data.
  We can also perform Latent Dirichlet Allocation. I have worked on Guided LDA and it seems to work better than LDA for intractable data. Moreover, if it is text data then we can perform COREX or Anchorred Corex on the same. You can see my work on them over here.
  https://github.com/GB8-24/ELISA-using-COREX---Information-Sciences-Institute/tree/master/Project/Notebooks%20for%20review%205

- Question 5:

  In my program I have shown how Kmeans and random allocation affects the final output. We can also use K-MLE and K-MAP and others methods for initializing. I have not experimented with it but as far as I know Kmeans with maximum log likelihood works well.