CS669: Pattern Recognition
Programming Assignment 2 Version 3

Ganesan S18005
Preethi Srinivasan S18001

Introduction

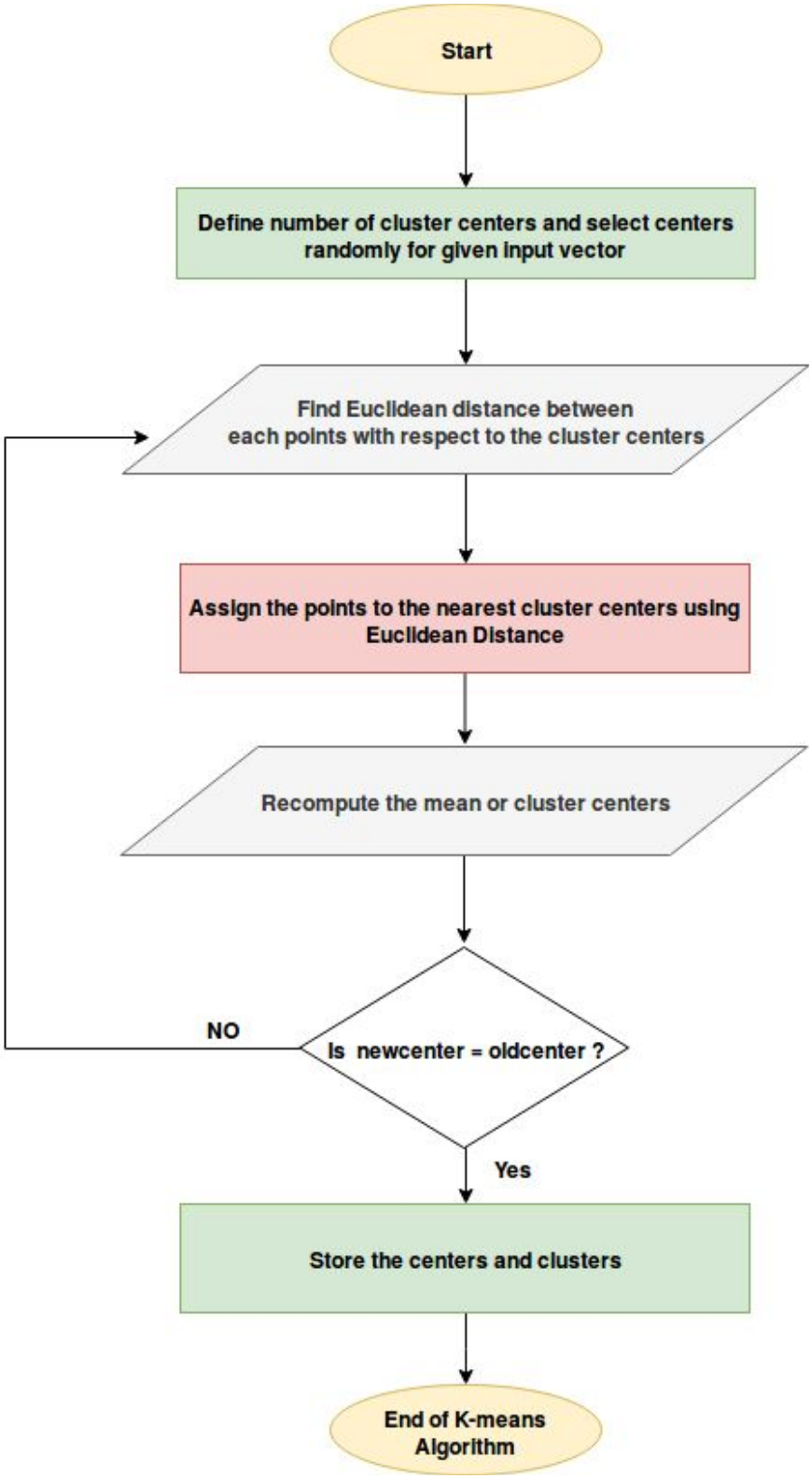
In this assignment, we assume that data from a single class, is coming from multi-modal Gaussian distribution. But we do not know how many modes / clusters are there in the data. We begin by considering the problem of identifying groups or clusters of data points in a multidimensional space. So, we perform the experiment on different number of modes (k's) and compute the accuracy in each case.
After assuming a certain k, we neither know, the membership of point belongs to which cluster and the mean of each cluster. This is a chicken and egg problem. To solve this, K-Means clustering technique is used.

K-Means Clustering Technique:

Given a dataset D = {**x1**, **x2**,,,, **xn}** d dimensional vector, goal is to partition the data set into some number of k clusters. Intuitively, we can think of a cluster as comprising of a group of data points whose inter-point distances are small compared with the distances form the points outside of the class. This is a hard clustering technique. The probability of a data point belonging to a cluster is either 0 or 1. Znk takes binary values = {0, 1}. K-Means is run for different mixtures, K = 1, 2, 4, 8, 16, 32, 64.

Figure 1.1 shows the flowchart of K-Means Clustering Algorithm.

Figure 1.1 **Flow Chart of Algorithm for K-Means:**

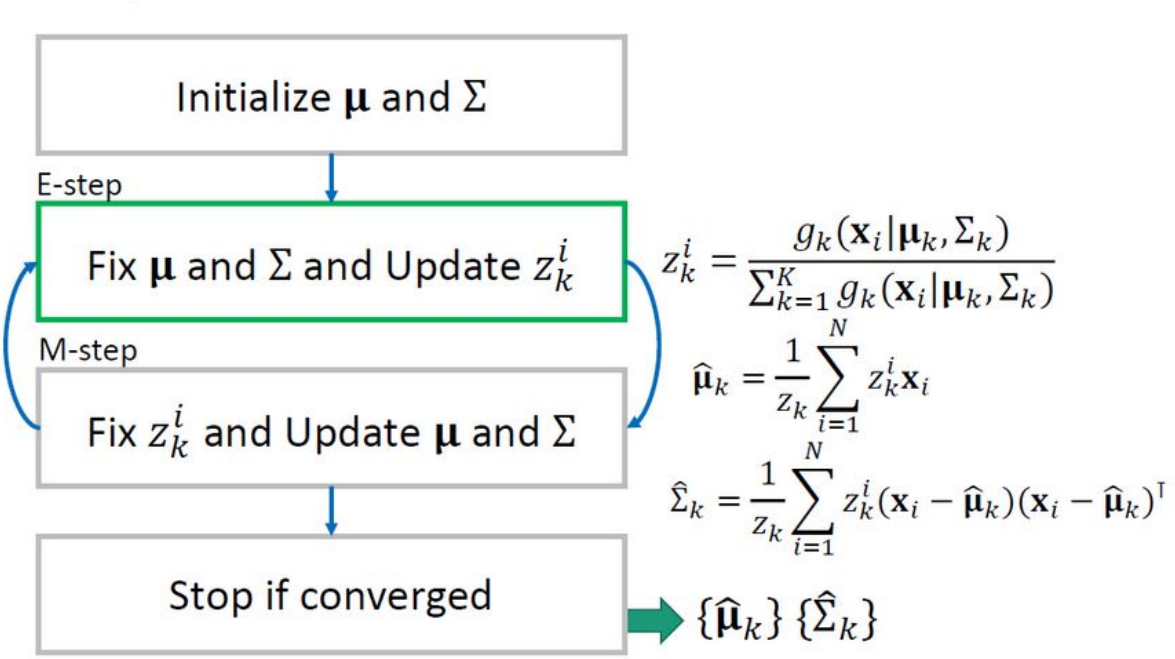


Estimation - Maximization step of Soft Clustering technique called GMM is used. We assume that GMM distribution can be written as a linear combination of Gaussians.

Gmm Clustering Technique:
The centers obtained from K-Means is used to perform GMM.
GMM is a soft clustering technique. Data point belong to a particular cluster with probability gamma.

Figure 1.2 shows the flowchart for GMM Clustering algorithm

Figure 1.2 Flow Chart representing Estimation-Maximization for GMM



2. K-Means and GMM for Non Linearly Separable Data

In this section, we apply the concept of GMM to build the model for the given nonlinearly separable data. We study the performance of the classifier by varying only the number of mixture components (K).

K=1

For k = 1, the parameters are directly computed because data is coming from only 1 cluster. Here, we know the which data point belongs to which cluster. Therefore, we need not run K-Means or GMM in this case.

From the figure 2.2, we can see the decision boundaries between every pair of classes. From Image 1 of the below figure, we can see that decision boundary is linear and behaviour is same as the Bayes Classifier in the previous assignment. This is because the probabilities of both classes are same. In the other three images, decision boundary is non linear because the covariance matrices are assumed to be full.

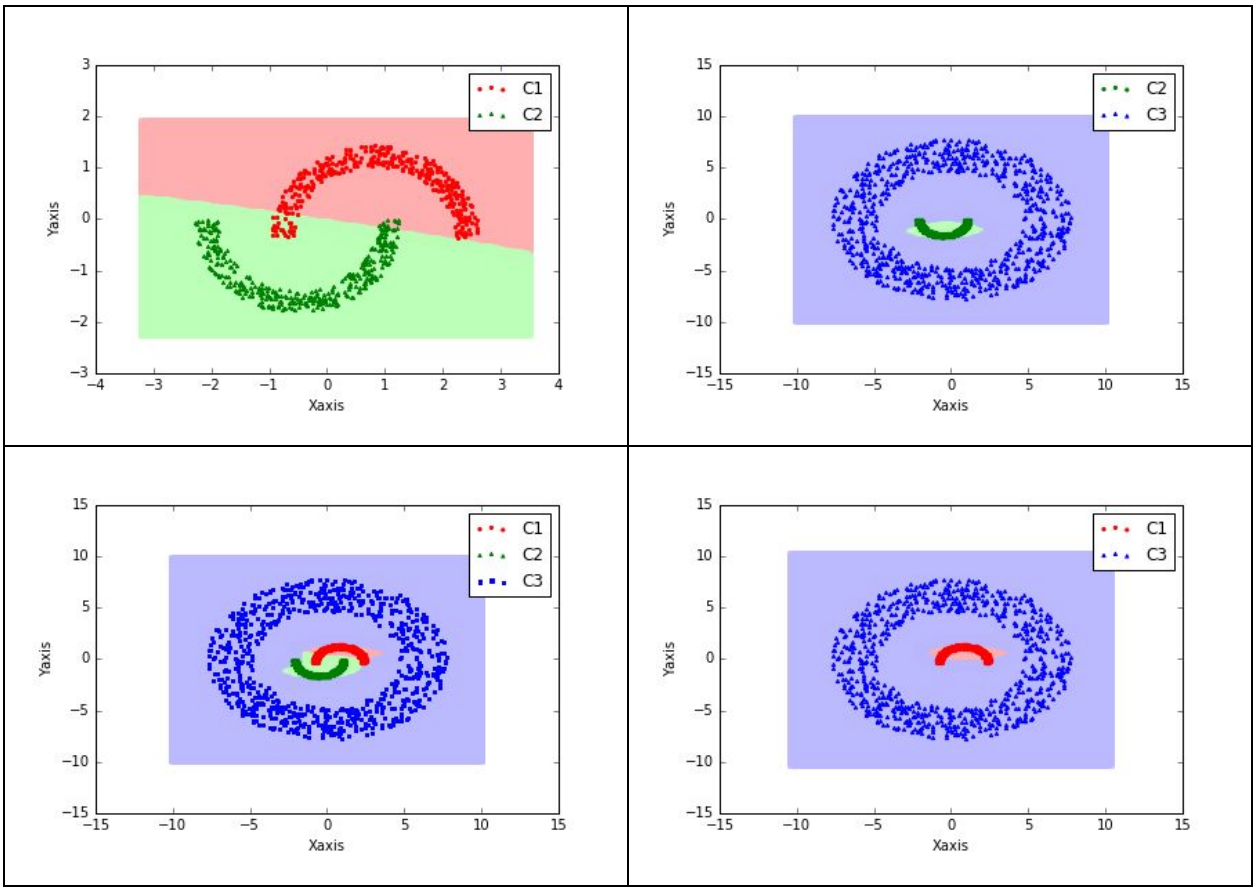


Figure 2.2: Decision Region graph for Non Linearly Separable data, K = 1

From the Figure 2.3a and 2.3b, we can see the Contour plots of the 3 classes. Not much of a difference is observed between both the contours.

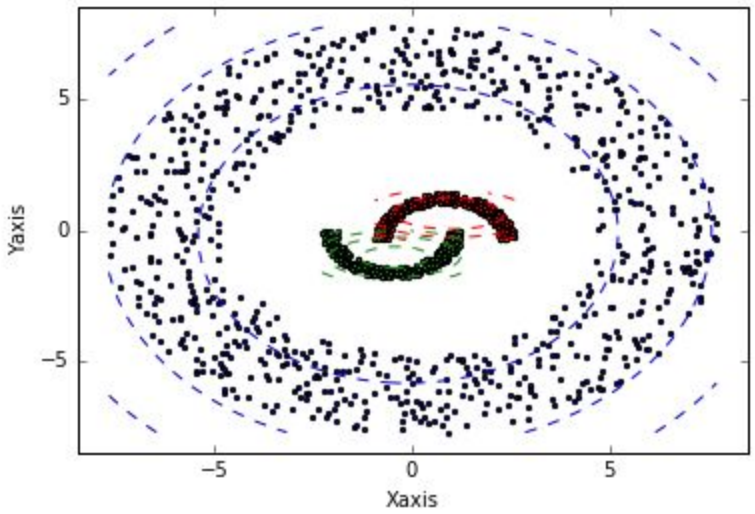


Figure 2.3a: Contour Plot of NLS data after K-Means with K = 1

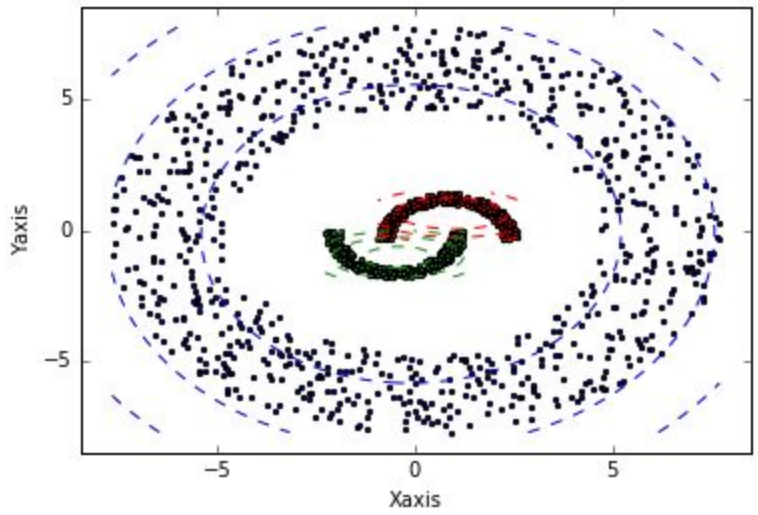


Figure 2.3b: Contour Plot of NLS data after GMM with K = 1

From Table 2.4 and 2.5, we can observe that some of the points have been mis classified and therefore the accuracy is low. This is because we are assuming that entire class is coming from a single cluster.

Table 2.4 The Confusion Matrix for Non Linearly Separable data, K = 1

	Class assigned by the Classifier			
Actual Values		*Class 1*	*Class 2*	*Class 3*
	Class 1	*120*	*5*	*0*
	Class 2	*4*	*121*	*0*
	Class 3	*0*	*0*	*250*

Table 2.5 The Performance Matrix for Non Linearly Separable data, K = 1

	Precision (%)	*Recall Rate (%)*	*F Score (%)*
Class 1	*96.77*	*96.0*	*96.39*
Class 2	*96.03*	*96.8*	*96.41*
Class 3	*100*	*100*	*100*
Mean Value	*97.6*	*97.6*	*97.6*

Classification Accuracy: 98.2

K = 2

Figure 2.6 shows the log likelihood values for all three classes. Class A, Class B and Class C converged after 14, 3 and 12 iterations respectively. This observation has been made after GMM.

With the centers obtained after K-Means, responsibility term is obtained and the responsibility term is used to compute the effective number of points in the class, mu_new, sigma_new and pi_new. Then, the new log likelihood is computed and compared with the old one. When the absolute difference between the two is less than 0.001, we consider state of convergence.

The responsibility term

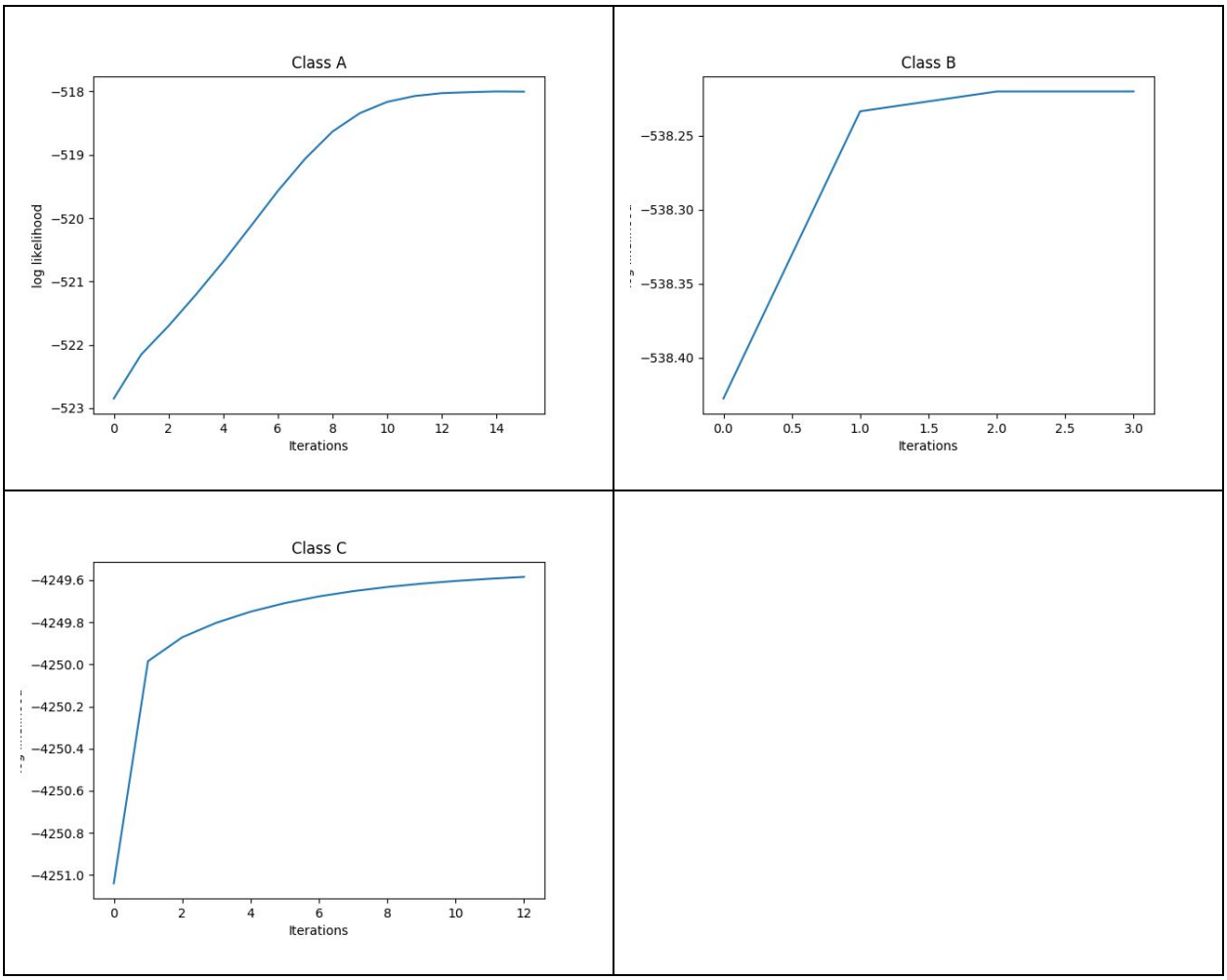


Figure 2.6 The Log Likelihood graph for Non Linearly Separable data, K = 2

From Image 1 of Figure 2.7, we can see that, simply by increasing the number of clusters, that is by assuming that the data of a single is coming from 2 clusters instead of 1, decision is way more sophisticated and all the points are classified correctly.
In case of Image 2, since the probability of Class C is very high, meaning more number of points in Class C, all the surrounding points are classified as Class 3. In case of Class A and Class B, the surrounding points are more or less divided equally as Class A and Class B because the probabilities of each of those classes are same.

Also, the decision boundary between them is nonlinear. This is because the decision boundary depends on the covariance matrices of individual clusters in the class.

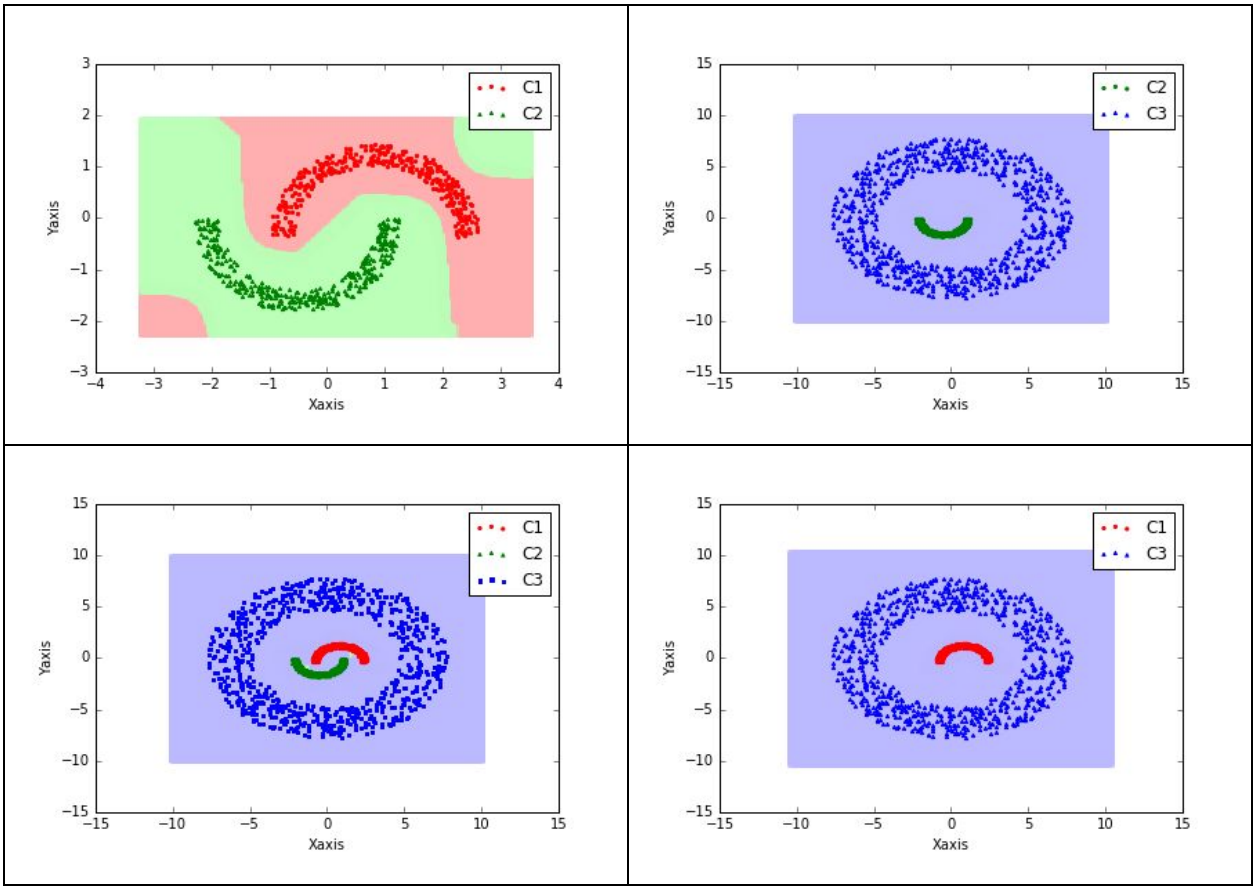


Figure 2.7: Decision Region graph for Non Linearly Separable data, K = 2

From decision boundary itself, it is clear that all points are classified correctly. The tables 2.8 and 2.9 are also reflecting the same.

Table 2.8 The Confusion Matrix for Non Linearly Separable data, K = 2

	Class assigned by the Classifier			
Actual Values		*Class 1*	*Class 2*	*Class 3*
	Class 1	*125*	*0*	*0*
	Class 2	*0*	*125*	*0*
	Class 3	*0*	*0*	*125*

Table 2.9 The Performance Matrix for Non Linearly Separable data, K = 2

	Precision (%)	*Recall Rate (%)*	*F Score (%)*
Class 1	*100.0*	*100.0*	*100.0*
Class 2	*100.0*	*100.0*	*100.0*
Class 3	*100.0*	*100.0*	*100.0*
Mean Value	*100.0*	*100.0*	*100.0*

Classification Accuracy: 100.0

From Figure 2.10a and 2.10b we can see that there are 2 clusters in each class. The contours help us to visualize how the clusters have been assumed.

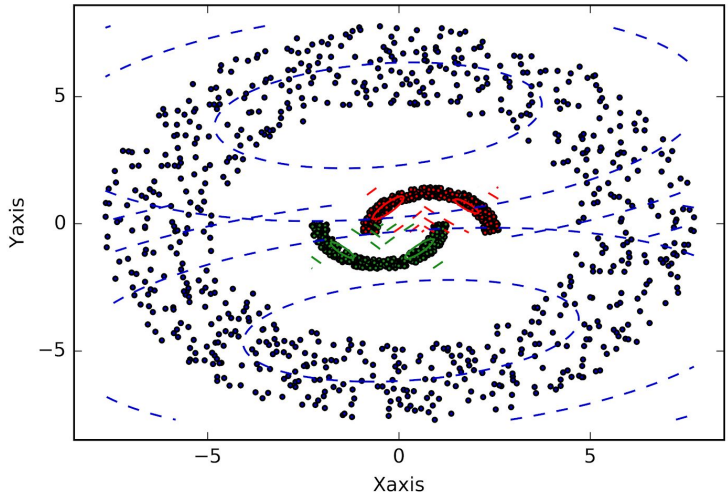


Figure 2.10a: Contour Plot of NLS data after K-Means with K = 2

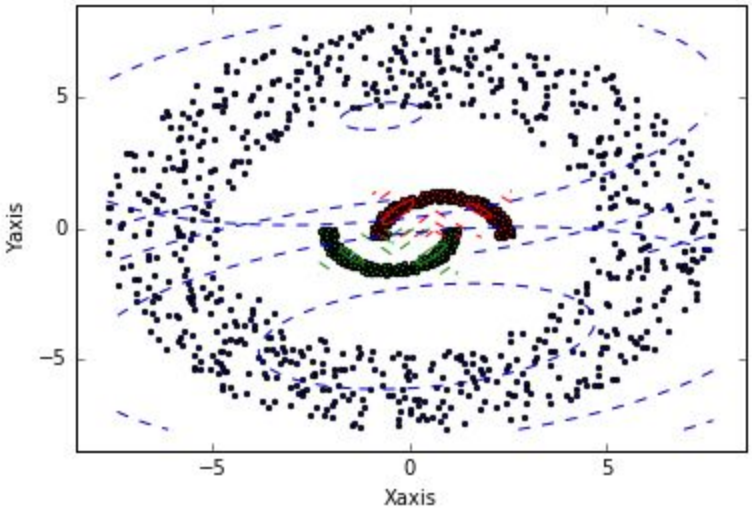


Figure 2.10a: Contour Plot of NLS data after GMM with K = 2

K = 4

Figure 2.11, we can see that log likelihood converged after multiple iterations. The initial centers for applying Estimation step of GMM is taken from the K-Means algorithm.

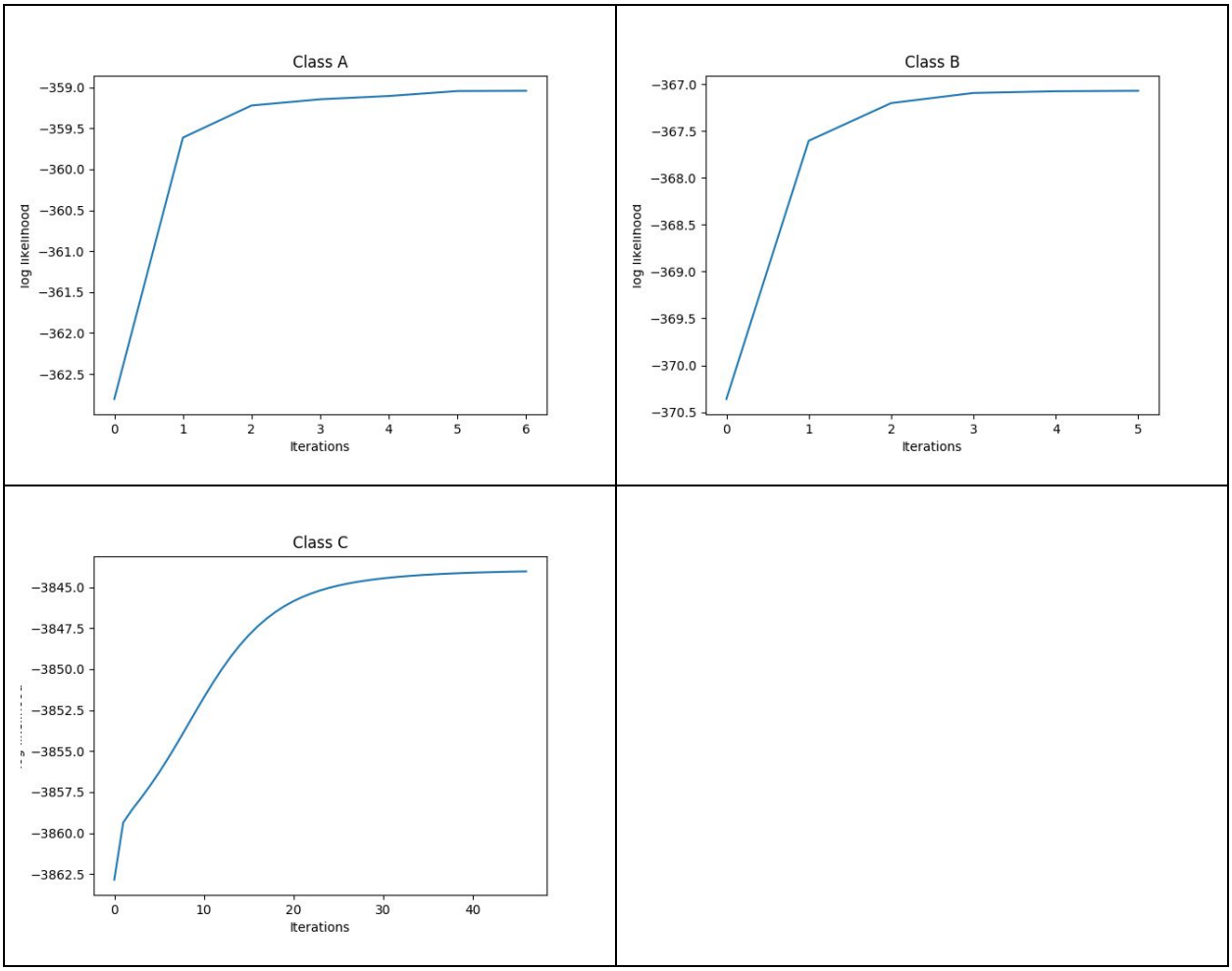


Table 2.11 The Log Likelihood graph for Non Linearly Separable data, K = 4

From Image 1 of Figure 2.12, we can see way more sophisticated decision boundary. Image 3 of Figure 2.12 shows that few points around Class B also classified as Class B. This is because we considered the data is coming 4 clusters and the respective contour plots also reveal that.

Figure 2.13 is shows the decision boundary figure without superimposing the training data points which gives clarity that points around the region of Class A and Class B got classified correctly as Class A and Class B.

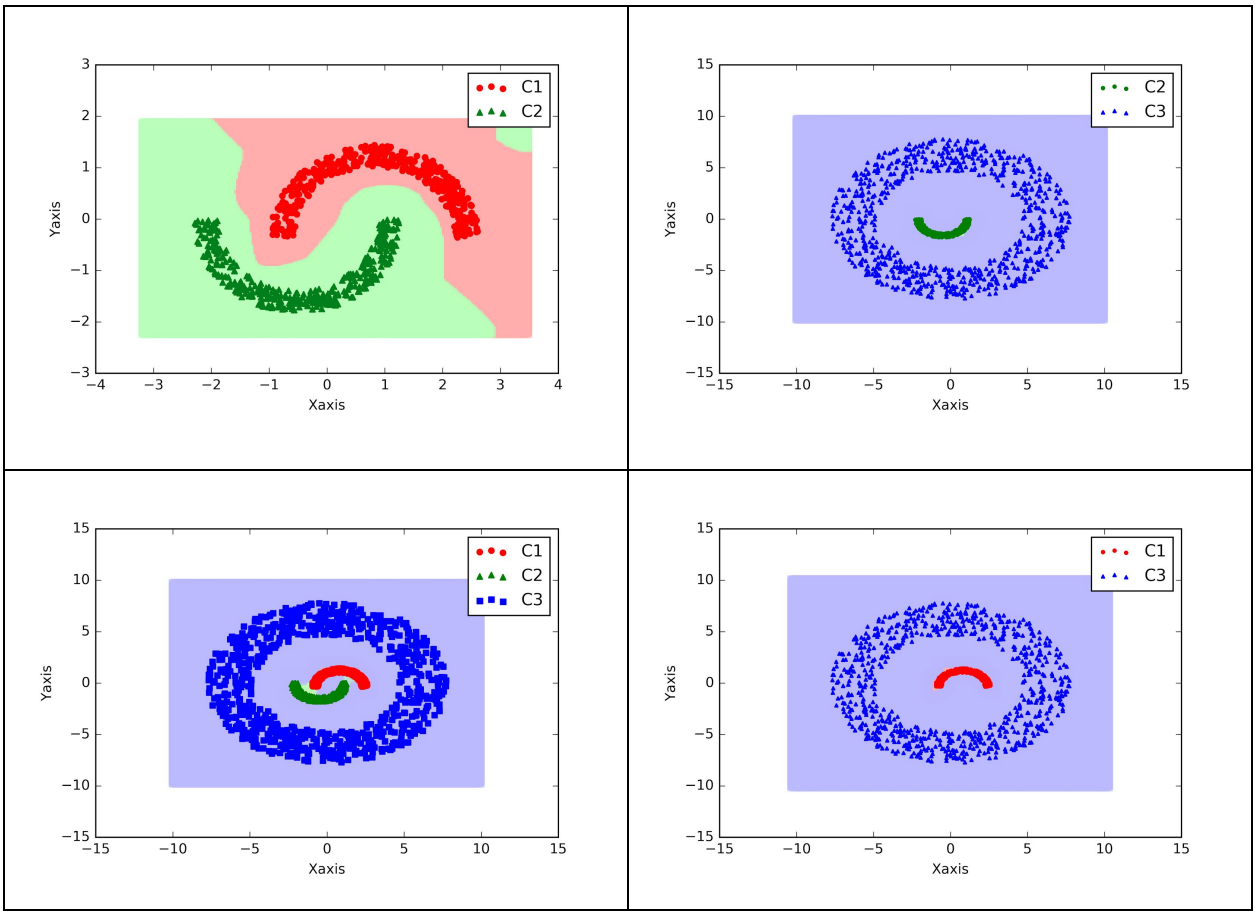


Figure 2.12: Decision Region graph for Non Linearly Separable data, K = 4

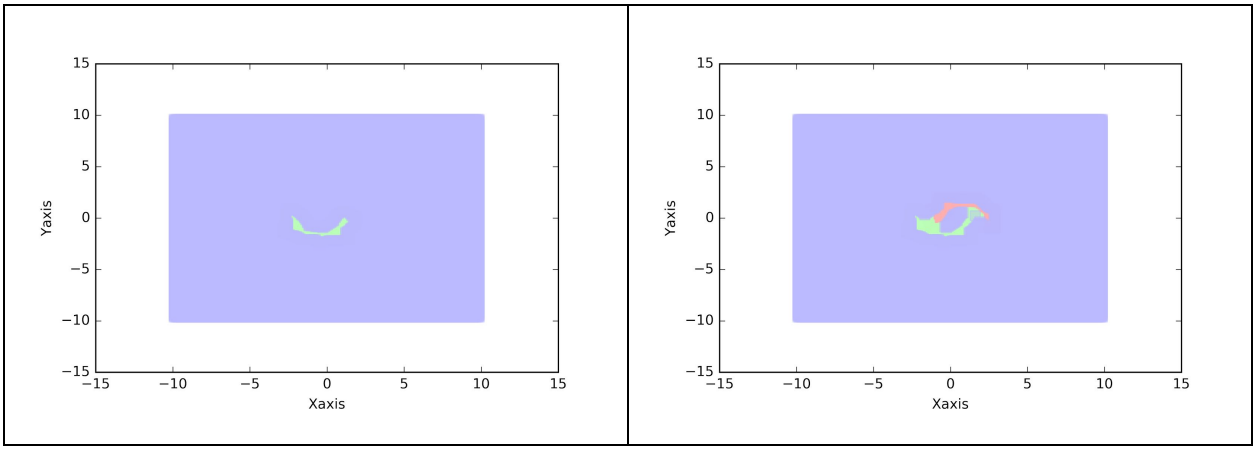


Figure 2.13: Decision Region graph for Non Linearly Separable data before superimposing the training data points, K = 4

Figure 2.14a and 2.14b show the clustering after K-Means and GMM respectively. Since K-Means is hard clustering and GMM is soft clustering, clustering of the data points is expected to happen more efficiently.

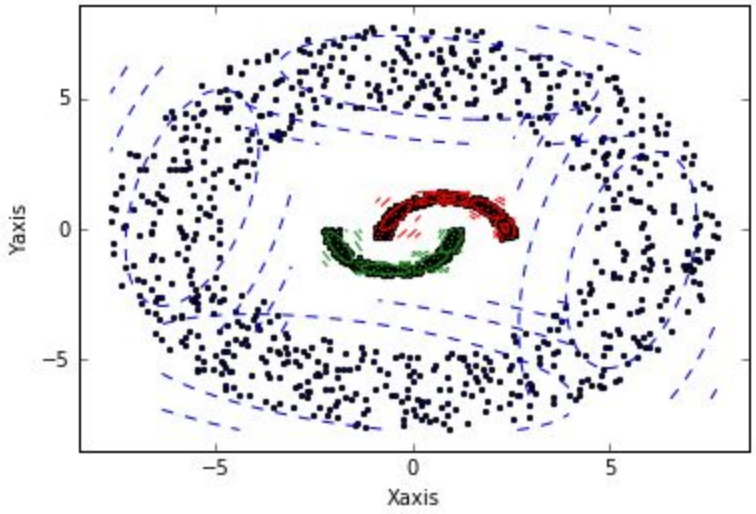


Figure 2.14a: Contour Plot of NLS data after K-Means with K = 4

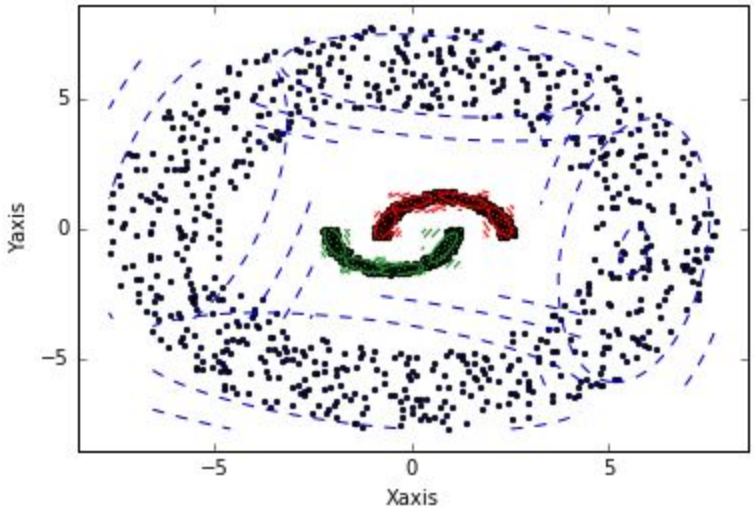


Figure 2.14b: Contour Plot of NLS data after GMM with K = 4

We can again observe that the test points are classified with 100% accuracy.

Table 2.15a The Confusion Matrix for Non Linearly Separable data, K = 4

	Class assigned by the Classifier			
Actual Values		*Class 1*	*Class 2*	*Class 3*
	Class 1	*125*	*0*	*0*
	Class 2	*0*	*125*	*0*
	Class 3	*0*	*0*	*125*

Table 2.15b The Performance Matrix for Non Linearly Separable data, K = 4

	Precision (%)	*Recall Rate (%)*	*F Score (%)*
Class 1	*100.0*	*100.0*	*100.0*
Class 2	*100.0*	*100.0*	*100.0*
Class 3	*100.0*	*100.0*	*100.0*
Mean Value	*100.0*	*100.0*	*100.0*

Classification Accuracy: 100.0%

K = 8

From 2.16, we can see that the number of iterations required to converge is increasing with increasing number of mixtures.

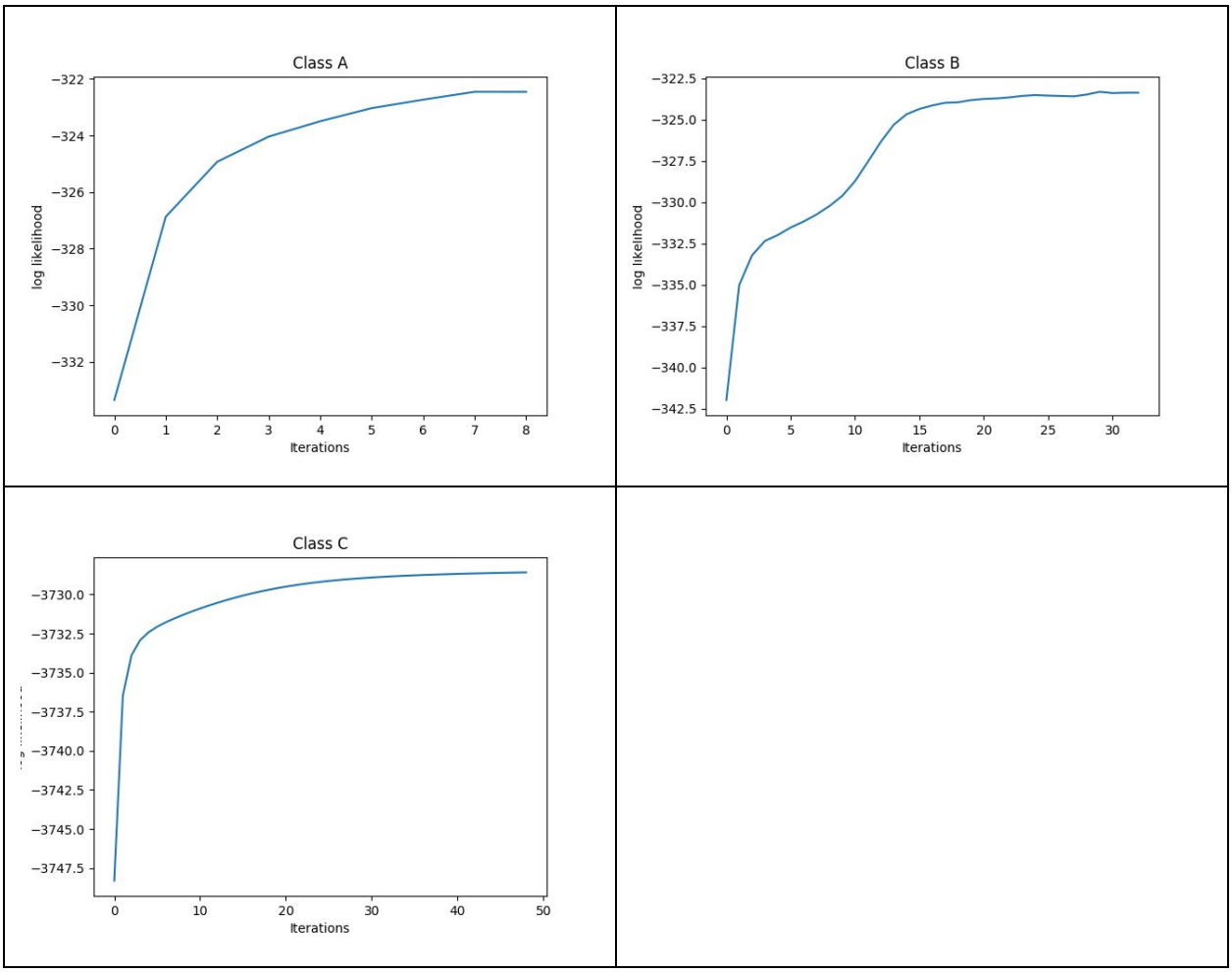


Table 2.16 The Log Likelihood graph for Non Linearly Separable data, K = 8

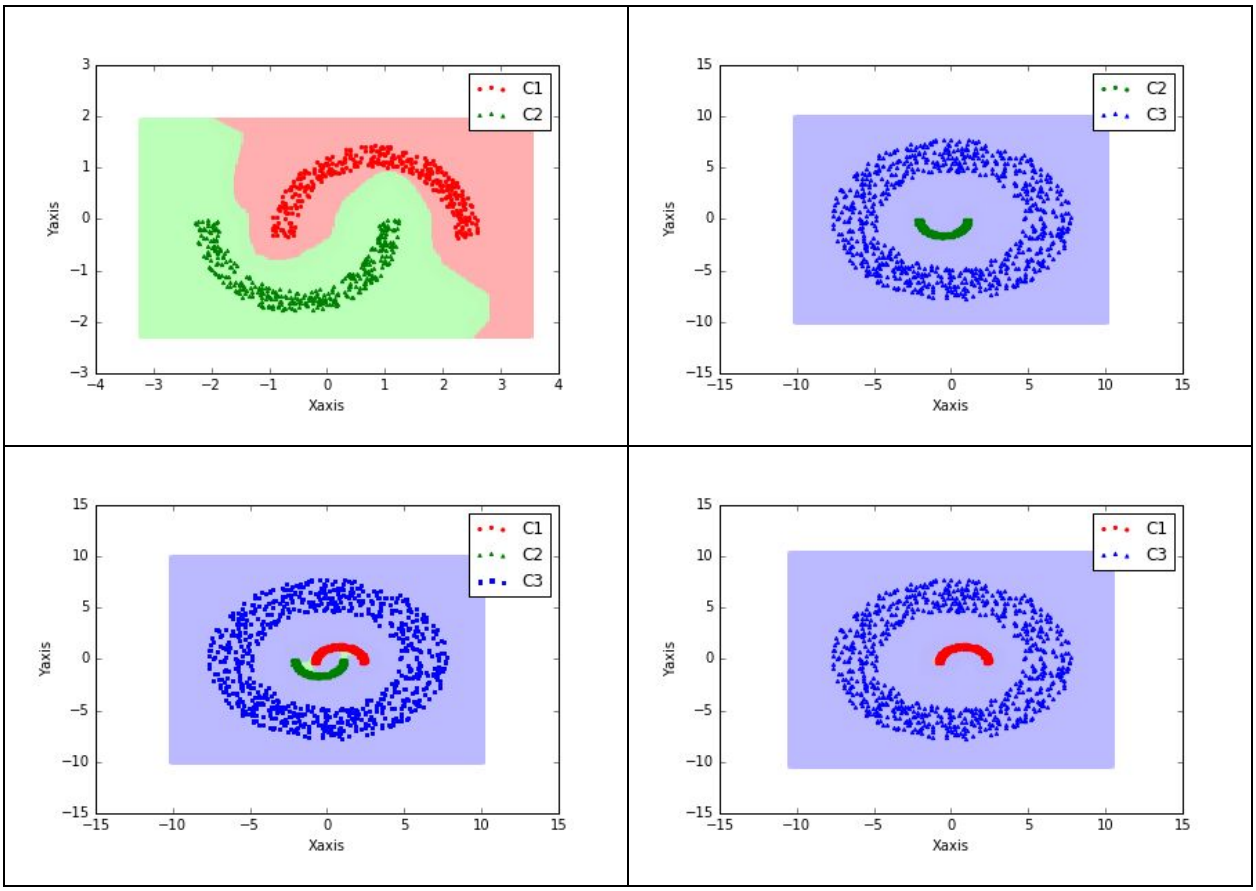


Figure 2.17: Decision Region graph for Non Linearly Separable data, K = 8

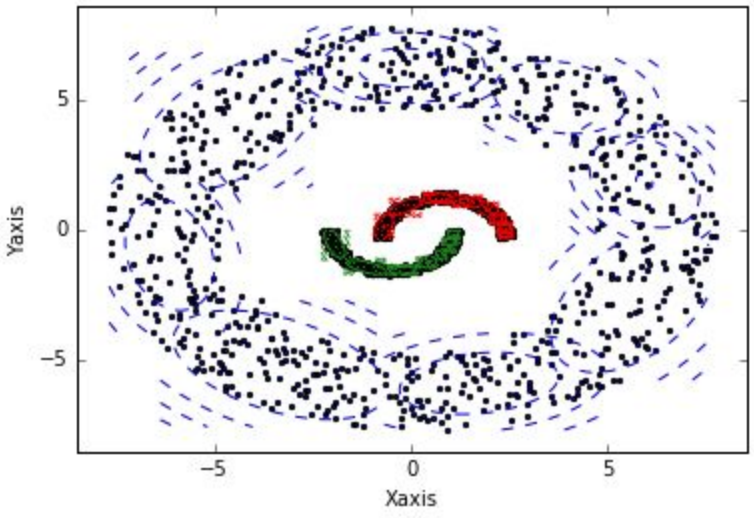


Figure 2.18a: Contour Plot of NLS data after K-Means with K = 8

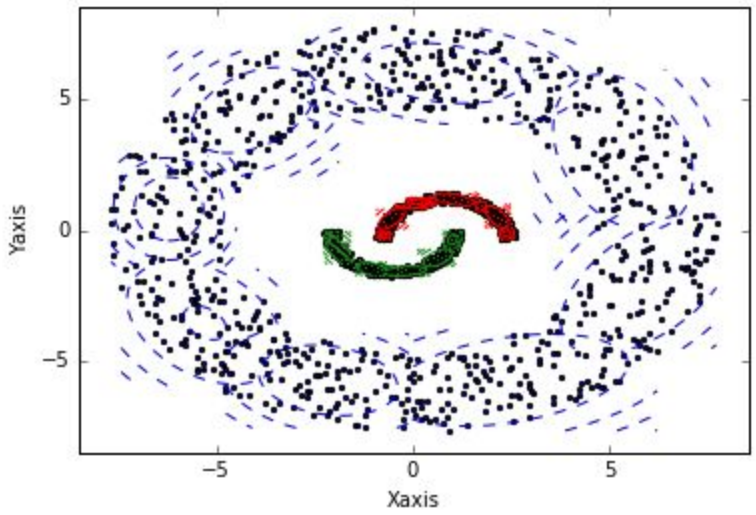


Figure 2.18b: Contour Plot of NLS data after GMM with K = 8

Table 2.19 The Confusion Matrix for Non Linearly Separable data, K = 8

	Class assigned by the Classifier			
Actual Values		*Class 1*	*Class 2*	*Class 3*
	Class 1	*125*	*0*	*0*
	Class 2	*0*	*125*	*0*
	Class 3	*0*	*0*	*125*

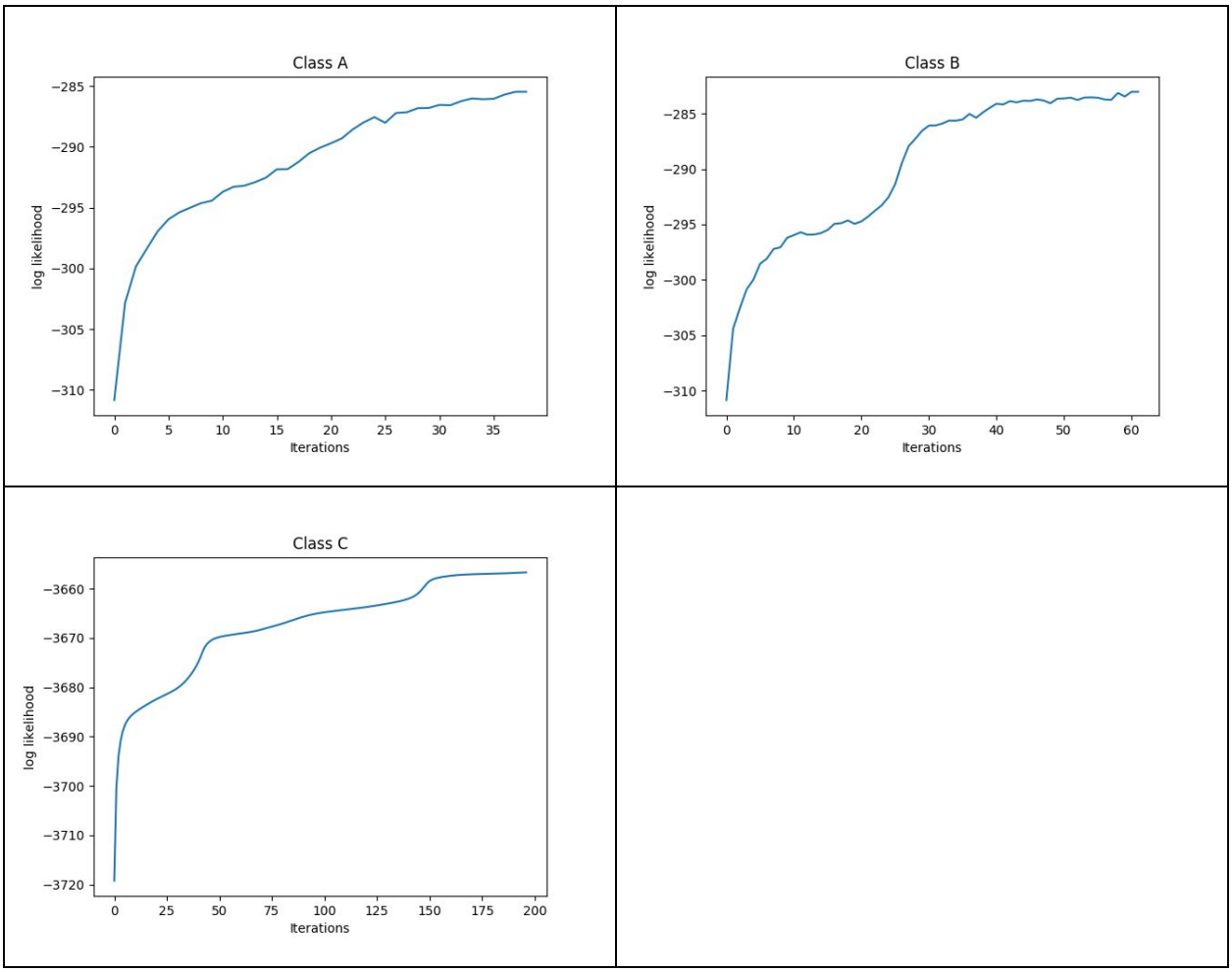
Table 2.19 The Performance Matrix for Non Linearly Separable data, K = 8

	Precision (%)	*Recall Rate (%)*	*F Score (%)*
Class 1	*100.0*	*100.0*	*100.0*
Class 2	*100.0*	*100.0*	*100.0*
Class 3	*100.0*	*100.0*	*100.0*
Mean Value	*100.0*	*100.0*	*100.0*

Classification Accuracy: 100.0

K = 16

From below Figure 2.20, we can see that the number of iterations required to converge is increasing a lot. It is going up and down and this is because some of the points are not belonging to any cluster.



2.20 The Log Likelihood graph for Non Linearly Separable data, K = 16

From Image 3 of Figure 2.17, we can see that more points around Class A and Class B are classified as Class A and Class B.

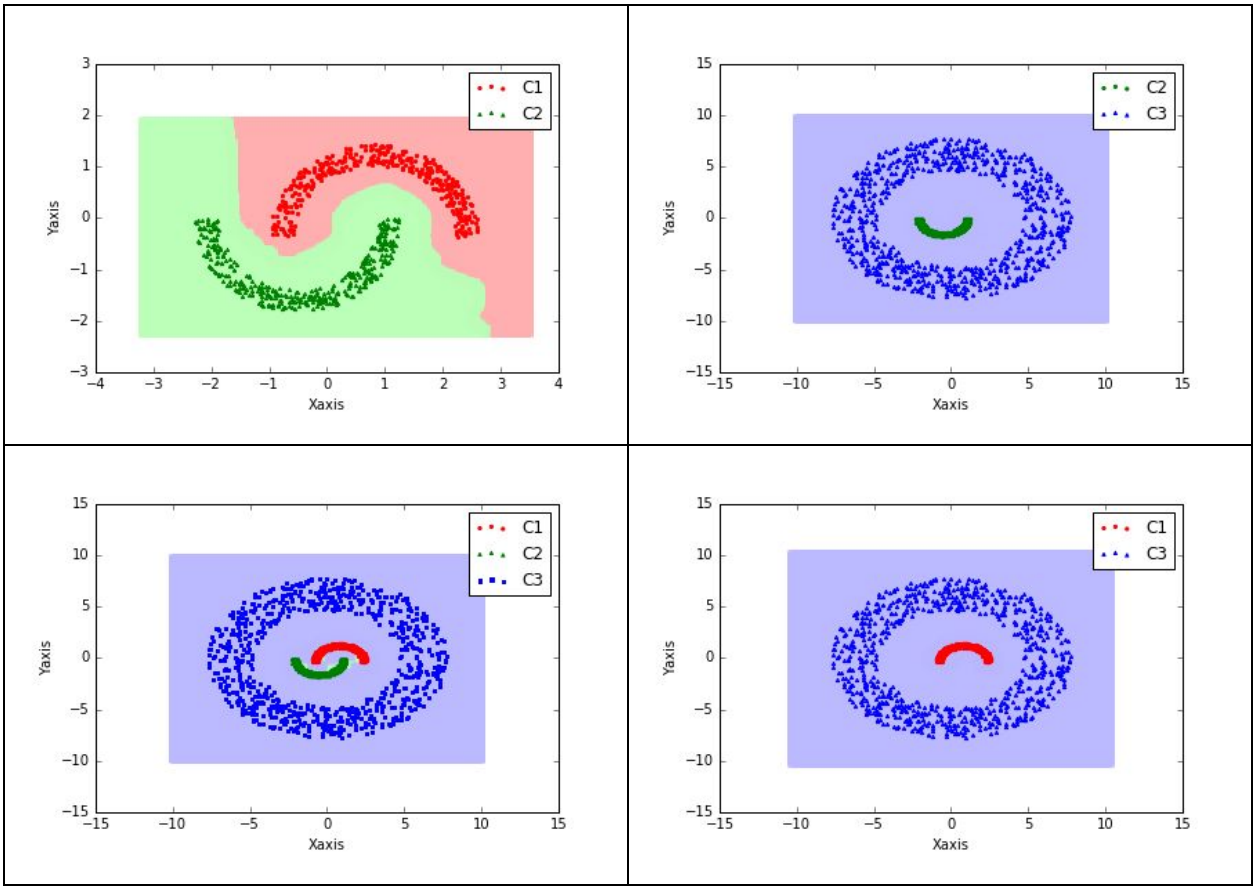


Figure 2.21: Decision Region graph for Non Linearly Separable data, K = 16

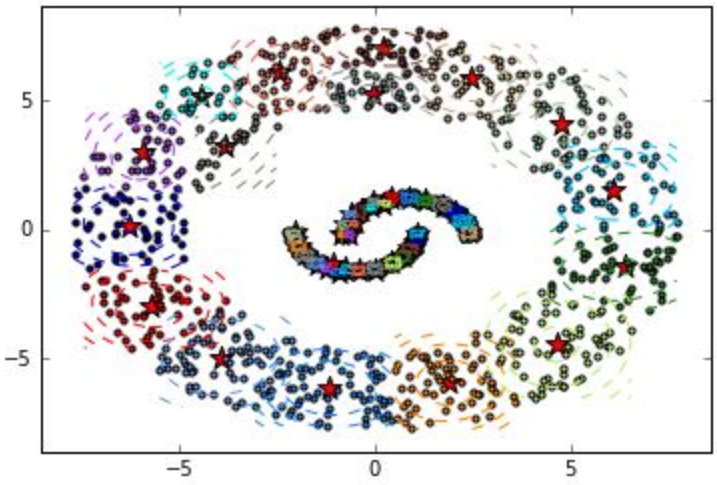


Figure 2.22a: Contour Plot of NLS data after K-Means with K = 16

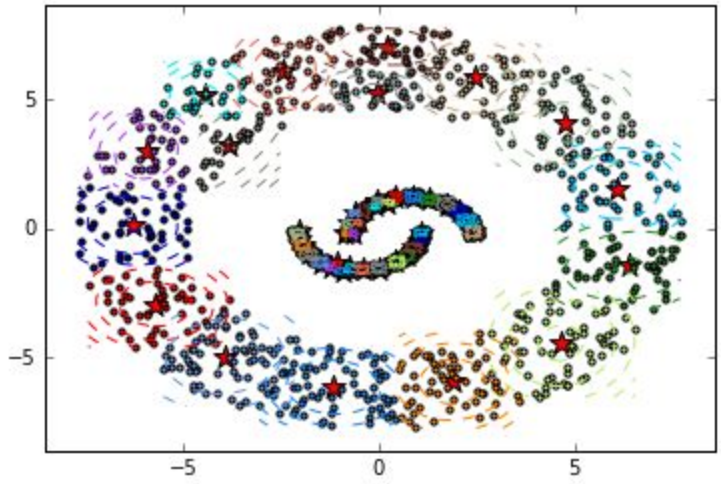


Figure 2.22b: Contour Plot of NLS data after GMM with K = 16

Table 2.23a The Confusion Matrix for Non Linearly Separable data, K = 16

	Class assigned by the Classifier			
Actual Values		*Class 1*	*Class 2*	*Class 3*
	Class 1	*125*	*0*	*0*
	Class 2	*0*	*125*	*0*
	Class 3	*0*	*0*	*125*

Table 2.23b The Performance Matrix for Non Linearly Separable data, K = 16

	Precision (%)	*Recall Rate (%)*	*F Score (%)*
Class 1	*100.0*	*100.0*	*100.0*
Class 2	*100.0*	*100.0*	*100.0*
Class 3	*100.0*	*100.0*	*100.0*
Mean Value	*100.0*	*100.0*	*100.0*

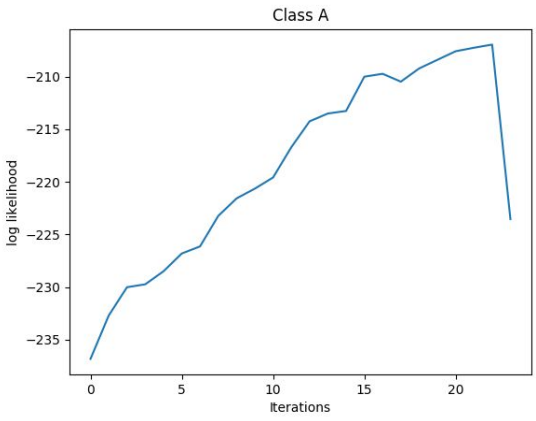
Classification Accuracy: 100.0

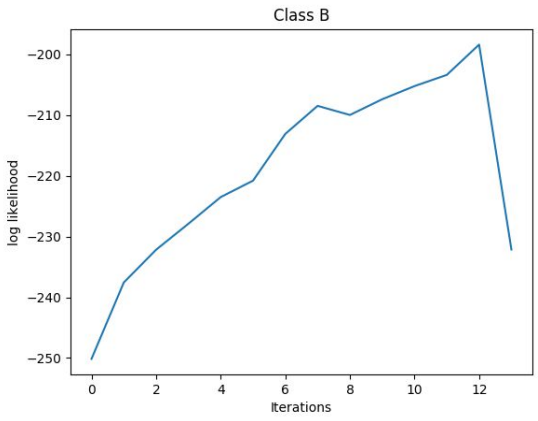
K = 32

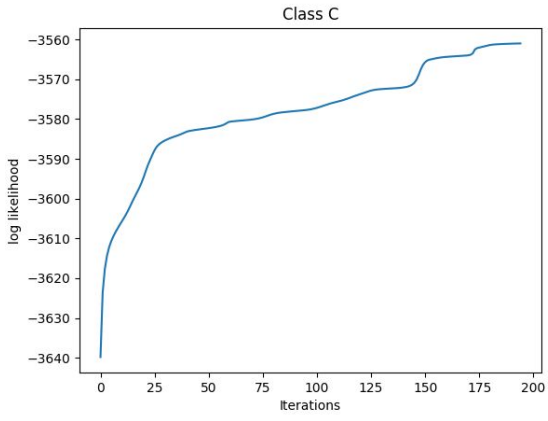
From Figure 2.20, we can see that the log likelihood is decreasing after a certain number of iterations in Class A and Class B. That is because there are lot of points that are not belonging to any of the 32 clusters.

The reason the log likelihood graphs went down after a point is because we used the abs(log_likelihood_new - log_likelihood_old) > 0.01. We did not check for strict criteria of increasing function.

Table 2.24 The Log Likelihood graph for Non Linearly Separable data, K = 32







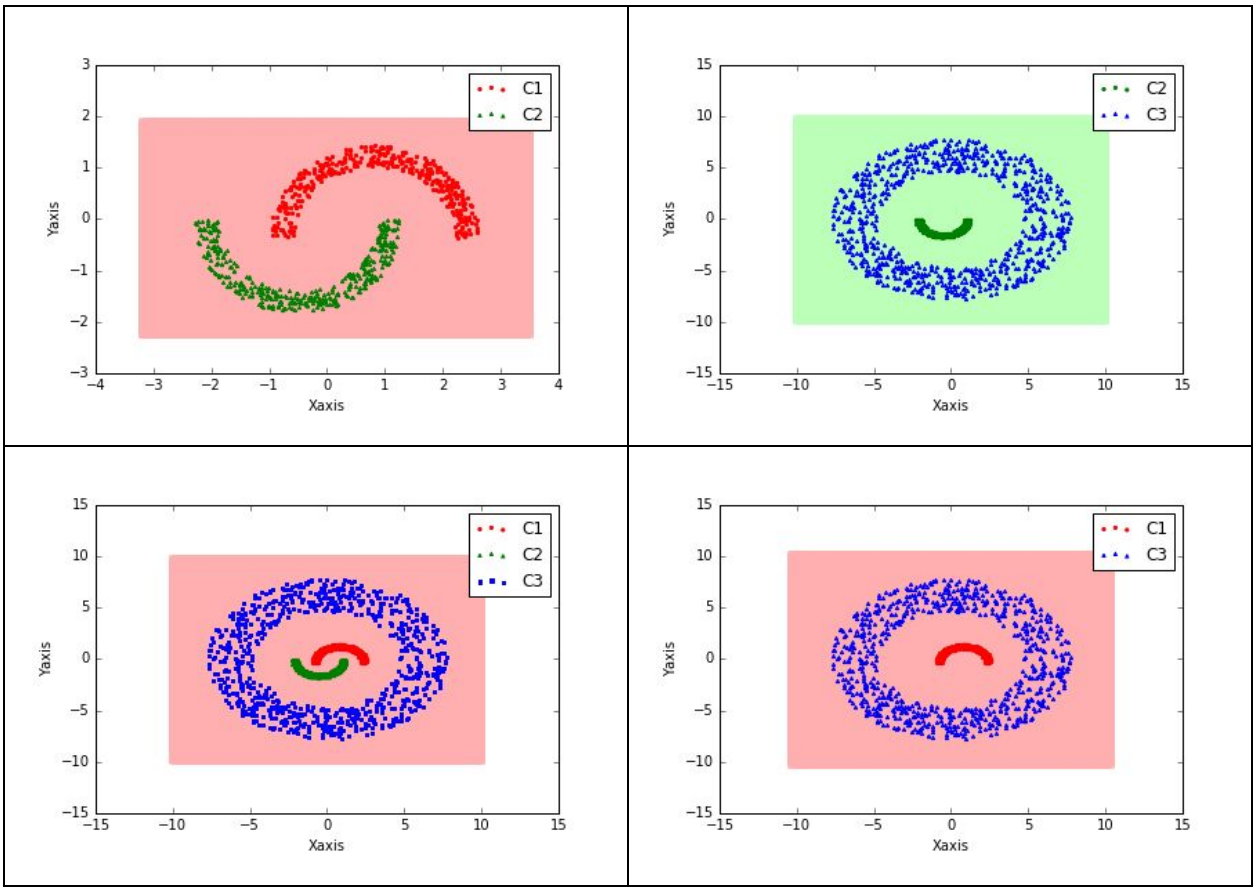


Figure 2.25: Decision Region graph for Non Linearly Separable data, K = 32

In this figure, we can observe that all the points of other classes Class B and Class C are misclassified as Class A, the confusion matrix also shows the results as shown below.

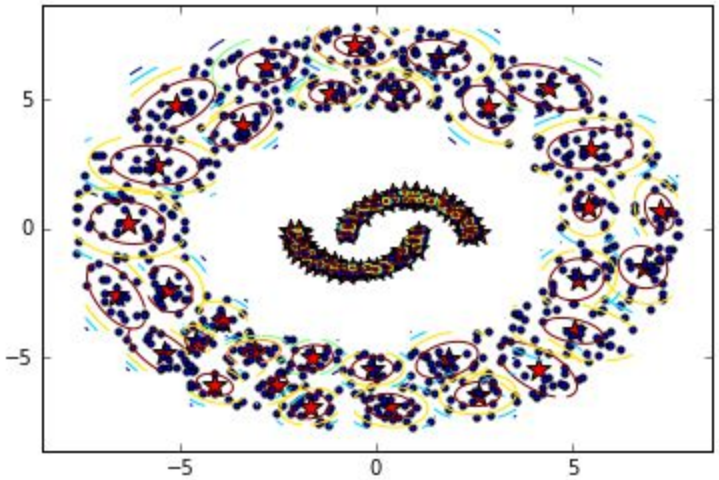


Figure 2.26a: Contour Plot of NLS data after GMM with K = 32

As the number of K increases to 32, the a number of points lie outside the cluster. We can see that from the Contour plot. There are lot of points evident in Class C which do not belong to any of the ellipses.

Table 2.27a The Confusion Matrix for Non Linearly Separable data, K = 32

	Class assigned by the Classifier			
Actual Values		*Class 1*	*Class 2*	*Class 3*
	Class 1	*125*	*0*	*0*
	Class 2	*125*	*0*	*0*
	Class 3	*250*	*0*	*0*

Table 2.27b The Performance Matrix for Non Linearly Separable data, K = 32

	Precision (%)	*Recall Rate (%)*	*F Score (%)*
Class 1	*25*	*100*	*40*
Class 2	*0*	*0*	*0*
Class 3	*0*	*0*	*0*
Mean Value	*8.33*	*33.33*	*13.3*

Classification Accuracy: 25%

K = 64

No data point got associated with any cluster. So, GMM with centers from 64-Means on the data, did not converge.

Real World Data

Now we shall conduct the same experiment on the Speech data represented as 2d points in space. *We repeat the experiment as above with several different mixtures.*

K = 1

For k = 1, the parameters are directly computed because data is coming from only 1 cluster. Here, we know the which data point belongs to which cluster. Therefore, we need not run K-Means or GMM in this case.

From Figure 3.2, we can see that the decision boundaries are very similar to the Bayesian Classifier, Case 4. That is because we are assuming entire class is one cluster and the covariance matrices are as it is without any modification. These are same constraints imposed on the Real World data for Bayesian Classifier in Case 4.

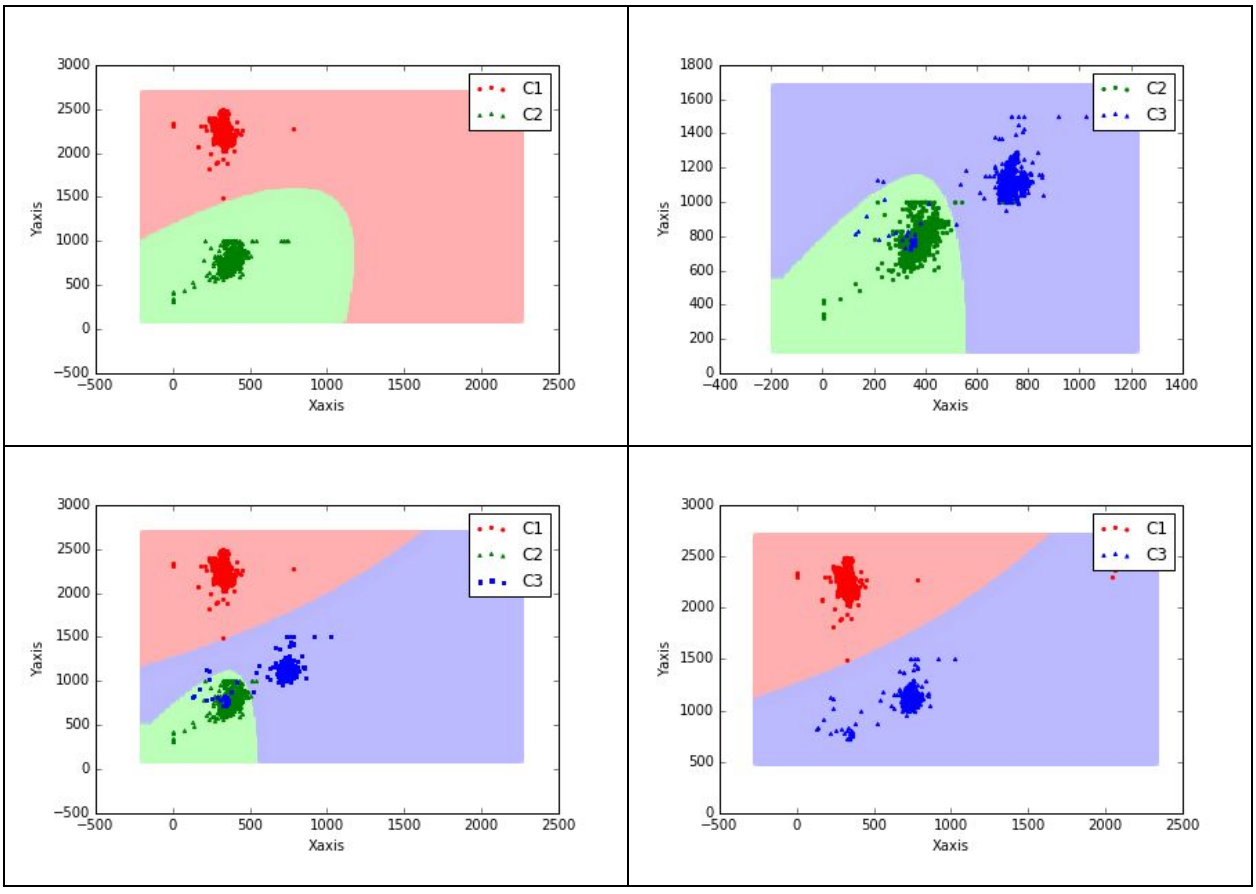


Figure 3.2: Decision Region graph for Real World data, K = 1

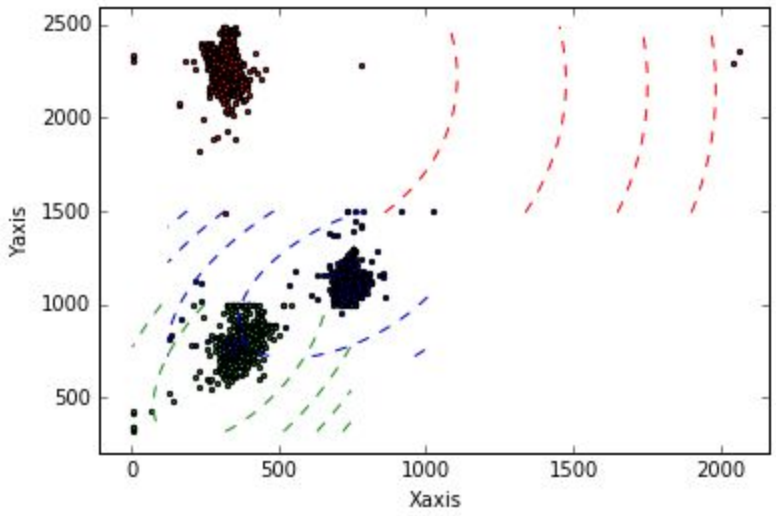


Figure 3.3a: Contour Plot of RD data after K-Means with K = 1

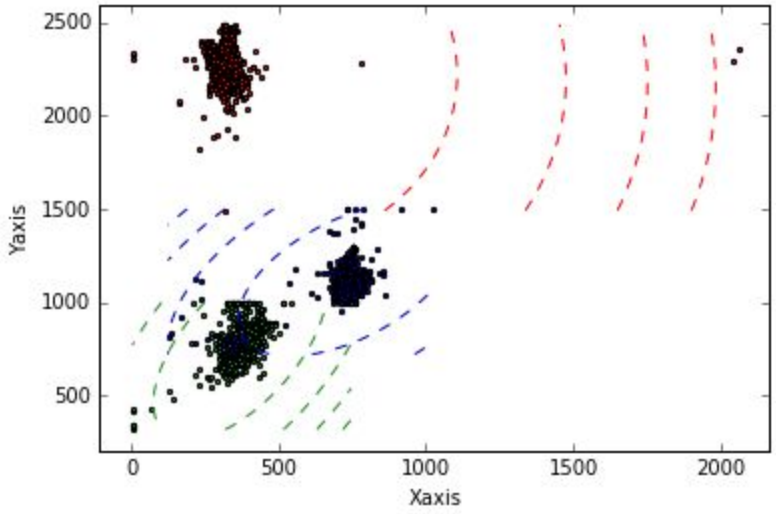


Figure 3.3b: Contour Plot of RD data after GMM with K = 1

From the above contour plots, we can see that the major and minor axis of the ellipse is not aligned with the coordinate axis because the covariance matrices are full.

Table 3.4a The Confusion Matrix for Real World data, K = 1

	Class assigned by the Classifier			
Actual Values		*Class 1*	*Class 2*	*Class 3*
	Class 1	*581*	*3*	*12*
	Class 2	*0*	*611*	*11*
	Class 3	*1*	*28*	*512*

Table 3.4b The Performance Matrix for Real World data, K = 1

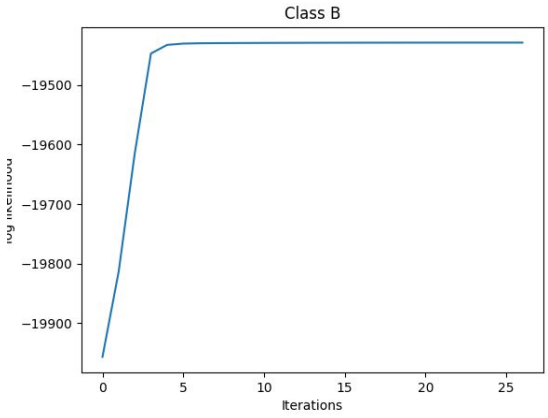
	Precision (%)	*Recall Rate (%)*	*F Score (%)*
Class 1	*99.83*	*97.48*	*98.64*
Class 2	*95.17*	*98.23*	*96.68*
Class 3	*95.7*	*94.64*	*95.17*
Mean Value	*96.9*	*96.78*	*96.83*

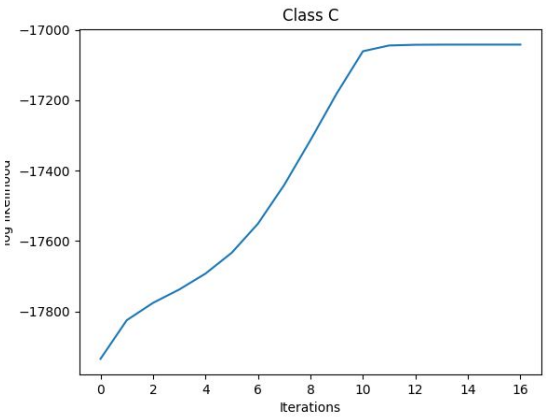
Classification Accuracy: 96.87%

From Table 3.3a and 3.3b, we can see that the accuracy is low and the decision boundary also shows that some points are mis classified.

K = 2

Table 3.5 The Log Likelihood graph for Real World data, K = 2





From the above figures, we can see that the log likelihood converged after more iterations than the earlier case.

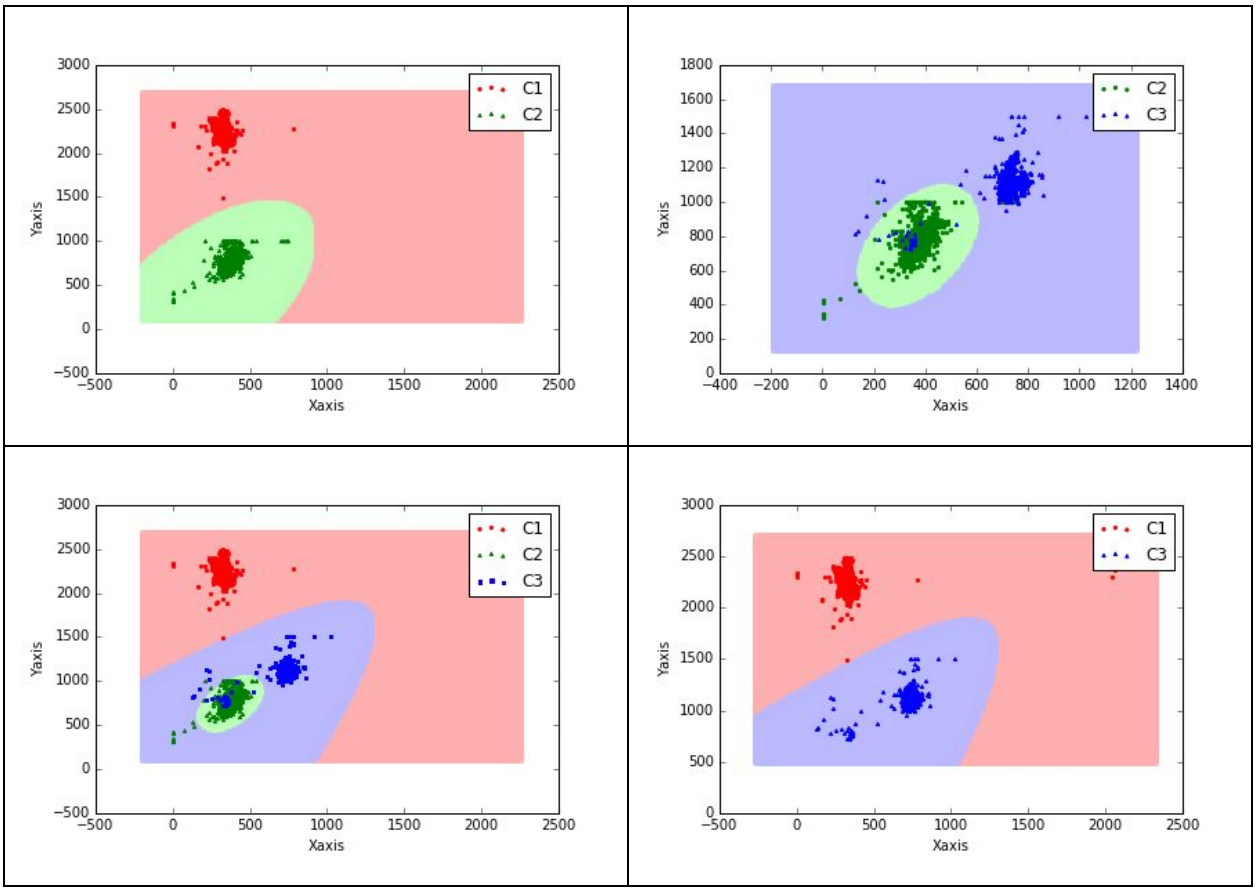


Figure 3.6: Decision Region graph for Real World data, K = 2

From the above decision boundary plots, we can see that more points are classified corrected by clustering into 2 mixtures. The accuracy also as is evident from the performance matrix.
Points around the cluster are also clustered correctly.

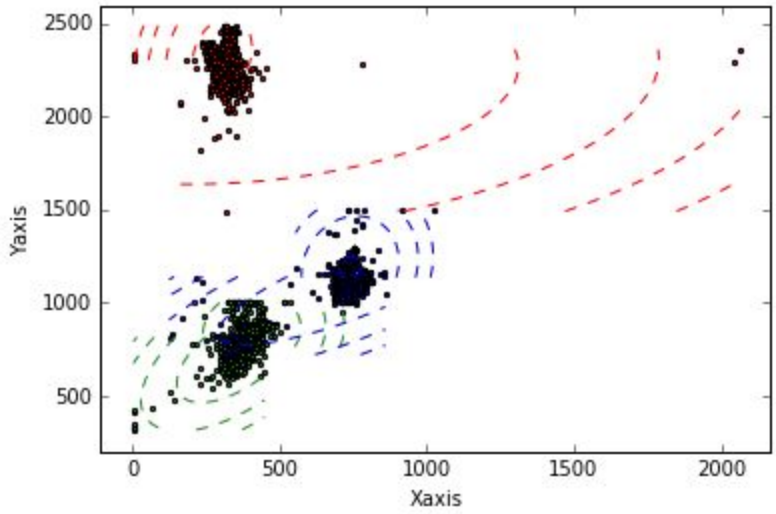


Figure 3.7a: Contour Plot of RD data after K-Means with K = 2

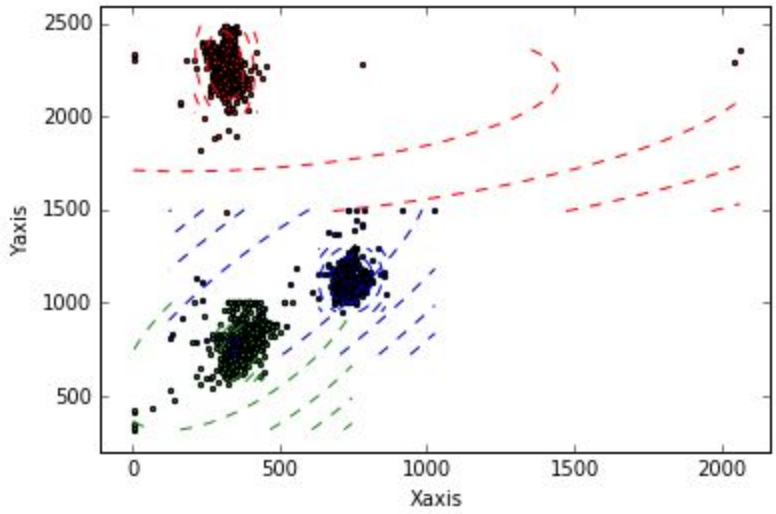


Figure 3.7b: Contour Plot of RD data after GMM with K = 2

Table 3.8a The Confusion Matrix for Real World data, K = 2

	Class assigned by the Classifier			
Actual Values		*Class 1*	*Class 2*	*Class 3*
	Class 1	*592*	*2*	*2*
	Class 2	*0*	*599*	*23*
	Class 3	*2*	*19*	*520*

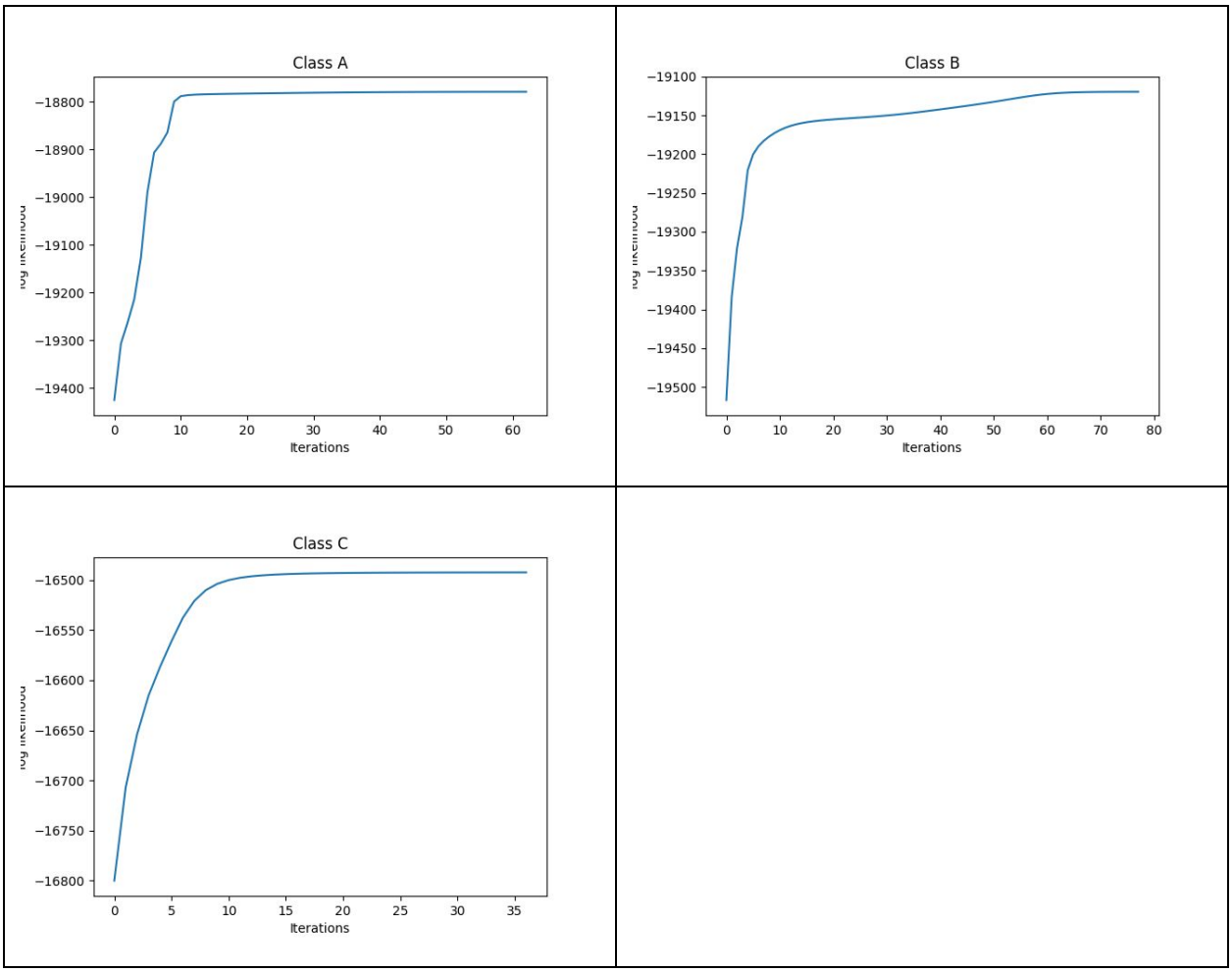
Table 3.8b The Performance Matrix for Real World data, K = 2

	Precision (%)	*Recall Rate (%)*	*F Score (%)*
Class 1	*99.66*	*99.33*	*99.5*
Class 2	*99.61*	*96.3*	*96.46*
Class 3	*95.41*	*96.12*	*95.76*
Mean Value	*97.23*	*97.25*	*97.24*

Classification Accuracy: 97.27%

K = 4

Table 3.9 The Log Likelihood graph for Real World data, K = 4



Although the log likelihood converged, with k = 4, many points were mis-classified as Class A.

Figure 3.10: Decision Region graph for Real World data, K = 4

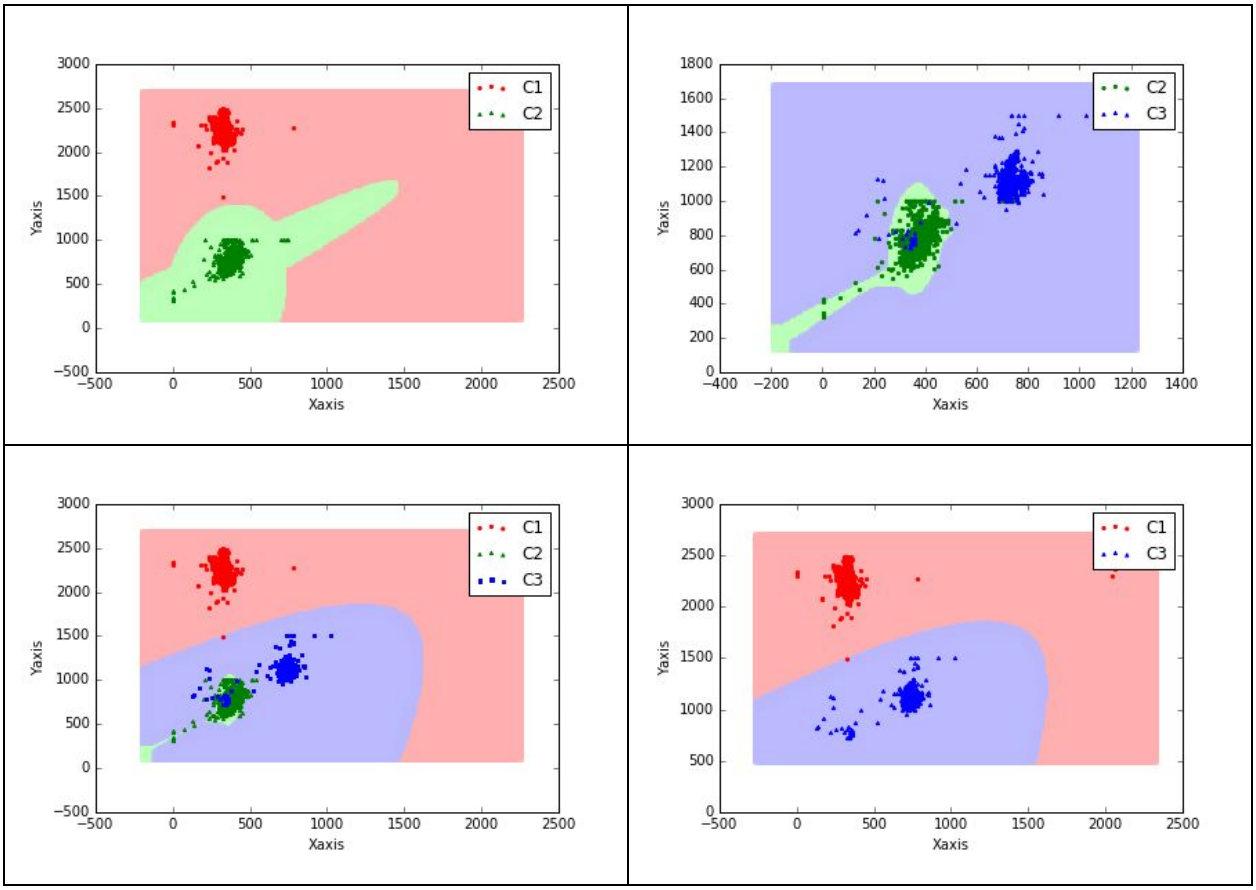


Figure 3.11a: Contour Plot of RD data after K-Means with K = 4

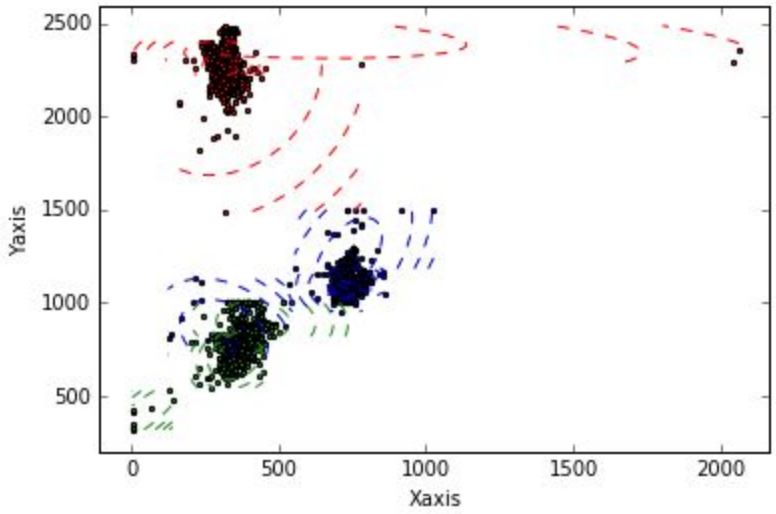
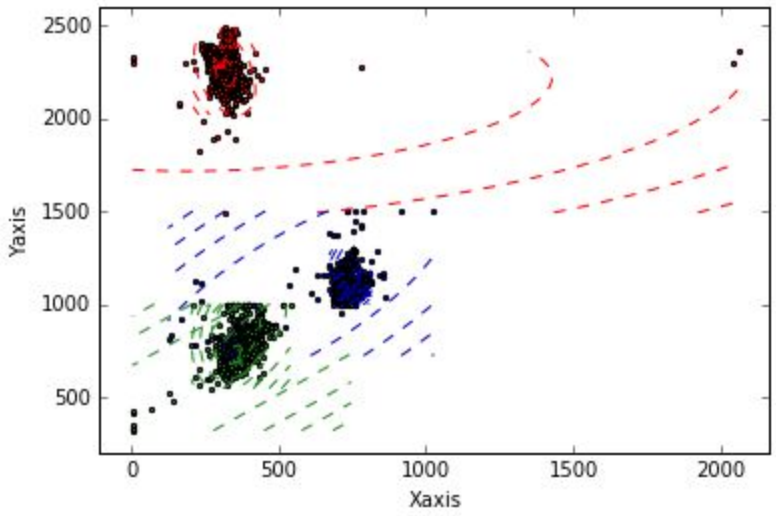


Figure 3.11b: Contour Plot of RD data after GMM with K = 4



We can see that all points are mis classified as Class A. As the number of mixtures increases, accuracy decreases drastically.

Table 3.12 The Confusion Matrix for Real World data, K = 4

	Class assigned by the Classifier			
Actual Values		*Class 1*	*Class 2*	*Class 3*
	Class 1	*596*	*0*	*0*
	Class 2	*642*	*0*	*0*
	Class 3	*591*	*0*	*0*

Table 3.12 The Performance Matrix for Real World data, K = 4

	Precision (%)	*Recall Rate (%)*	*F Score (%)*
Class 1	*32.5*	*100.0*	*49.0*
Class 2	*0*	*0*	*0*
Class 3	*0*	*0*	*0*
Mean Value	*10.83*	*33.33*	*16.33*

Classification Accuracy: 32.5%

K = 8

In Class A, no point is belonging to any cluster, so probability of a data point belonging to a class which is a linear combination of gaussians is resulting in 0. So GMM is not converging. Upon trying with **K = 5** also, the same characteristics are observed.

4. Scene Image Data

K = 1

In this case, we considered, 32*32 patches of each image in each class, made a 24d colored histogram out of it representing the intensities of every pixel. Combined all of the feature vectors and trained the GMM after K-Means.

The covariance matrices are considered to be diagonal to avoid the curse of dimensionality problem.

For k = 1, the parameters are directly computed because data is coming from only 1 cluster. Here, we know the which data point belongs to which cluster. Therefore, we need not run K-Means or GMM in this case.

Table 4.2a The Confusion Matrix for Scene Image data, K = 1

	Class assigned by the Classifier			
Actual Values		*Coast*	*Industrial Area*	*Pagoda*
	Coast	*26*	*16*	*7*
	Industrial Area	*12*	*24*	*14*
	Pagoda	*20*	*3*	*27*

Table 4.2b The Performance Matrix for Scene Image data, K = 1

	Precision (%)	*Recall Rate (%)*	*F Score (%)*
Coast	*44.83*	*53.06*	*48.6*
Industrial Area	*55.8*	*48.0*	*51.61*
Pagoda	*56.25*	*54.0*	*55.1*

Mean Value	*52.3*	*51.69*	*51.77*

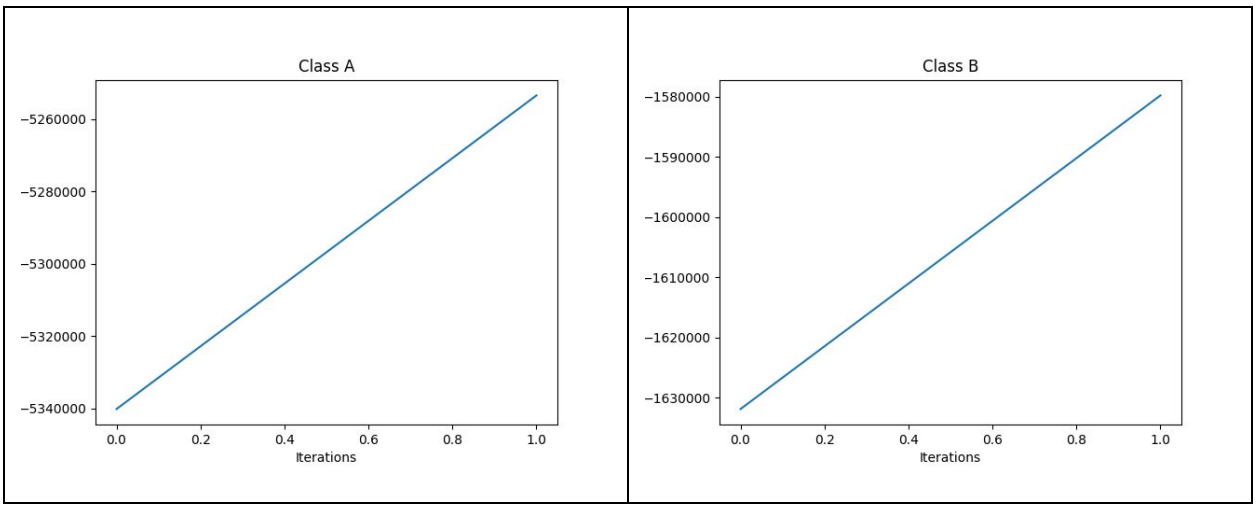
Classification Accuracy: 51.6%

But as the number of clusters increased, accuracy reduced even lower.
This can attributed to the fact that the color histogram of intensities are good enough for classification.

K = 2

From Figure 4.2, we can see that the likelihood graph does not converge. It becomes -inf and NAN after couple of iterations itself. This is because the points are not belonging to any cluster which turns the det of covariance matrix 0.

Here the log likelihood was expected to increase in every iteration. That is why we see a strictly increasing function.



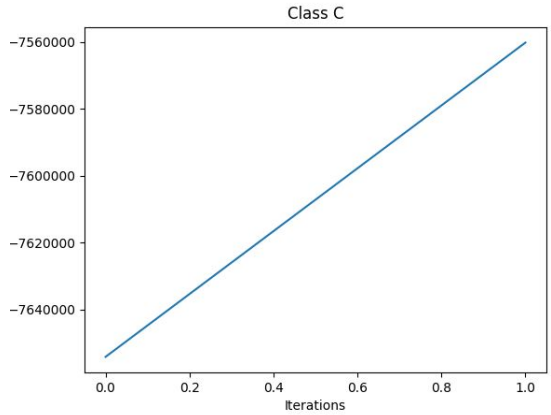


Figure 4.3 The Log Likelihood graph for Real World data, K = 2

Table 4.4a The Confusion Matrix for Scene Image data, K = 2

	Class assigned by the Classifier			
Actual Values		*Coast*	*Industrial Area*	*Pagoda*
	Coast	*29*	*14*	*6*
	Industrial Area	*16*	*24*	*10*
	Pagoda	*16*	*7*	*27*

Table 4.4b The Performance Matrix for Scene Image data, K = 2

	Precision (%)	*Recall Rate (%)*	*F Score (%)*
Coast	*47.54*	*59.18*	*52.73*
Industrial Area	*53.33*	*48.0*	*50.53*
Pagoda	*62.79*	*54.0*	*58.06*

Mean Value	*54.56*	*53.73*	*53.77*

Classification Accuracy: 53.69%

K = 4

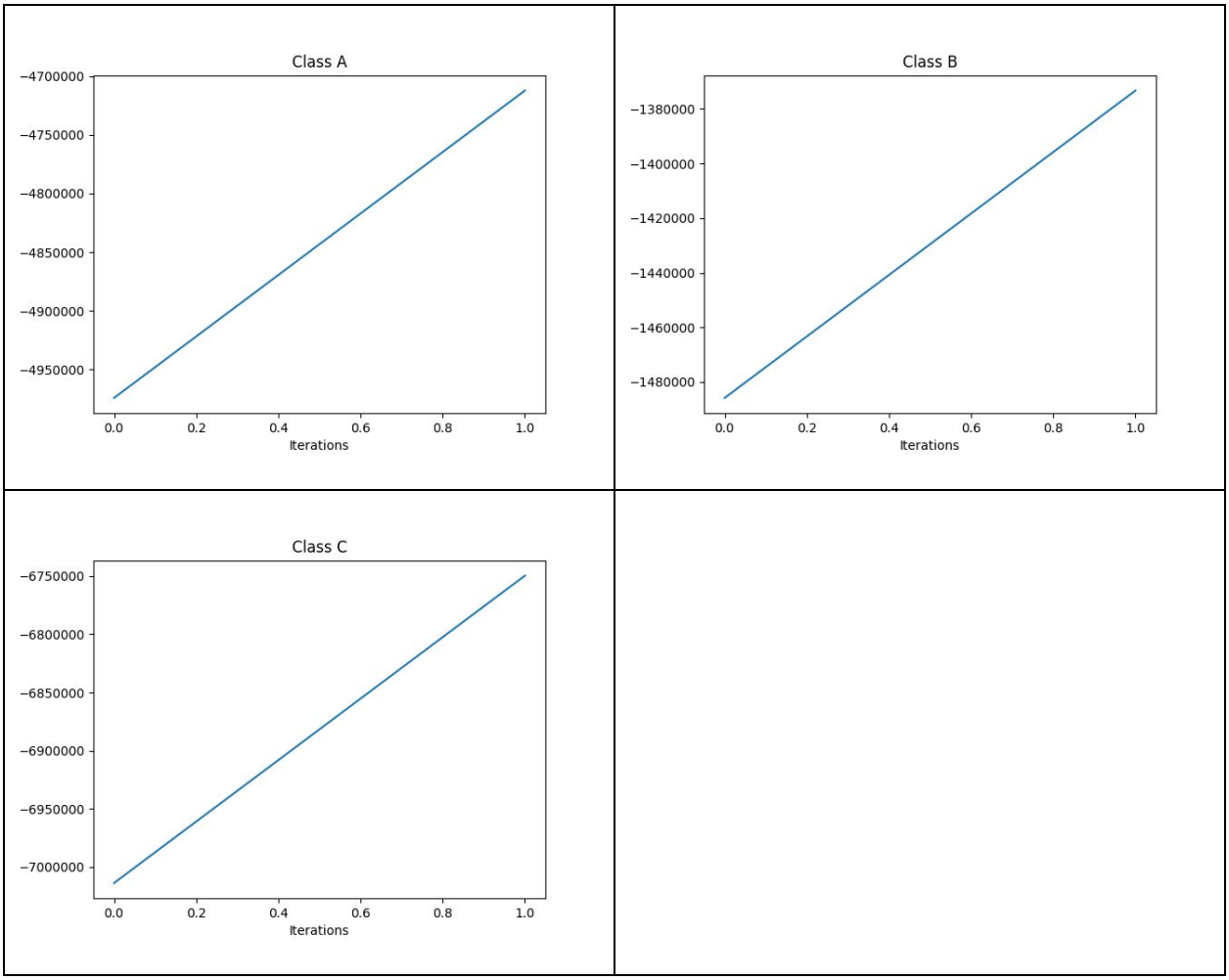


Figure 4.5 The Log Likelihood graph for Real World data, K = 4

In case, the covariance matrix is taken as full covariance matrix --> determinant is resulting in <=0. Since, log of <=0 values is not defined, we forced the multivariate_normal_pdf function to returns 0 in such cases → which is leading to → very big NAN value for probability of certain classes → which is leading to → all the points being classified as that class because max(nan, integer) return nan.

Now, we corrected and made all the covariance matrices diagonal. So, this leads to expected results as below.

Table 4.6a The Confusion Matrix for Scene Image data, K = 4

	Class assigned by the Classifier			
Actual Values		*Coast*	*Industrial Area*	*Pagoda*
	Coast	*36*	*7*	*6*
	Industrial Area	*21*	*16*	*13*
	Pagoda	*36*	*2*	*12*

Table 4.6b The Performance Matrix for Scene Image data, K = 4

	Precision (%)	*Recall Rate (%)*	*F Score (%)*
Coast	*38.71*	*73.47*	*50.7*
Industrial Area	*64.0*	*32.0*	*42.67*
Pagoda	*38.71*	*24.0*	*29.63*
Mean Value	*47.14*	*43.16*	*41.0*

Classification Accuracy: 42.95%

K = 8

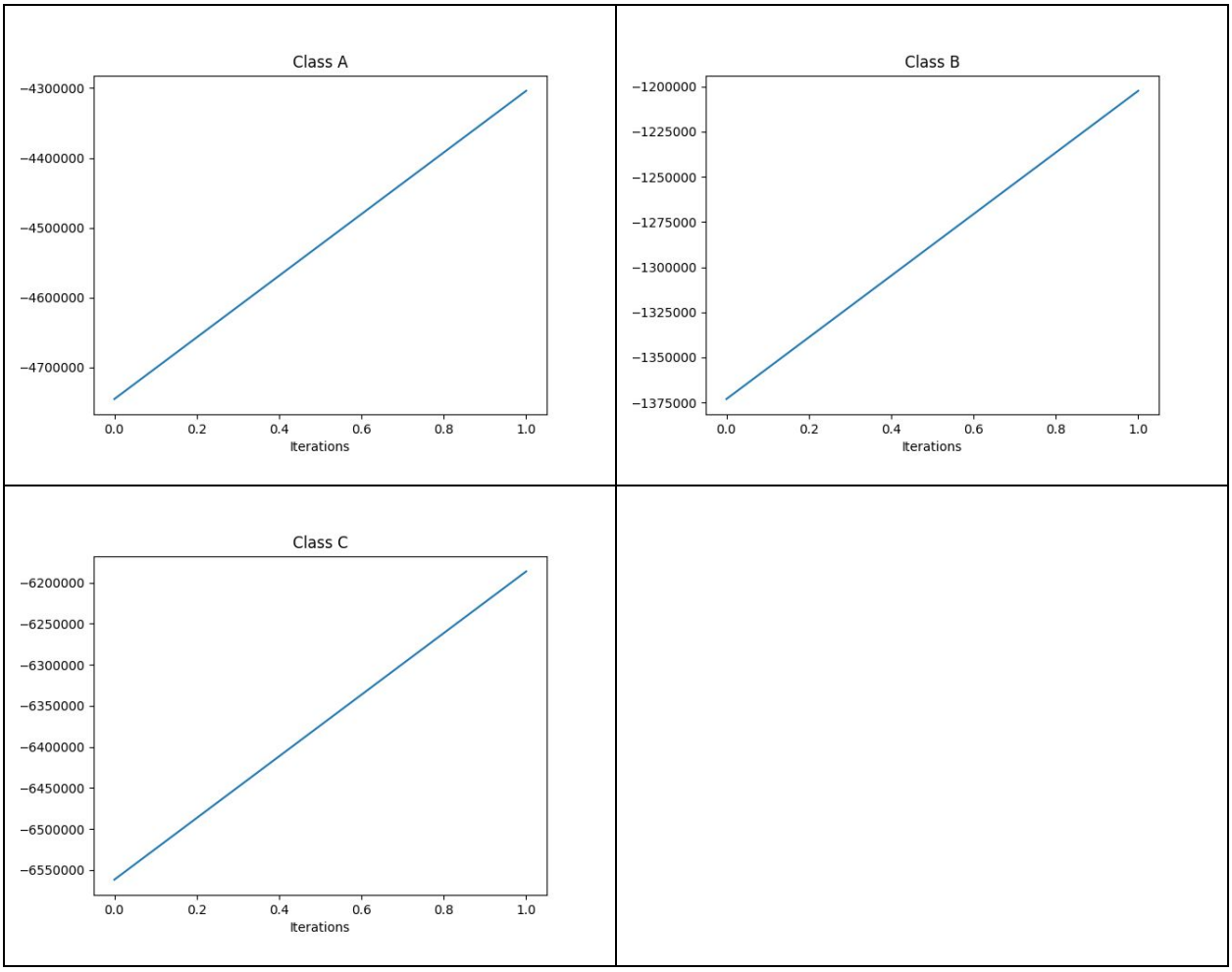


Figure 4.7 The Log Likelihood graph for Real World data, K = 8

Table 4.8a The Confusion Matrix for Scene Image data, K = 8

	Class assigned by the Classifier			
Actual Values		*Coast*	*Industrial Area*	*Pagoda*
	Coast	*19*	*24*	*6*
	Industrial Area	*10*	*15*	*25*
	Pagoda	*29*	*4*	*17*

Table 4.8b The Performance Matrix for Scene Image data, K = 8

	Precision (%)	*Recall Rate (%)*	*F Score (%)*
Coast	*32.76*	*38.78*	*35.51*
Industrial Area	*34.88*	*30.0*	*32.26*
Pagoda	*35.42*	*34.0*	*34.69*
Mean Value	*34.35*	*34.26*	*34.16*

Classification Accuracy: 34.22%

K = 16

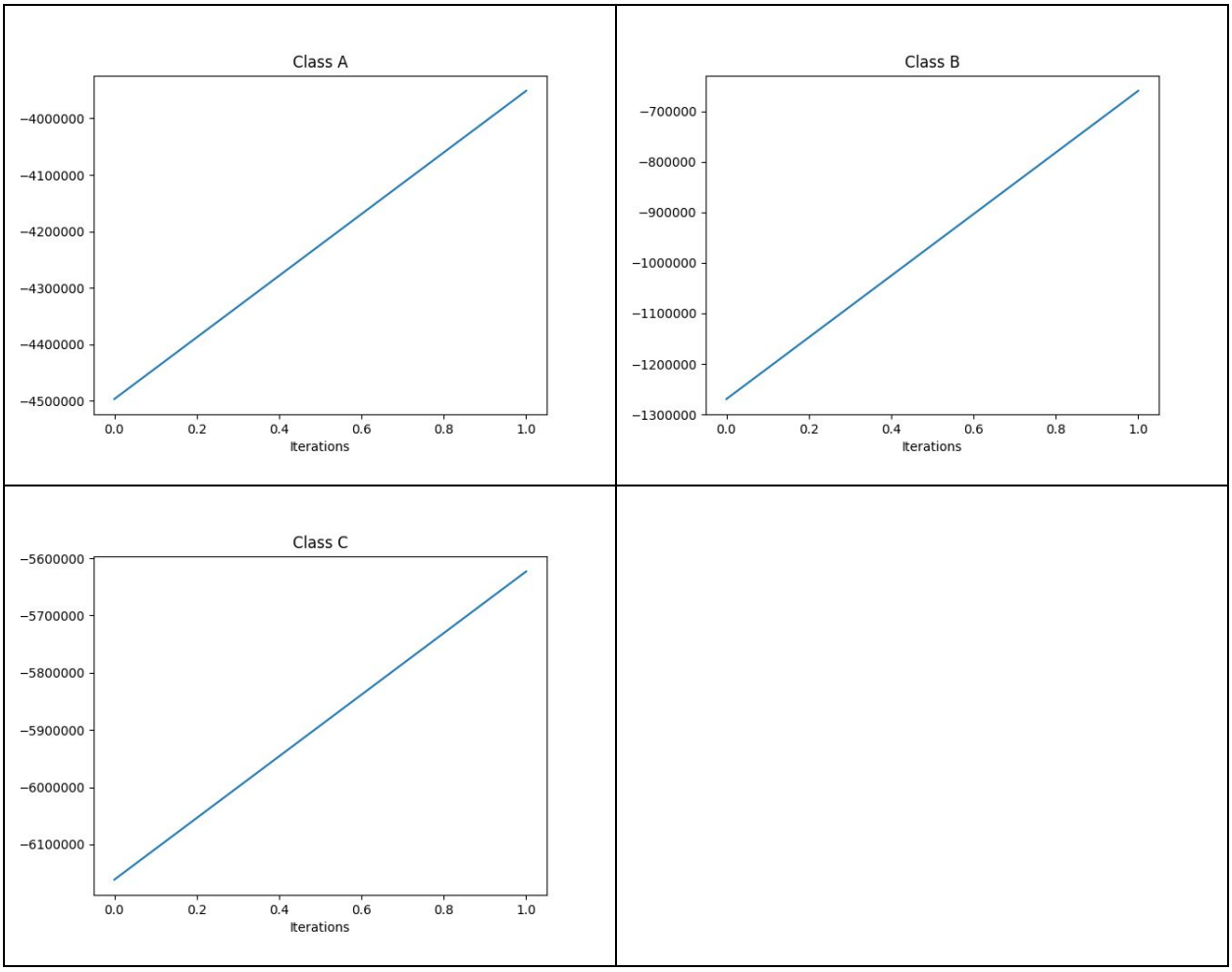


Figure 4.9 The Log Likelihood graph for Real World data, K = 8

Table 4.10a The Confusion Matrix for Scene Image data, K = 16

	Class assigned by the Classifier			
Actual Values		*Coast*	*Industrial Area*	*Pagoda*
	Coast	*28*	*18*	*3*
	Industrial Area	*19*	*18*	*13*
	Pagoda	*24*	*8*	*18*

Table 4.10b The Performance Matrix for Scene Image data, K = 16

	Precision (%)	*Recall Rate (%)*	*F Score (%)*
Coast	*39.44*	*57.14*	*46.67*
Industrial Area	*40.91*	*36.0*	*38.3*
Pagoda	*52.94*	*36.0*	*42.86*
Mean Value	*44.43*	*43.05*	*42.61*

Classification Accuracy: 42.95%

As we can see, with increase in number of mixtures, accuracy is decreasing. Also, the histogram of pixel intensities as feature vectors might not be an ideal feature vector for classification. That might be an additional reason for low accuracy.

5. Cell Image Data

Using K-means

Features of the Cell Images training and test images are extracted as per the procedure given:

1. Consider 7 x 7 overlapping patches with a shift of 1 pixel on every training cell
2. Compute mean and variance of intensities of pixels in the 7 x 7 patch.
3. Thus, a 7 x 7 patch is represented as 2-dimensional feature vector.
4. In the similar way, compute 2-dimensional feature vector from every patch from every training image.
5. Stack all the 2-dimensional feature vectors in a file.
6. For test images: Each test image is represented as a separate file of stacked 2-dimensional feature vectors..

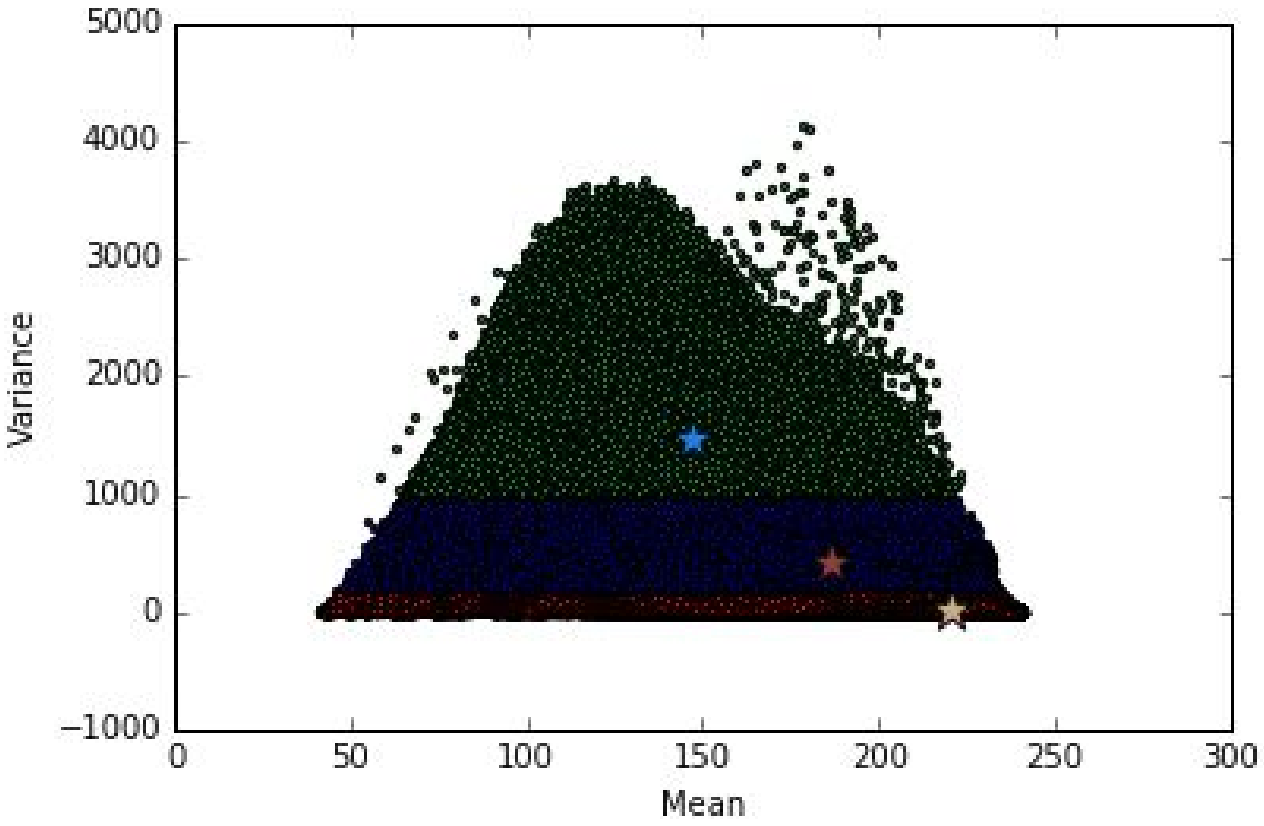
The following procedure is followed for doing this experiments:

1. Mean and variance of the given cell image data has been extracted as per the above procedure.
2. K-means clustering algorithm implemented is trained with training set extracted with the parameter K =3. Once the k-means is converged, it forms 3 cluster centers.
3. Mean and variance of the given test cell images has been extracted and kept as testset as per the above procedure.
4. Testset is passed to the trained k-means and the nearest Euclidean distance of each vector is found and a colour from R, G, B group is chosen for the particular patch of vector.
5. Similarly step 4 is repeated for all the features and the image is clustered and reconstructed.

5.1 Scatter Plot of Training Images after K-Means Clustering

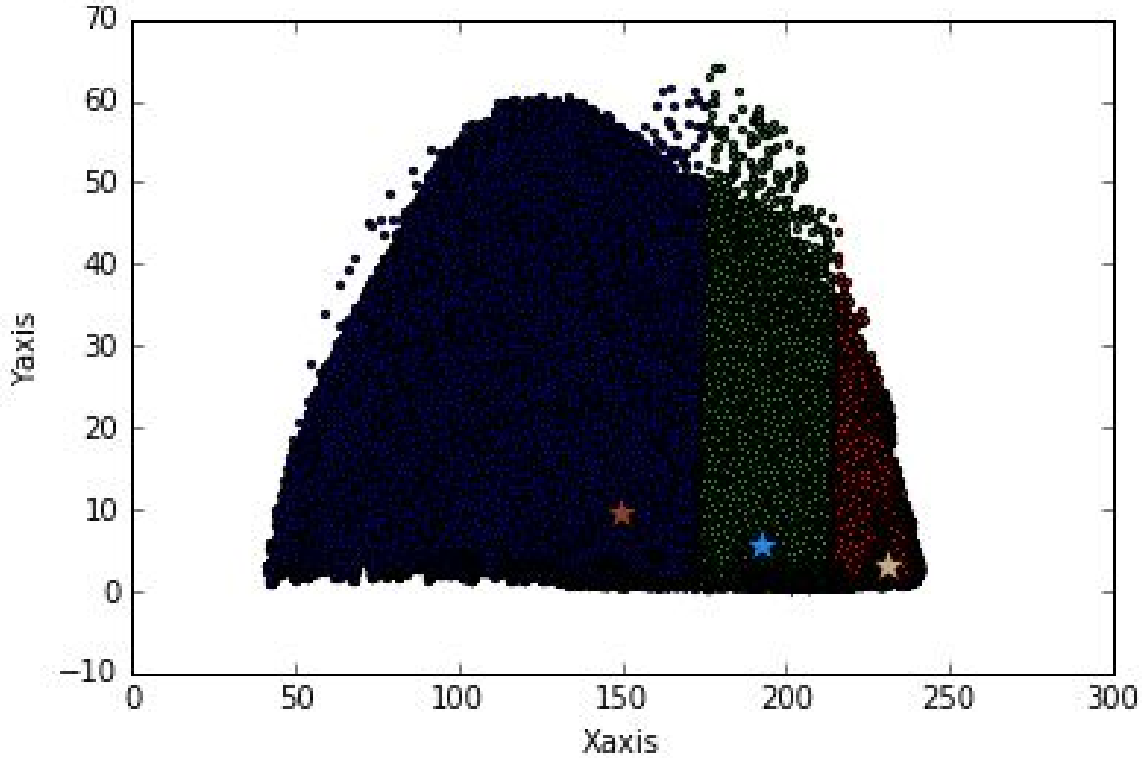
The cluster centers can be seen evidently in the below graph for 3 classes.

K-Means clustering is performed by taking K = 3 because a-priory know there are 3 clusters in the Cell Images..



5.1a Scatter plot of training data points of Cell Image after K-Means

The entire set training points of Cell images ran into 15 million data points. So data took very long to converge after GMM. So we obtained the scatter plot with reduced set of data points as shown below. We can see that the cluster centers at the center of the data points.



5.1b Scatter plot of training data points of Cell Image after K-Means with Mean and Sdv as features

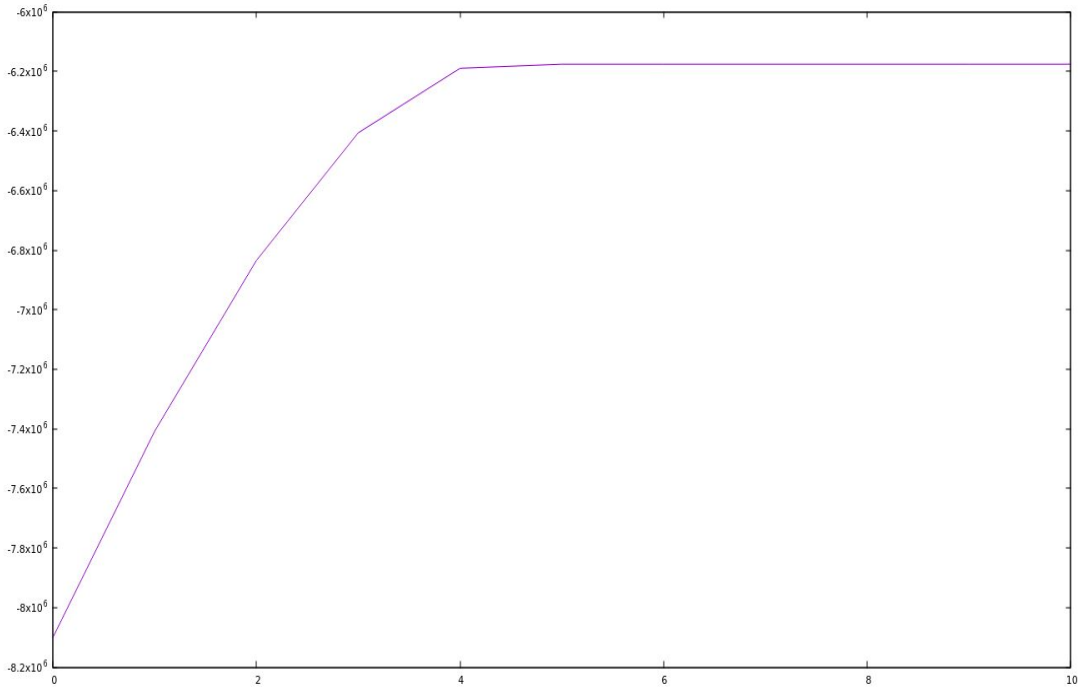
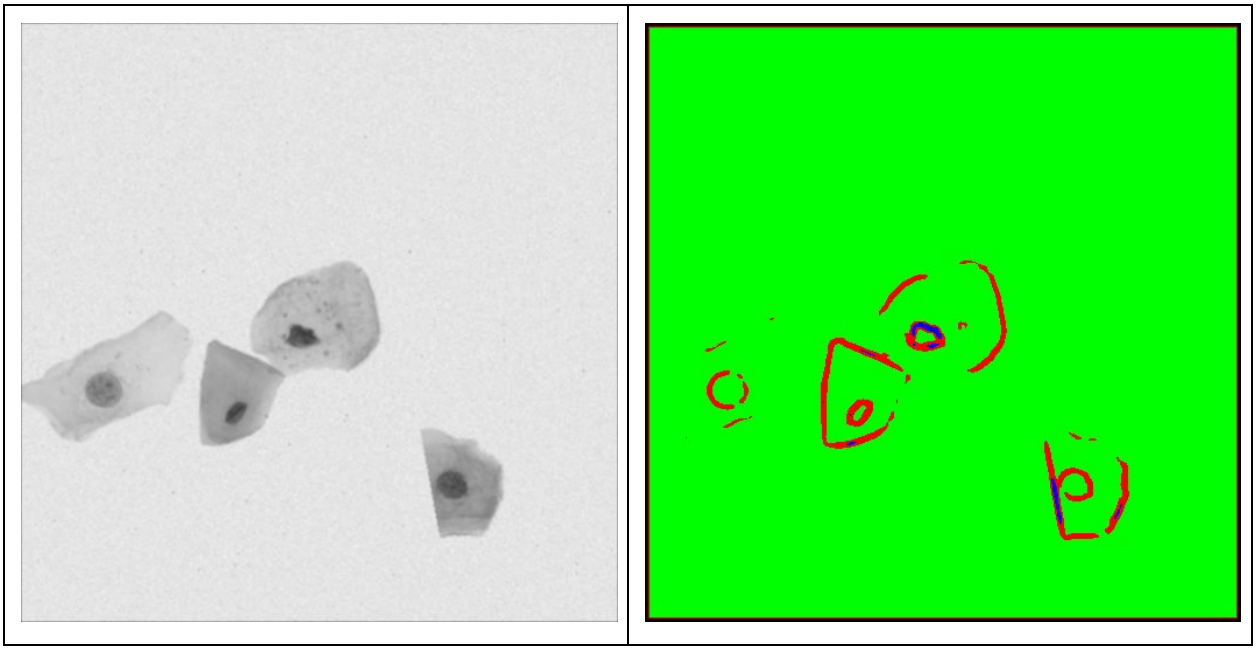


Figure 5.1c The Log Likelihood graph for Cell Image data on GMM

5.2a Scatter Plot of training Cell Test Image 1 Image data after k-means



5.2b Segmentation with mean and std deviation as features after K-Means and GMM

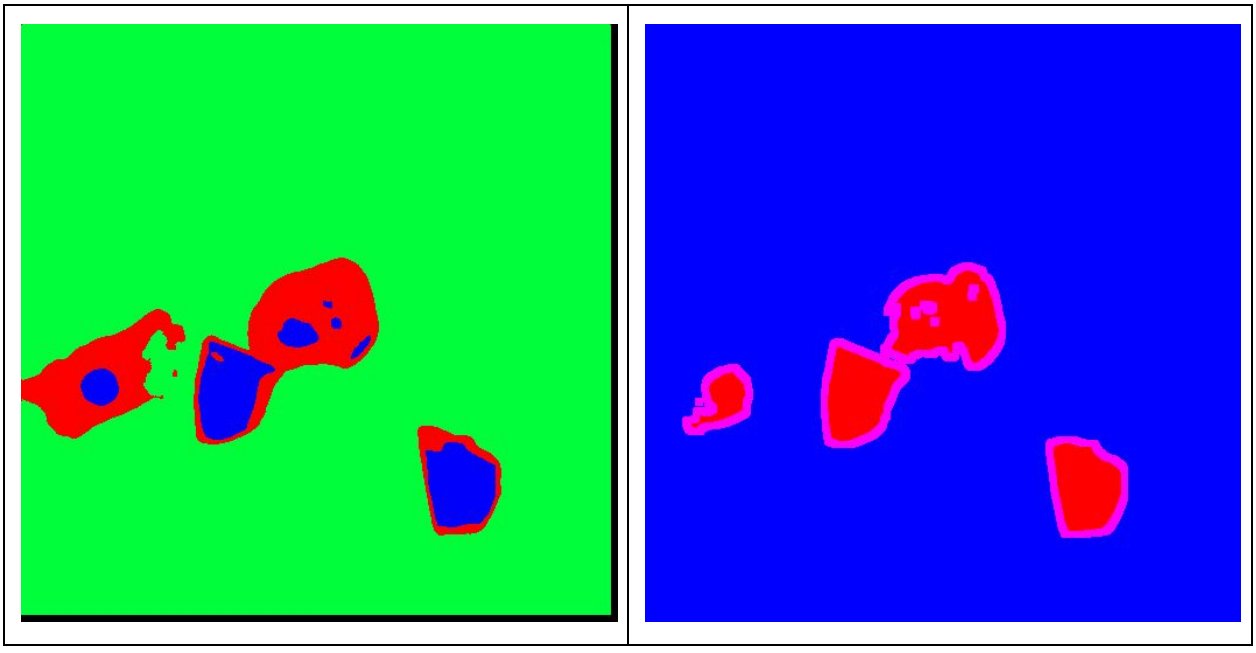
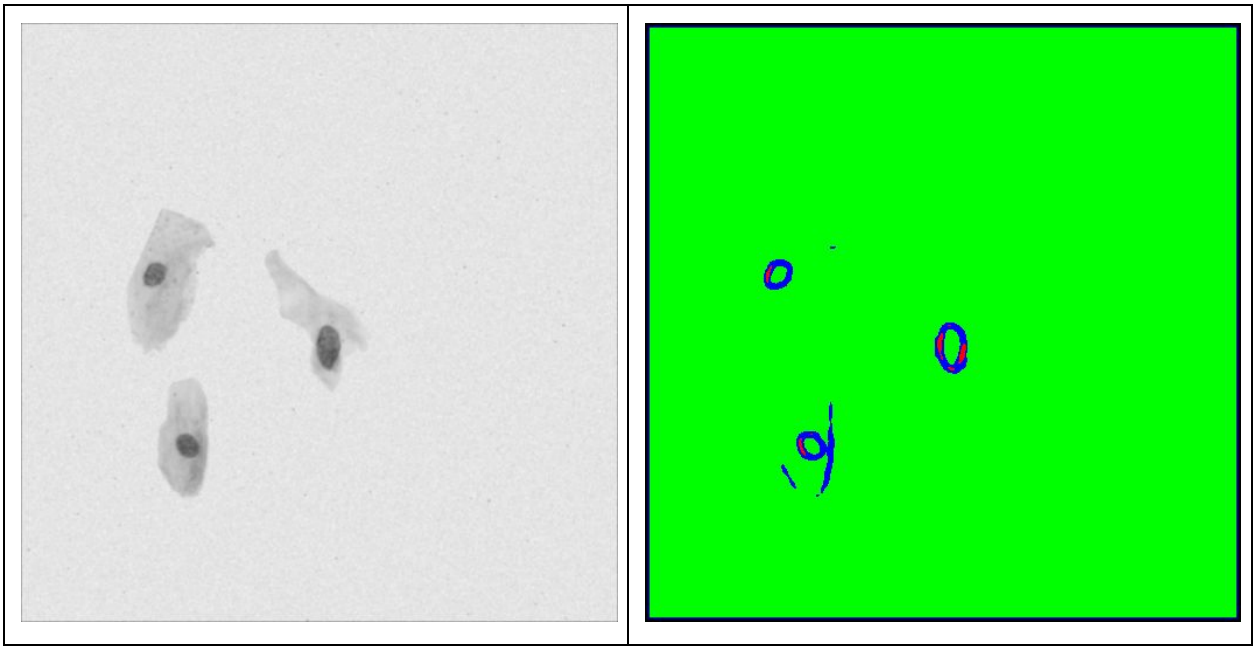
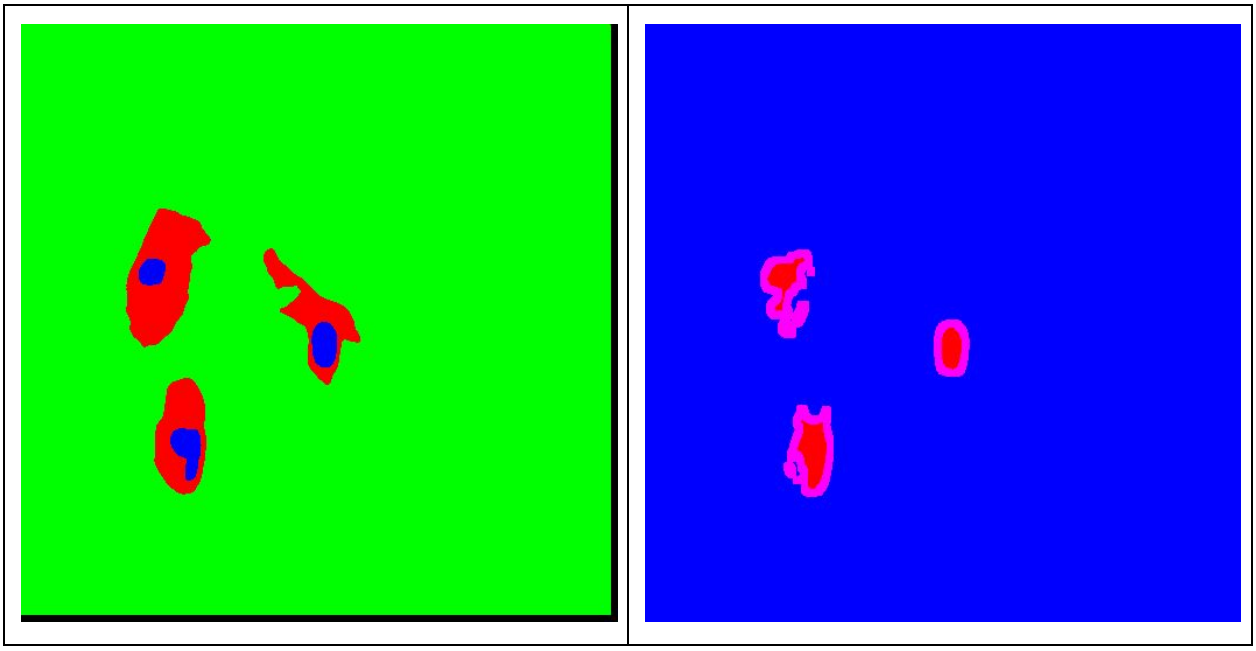


Figure 5.2a shows the image segmentation of cell images using mean and variance as features. So plasma is not segmented properly. But when mean and std deviation were used as features, image segmentation happened more accurately.

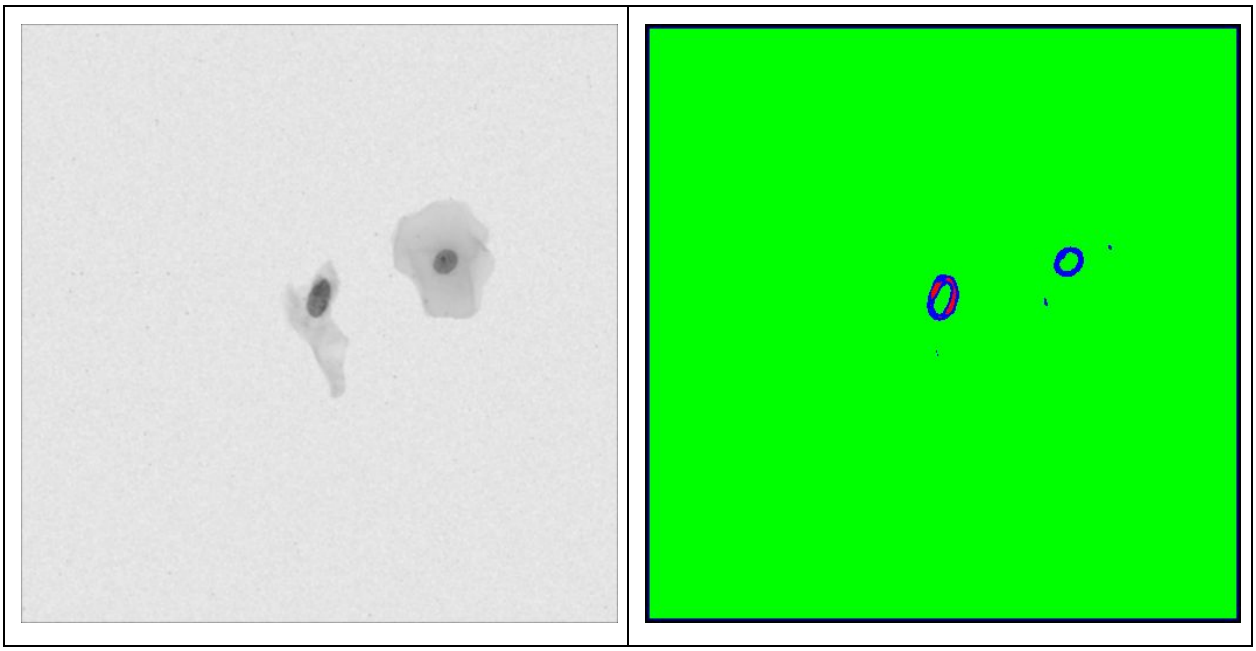
5.3a Scatter Plot of training Cell Test Image 2 Image data after k-means



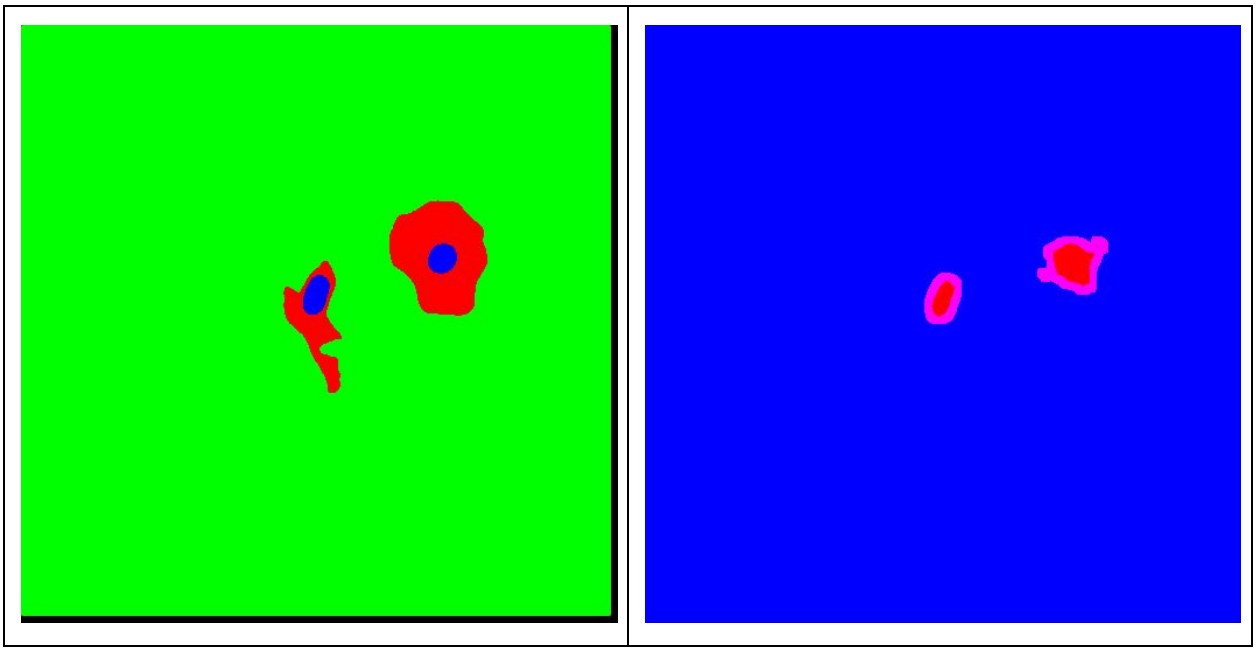
5.3b Segmentation with mean and std deviation as features after K-Means and GMM



5.4a Scatter Plot of training Cell Test Image 1 Image data after k-means



5.4b Segmentation with mean and std deviation as features after K-Means and GMM



6. Bag of Visual Word experiments.

After the K-Means on the 24d Scene Image data with K = 32, we made Bag of Visual Words by making a 32d histogram. Each bin in the histogram represents each of the 32 clusters. Each of the bin is incremented by checking the which cluster the feature got clustered into.

We got following such BOVW feature vectors for 150 images in all 3 training classes put together:

[0, 0, 81, 15, 5, 212, 2, 40, 32, 0, 3, 72, 0, 13, 45, 37, 0, 3, 3, 128, 33, 1, 0, 0, 102, 0, 13, 0, 1, 38, 1, 0]

Now, these 32d data are trained with different mixtures using K-Means and followed by GMM.

K = 1

For k = 1, the parameters are directly computed because data is coming from only 1 cluster. Here, we know the which data point belongs to which cluster. Therefore, we need not run K-Means or GMM in this case.
Log Likelihoods of Class A and Class C dropped from a non zero value to -inf in the 2nd iteration.

Table 6.2a The Confusion Matrix for BOVW data, K = 1

	Class assigned by the Classifier			
Actual Values		*Class 1*	*Industrial Area*	*Pagoda*
	Class 1	*6*	*37*	*7*
	Industrial Area	*5*	*44*	*1*
	Pagoda	*15*	*19*	*16*

Table 6.2b The Performance Matrix for BOVW data, K = 1

	Precision (%)	*Recall Rate (%)*	*F Score (%)*
Class 1	*23.08*	*12.0*	*15.79*
Industrial Area	*44.0*	*88.0*	*58.67*
Pagoda	*66.67*	*32.0*	*43.24*

Mean Value	*44.58*	*44.0*	*39.23*

Classification Accuracy: 44.0%

Beyond K = 2, BOVW did not converge after GMM.

Conclusion:

Non linear Separable data's accuracy is higher when we considered that the data from a single class is coming from several Gaussian Mixture models.

Real World data also shows similar behaviour.

The decision boundary obtained in Non Linear and Real world data is very sophisticated hyper plane compared to the decision boundary obtained with Bayesian Classifier.

Scene Image data, since we considered non overlapping 32*32 patches, the accuracy generally low.

Cell Image data, since we considered overlapping patches, although we ran GMM with reduced set of training data points, segmentation appears close to the original image.