OKOYA Timi'

OMML Assignment 3

SVM and Clustering

The fourclass dataset was utilized throughout for the SVM exercise. The breakdown of this dataset is shown below:

TYPE	CLASSIFICATION	***ORIGIN***	THH96a
FEATURES	2	***REAL/INTEGER/NOMINAL***	(2/0/0)
INSTANCES	862	***CLASSES***	2

THE LIBSVM FORMATTED FOURCLASS DATASET - http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#fourclass

1. The SVM-linear kernel on the FourClass dataset

   Using the *libsvm* interface via MATLAB, I implemented the linear kernel on the dataset and varied the values of C (1, 10 and 100). Since the dataset is not linearly separable, the result was undesired. The plots produced are shown below for documentation. This result can be reproduced using the *svmlinear.m* file included. No visible change while varying the C values.

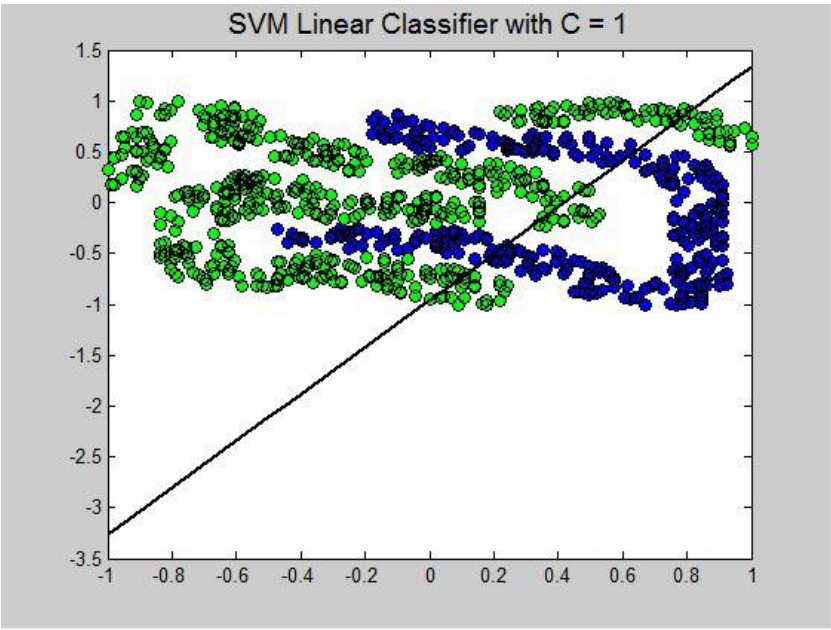


Figure 1

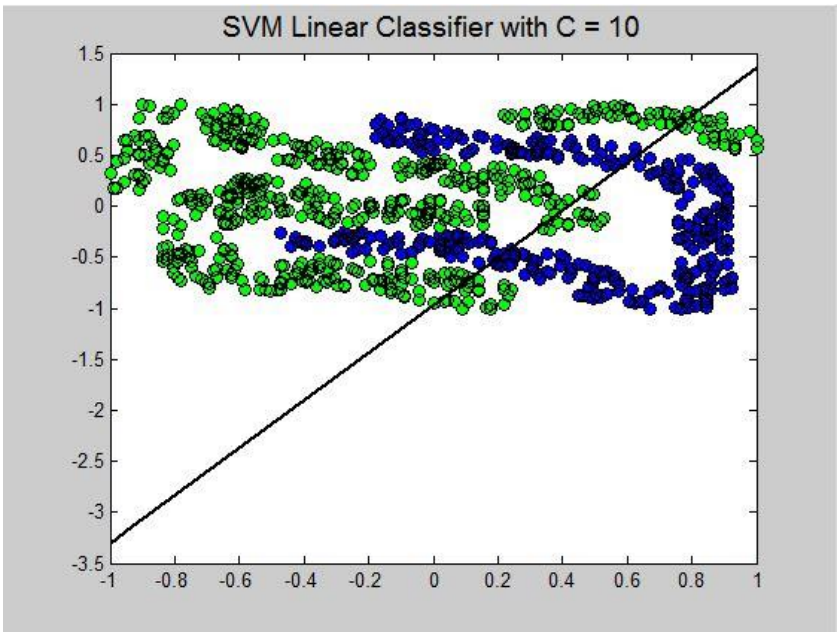


Figure 2

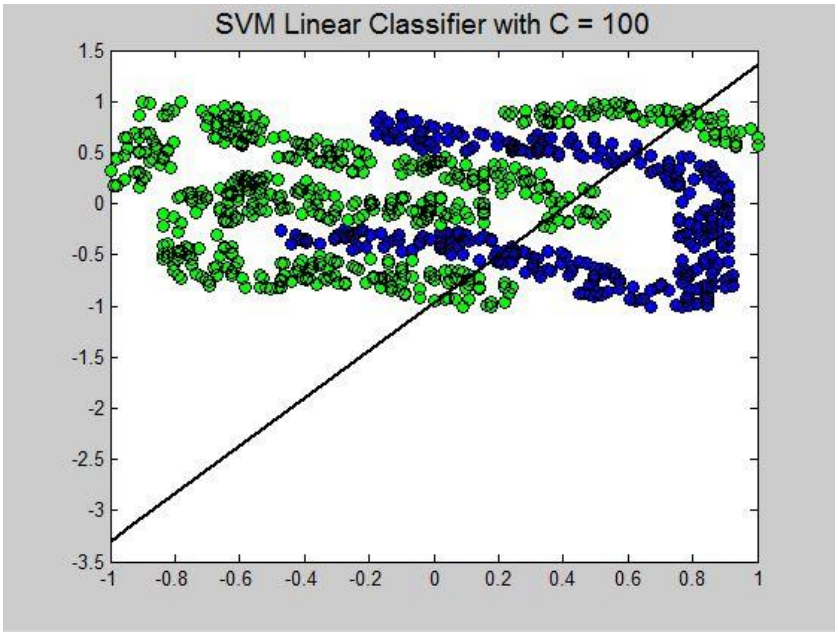


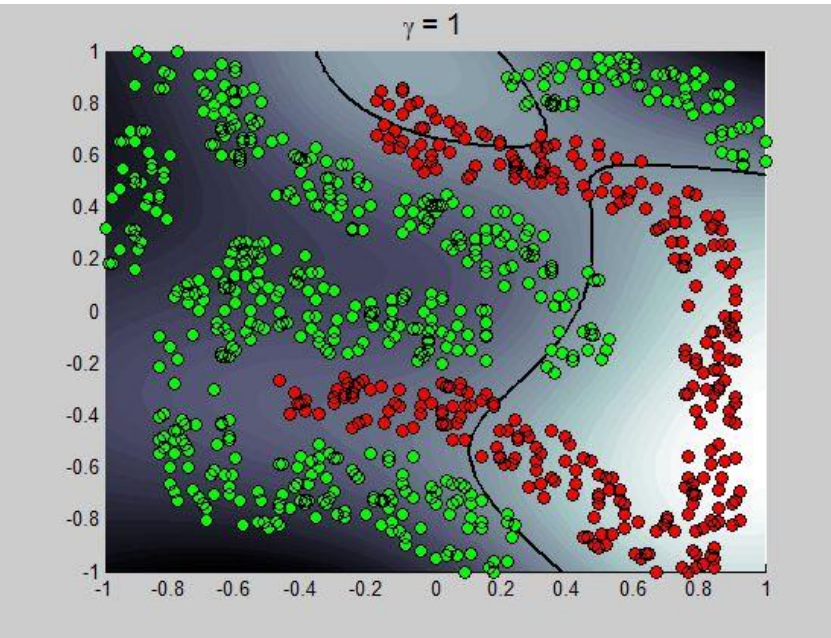
Figure 3

2. SVM-gaussian kernel on the fourclass dataset

The Radial Basis Function (RBF) kernel in LIBSVM was deployed to implement the Gaussian kernel. The formulas are similar. The RBF kernel was used to choose a non-linear decision boundary. The quadratic problem algorithm is implemented already within the *libsvm* setup to pick the optimal values of w and b. As we have seen earlier there is no linear decision boundary for this dataset.

The contour maps give a richer perspective by showing the filled-in contours of the function that gives the decision values used to make the classification. Intuitively, the white areas can be considered as the highest-confidence regions for positive classification, and the black areas as the highest-confidence regions for negative classification. The color gradient shows the change in decision values for making classifications. The *svmnonlinear.m* and *plotboundary.m* files produced this plots

The values of gamma were varied accordingly (gamma = 1, 10 and 100)



Figure

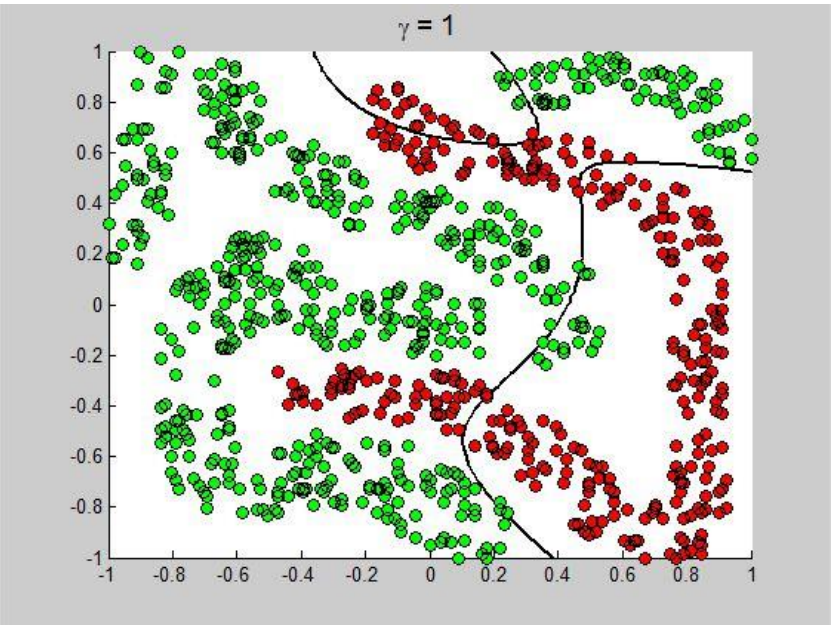


Figure 4

Accuracy = 85.0348% (733/862) (classification)

optimization finished, #iter	302
Nu	0.441489
Obj; rho	-328.880748, 1.699611
nSV; nBSV	384, 374
Total nSV	384

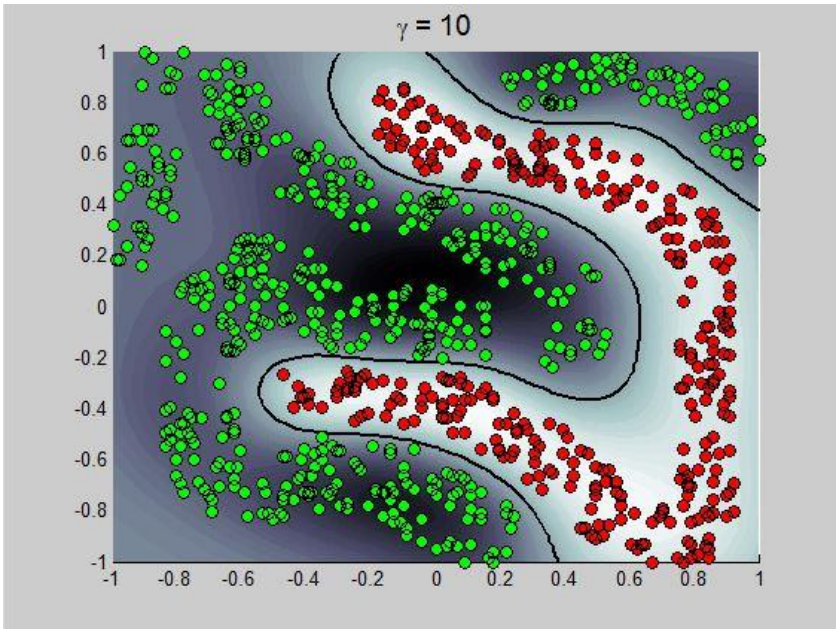


Figure 5

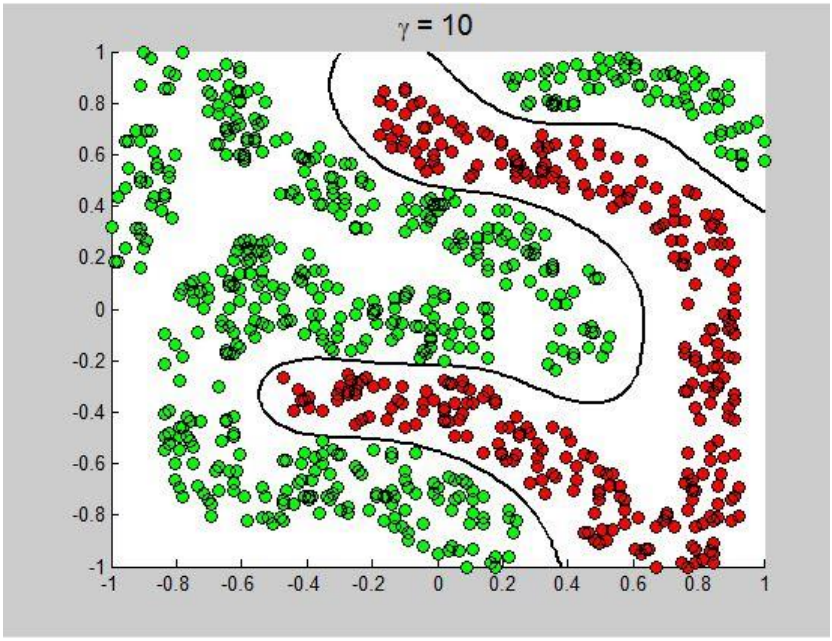


Figure 6

Accuracy = 100% (862/862)

optimization finished, #iter	357
Nu	0.099964
Obj; rho	-50.305413; 0.539181

nSV; nBSV	105; 72
Total nSV	105

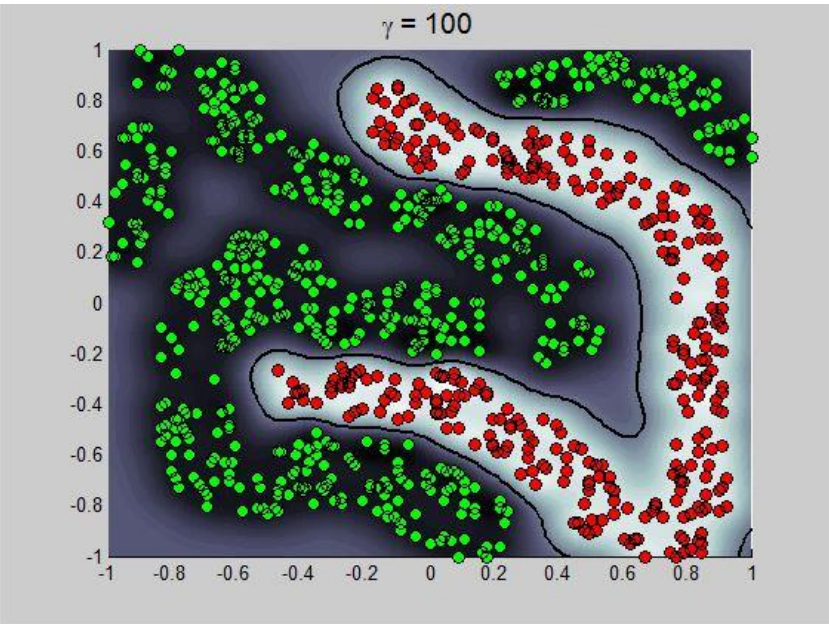


Figure 7

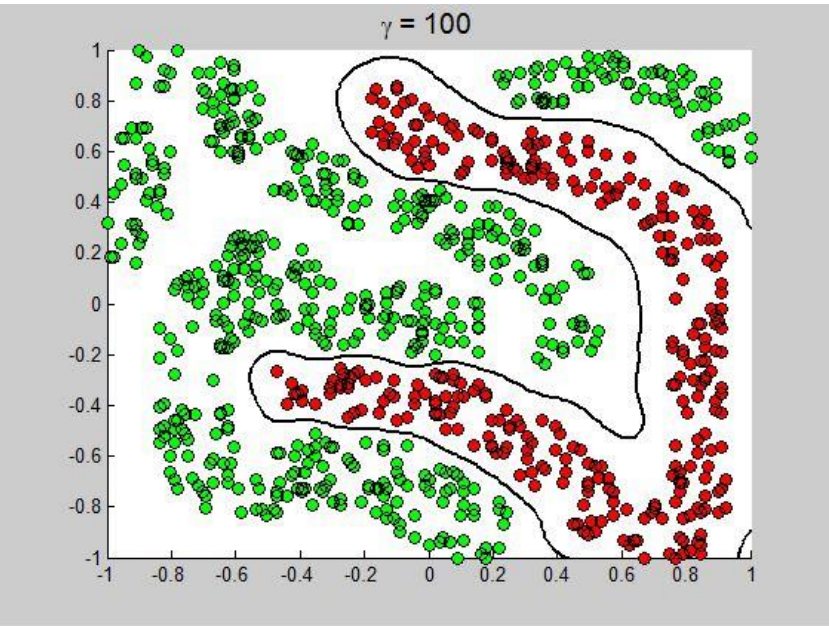


Figure 8

Accuracy = 100% (862/862)

optimization finished, #iter	1614

Nu	0.132870
Obj; rho	-58.473166; 0.283862
nSV; nBSV	309; 14
Total nSV	309

In conclusion, the figures above show that as ϒ increases, the algorithm tries harder to avoid misclassifying training data, which leads to overfitting. It is not quite straight forward how to set a value of ϒ that works well for a model without overfitting the data.

3. The Matlab kmeans dataset was used for this exercise. The error plot of the analytical approach and the visualization of the clusters can be seen in the figures below. *K_means_.m* and *kmeans_analytical.m* produced these plots.

TYPE	CLUSTERING	***ORIGIN***	MATLAB
FEATURES	4	***REAL/INTEGER/NOMINAL***	(4/0/0)
INSTANCES	96	***CLASSES***	-

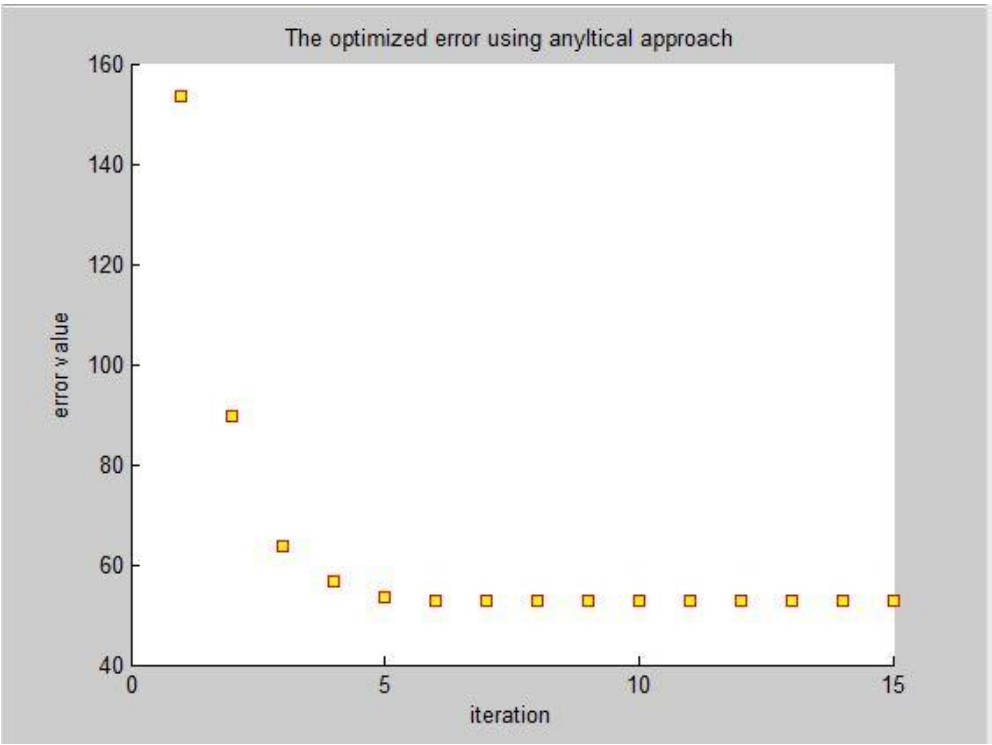


Figure 9: The error plot for the analytical approach

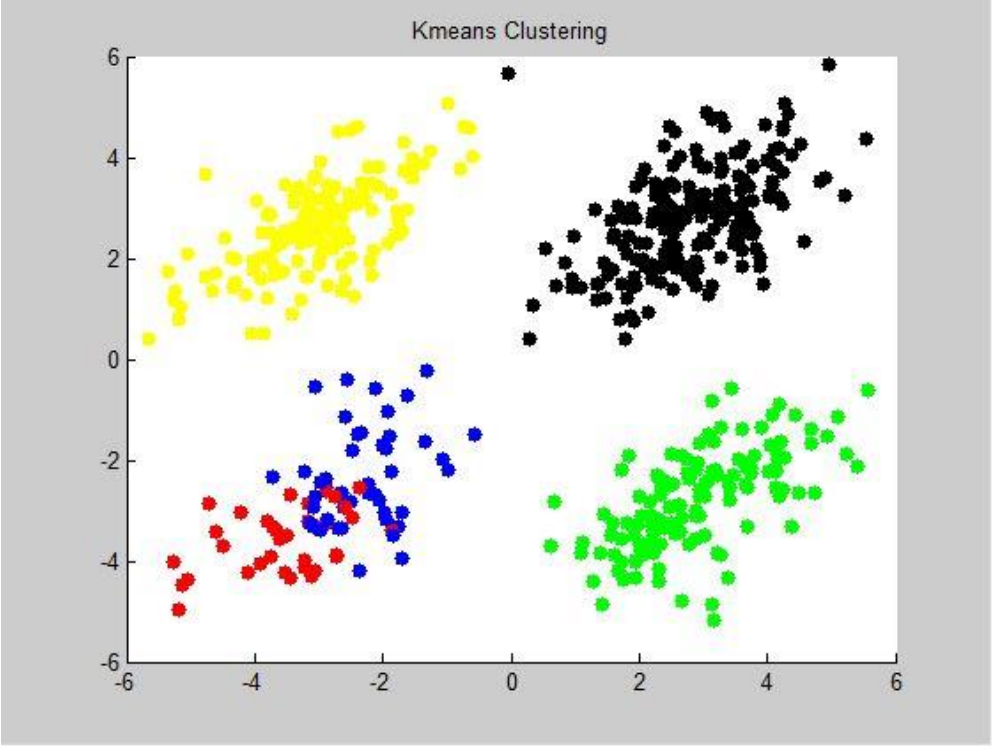


Figure 10: Visualization of the clusters

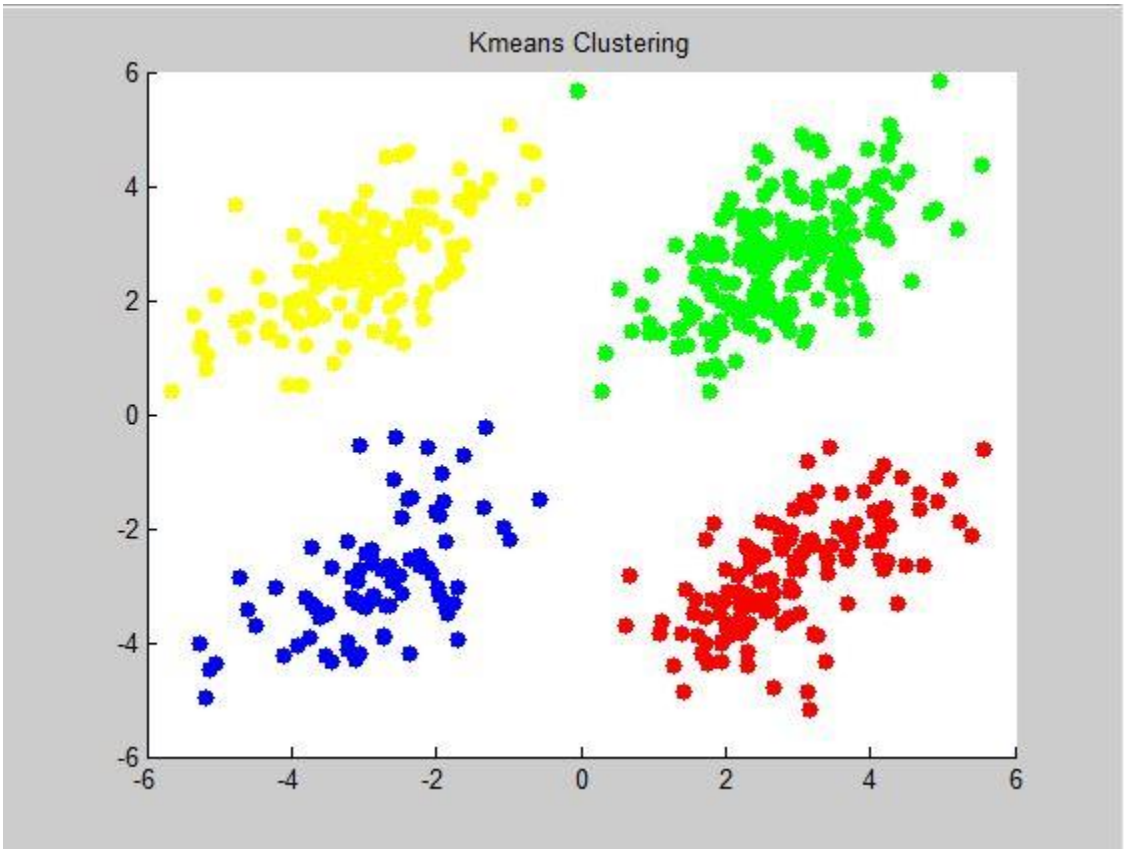


Figure 11: Visualization with 4 clusters