Informatics 2B coursework 2 – Learning
Stefan Sabev - 1024819

The aim of the assignment is to classify a set of 90 and 900 in 3 dimensional space using the k-means clustering algorithm and using the Gaussian multivariate probability density function for each point in the cluster.

1. Explanation of the code

My code should be fairly straight-forward and easy to follow. The code consists of 4 Matlab .m files, 3 of which are functions used throughout the code and the 4th one is the main program where all the computations and function calls are carried out. The whole code is ran from code.m.
The structure is as follows:

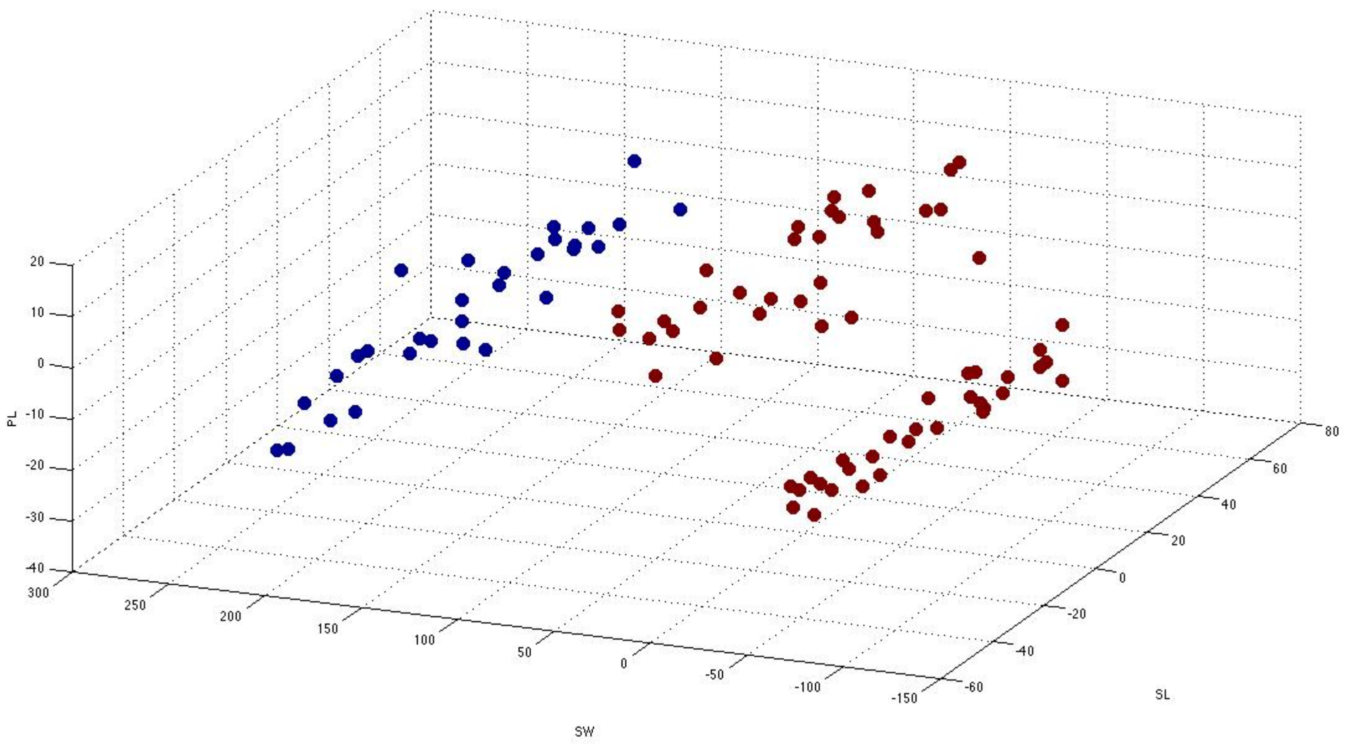
- init_values – provided the set of data points this function will return the distance matrix and the first two means. In order to do so, I just create a big 90x90 matrix which is symmetrical with respect to the diagonal and contains between two points. For instance the value of B(39,30) will be the distance between points 30 and 39. It finds the maximum value and its index and then returns a matrix of the two means.
- NM(short for NewMeans) – Given the data, the size of the data and the previous two means, this will calculate the new mean and return it. It's done by calculating the distances between a point and the current means and multiplying them together.
- mykmeans – This is my implementation of the algorithm. It uses the means as initial centres for the clusters and assigns the points for the clusters. I store the clusters in a variable called temp_clusters, which is used to determine when the algorithm converges and there is no change in the assignment of the points to the clusters. When the temp_clusters are equal to the produced clusters after one iteration, then the algorithm has converged. The function returns the centres, the clusters and the sum squared error.

In code.m I carry out most of the computations. Initially I load the data, then build the initial values using init_values. After that I cluster the data using 2-means, add addition mean and repeat for 3-, 4- and 5-means. I then fit the Gaussian for the three clusters, by finding the covariance matrices for each of them. Data_900 is classified using the Gaussians and confusion matrix for that is provided. After that the 3-means clusters are found for data_900. In order to produce correct confusion matrix I had to assign every point assigned to cluster 2 to cluster 3 and vice versa. This is because the means are taken in a different order.

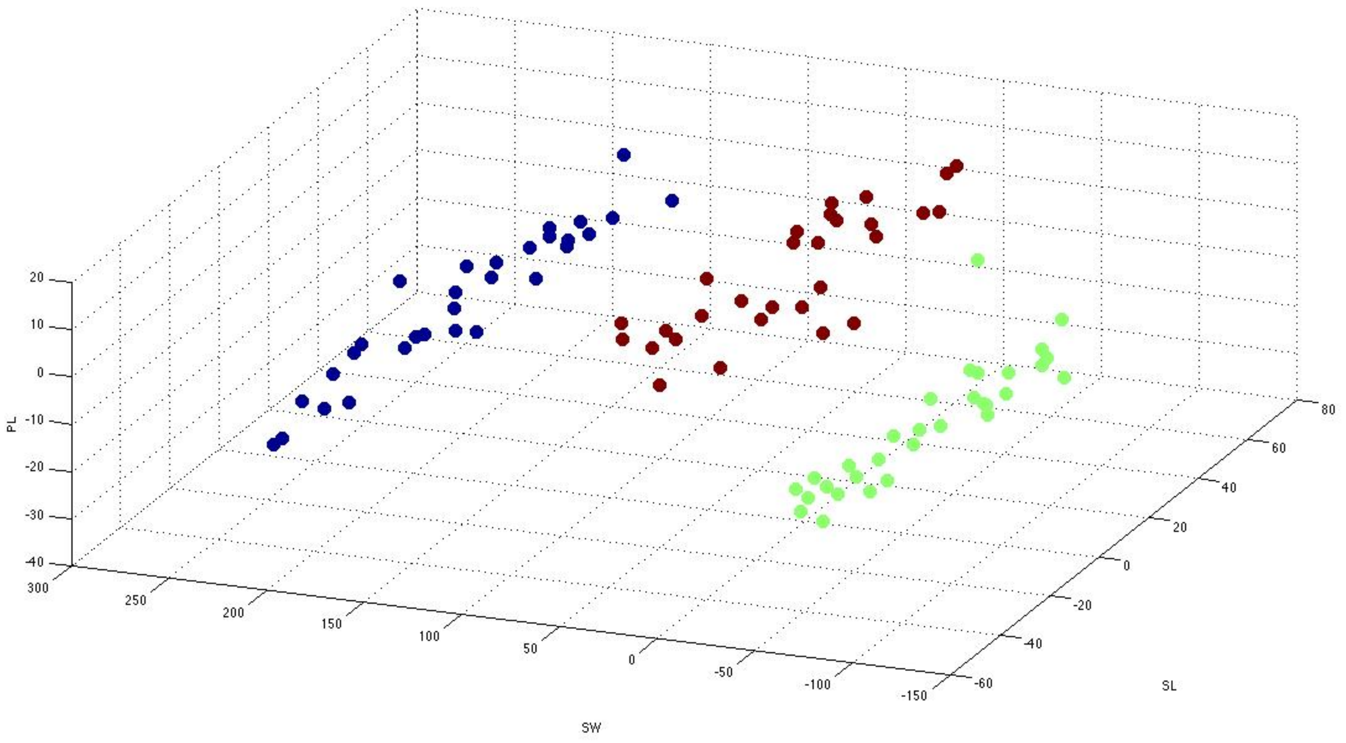
2. Results

The assignments consisted of several small subtasks. The 1st one was to generate clusters using the k-means algorithm. The number of clusters was 2,3,4 and 5. Here you can see how they look in 3d. I tried to take the screenshot from the angle where one can most easily see the distinction between the different clusters.

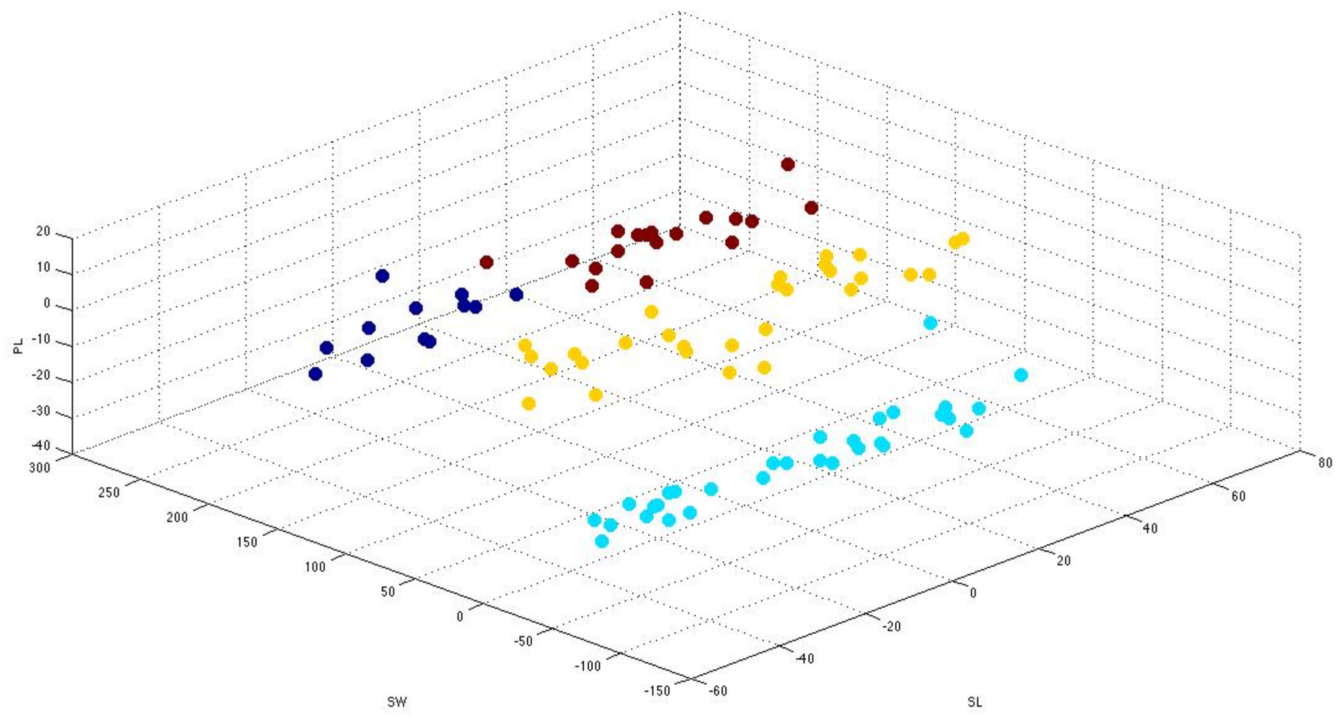
2-means for data_90:



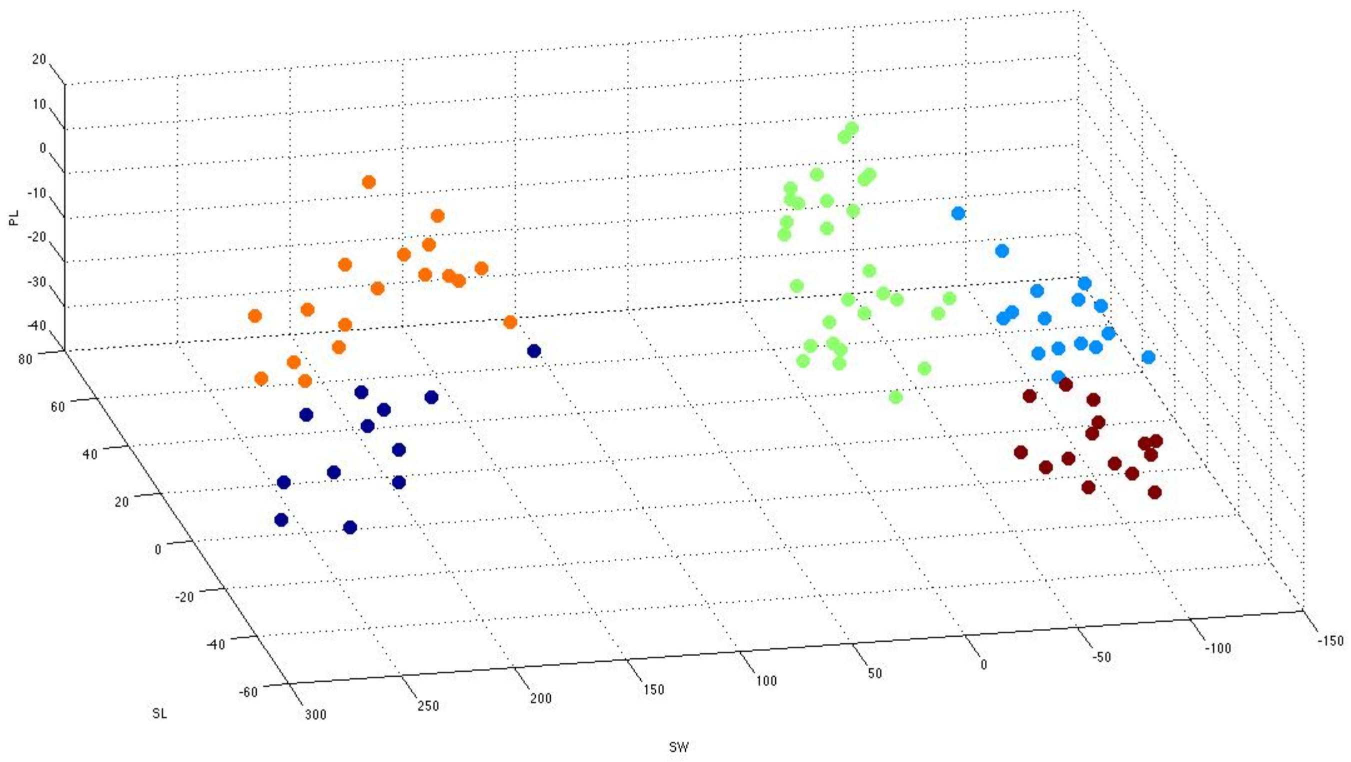
3-means for data_90:



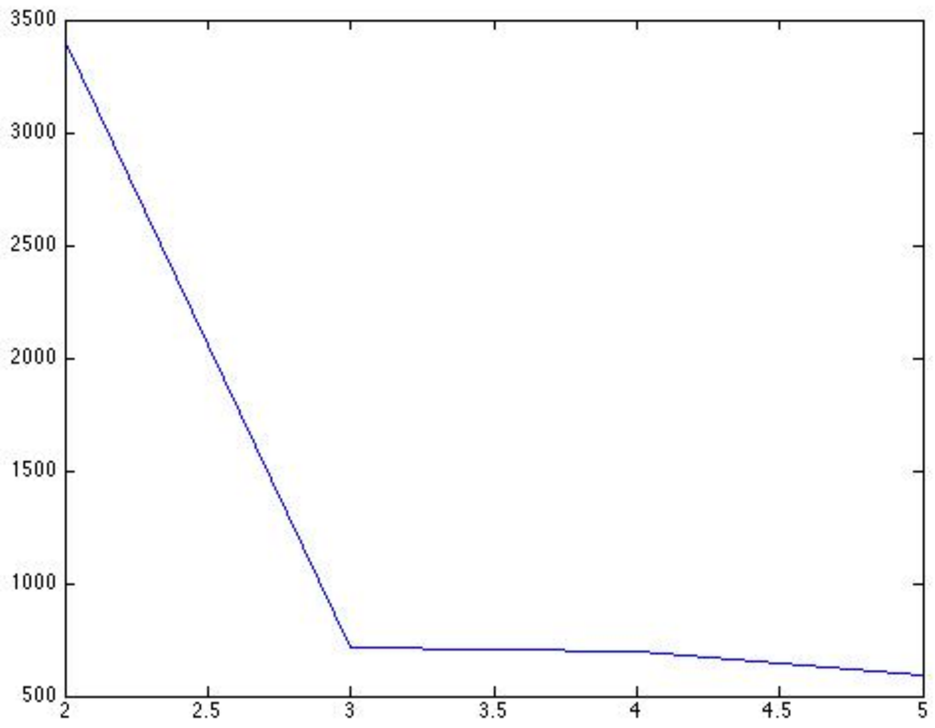
4-means for data_90



5-means for data_90



We were also asked to provide a plot of the sum squared error for the 2,3,4 and 5-means:



Using the clusters obtained in part 1, we had to fit 3 Gaussians to them. In order to do that, we had to calculate the covariance matrices for each of the clusters and the means, which are actually the centres of the clusters.

Here is the matlab output for the three Gaussians:

Means		
22.3244	203.5259	-13.2099
-14.8388	-81.5841	-9.3343
0.9240	9.5330	10.9567

Covariance 1		
775.8	-294.9	-1.7
-294.9	1033.2	-352.2
-1.6	352.2	151.8

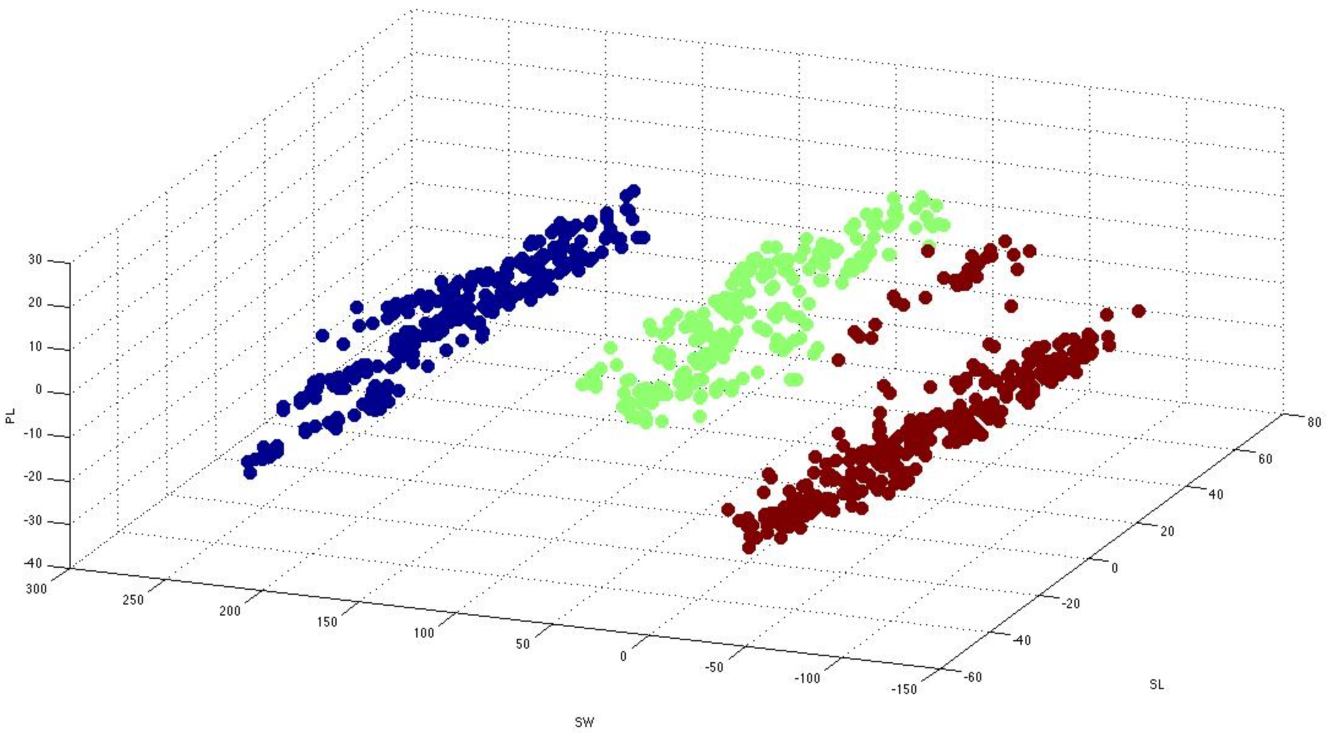
Covariance 2		
618.5044	-311.575	60.6104
-311.5759	493.5243	-3.3160
60.6104	-3.3160	11.8948

Covariance 3		
802.5672	-147.576	38.5467
-147.5768	317.3709	-19.371
38.5467	-19.3715	18.6775

Then we had to classify the data from data_900.m using the Gaussians. The confusion matrix for that is:

Confusion matrix gaussian		
300	0	0
0	300	2
0	0	298

And then classify the same data using the k-means algorithm and fit the data into 3 clusters.



The confusion matrix for that is:

Confusion matrix kmeans		
300	0	0
0	268	32
0	0	300

3. Analysis:

The results produced from the Gaussian classification did not differ from what was expected. The overall proportion of test patterns correctly classified was (300+300+298)/900 = 0.99(7). Some of my fellow students got 100% correct classes, which means I might have a small mistake somewhere in the formula. The results produced from the kmeans algorithm were also almost exactly what I expected. For data_90 there is only 1 point assigned to the wrong cluster and for data_900 that number is 32, which considering the fact that we have a set of 900 points. So, for the 3-means ran on the data_900 we have a proportion of

(300+268+300)/900=0.96(4), which I think is still a very high percentage of accuracy.

As a conclusion I can say that the results produced are not very different from the expected.