Symbols, Patterns and Signals: CW1 Report

Aaron Wray
`aw16997@my.bristol.ac.uk`

Nick Broom
`nb16568@my.bristol.ac.uk`

March 16, 2018

1 Introduction

The aim of this assignment was to gain experience in the practices and methods involved in clustering and classifying data. There are many methods that can be used in achieving this goal; for clustering some of these include connectivity, distribution, density and centroid based clustering. For classifying these include decision trees, neural networks, support vector machines, naïve Bayes and Bayesian belief networks. In our assignment, we looked to cluster our data using centroid based clustering and classify our data using two different approaches: a nearest centroid classifier and a maximum likelihood classifier. The nearest centroid classification generated three cluster centroids through use of the K Means algorithm on the training data. It then classified all the test data based on which centroid it was closest to. Using a maximum likelihood classifier was more complex as this required generating a multivariate normal distribution. Then we plotted the decision boundary as all the points in the plot which had an equal probability of belonging to two normal distributions.

2 Feature Selection

Feature selection is a technique used in data classification, given a set of features/attributes, as input, we want a subset of features which allows us to classify the data in the best possible way. All extraneous attributes are then essentially discarded with in the model.

The training data that we were provided with contained 150 values for 5 different features in total. There are various ways of performing feature selection. For example, using Pearson's correlation coefficient, mutual information or other ensemble methods[7]. We used a manual exhaustive search method, as there were a small number of features, which involved plotting the data points of each feature against another feature. This resulted in a 5x5 matrix as seen below in Figure 1. The leading diagonal of the matrix represents each feature of the training data plotted against itself. The graphs either side of the leading diagonal are equivalent, but with the axes switched. Upon visual inspection of the matrix below the graph in the 3rd column 2nd row or similarly, 2nd column 3rd row, hold the most significance. As they are the only combination of features that exhibit the training data grouped into 3 distinct clusters. Using this information, we selected theses features to generate our model.

3 Identifying the Classes

To classify exactly which points belonged to each cluster we used the K-Means clustering algorithm.

The K-Means algorithm returned to us three values. An array containing the centroids for each cluster, an array called labels that contained a value of 0,1 or 2 for each data point based on which cluster it belonged to and an inertia value which indicates how spread the data is within clusters. The np.where() function will take in an array of values and a condition based on each value. It then returns an array containing the indicies of all the values which satisfy the condition. We used the np.where() function[1] with the condition that labels $==$ 0 to get the indicies of all the points in the training data which belonged to cluster 1. Finally we repeated the process for clusters 2 and 3 but changed the condition so we were checking if the labels value equalled 1 or 2. We were then able to to use these indicies to filter the training data three times to obtain all the points that belonged to a particular cluster.

We initially decided run the K-Means algorithm using the default parameters. After inspecting Figure 2 we were satisfied that the algorithm had correctly classified all the points. Therefore we didn't feel the need to increase the amount of iterations of the algorithm or initial configurations for the cluster centres.

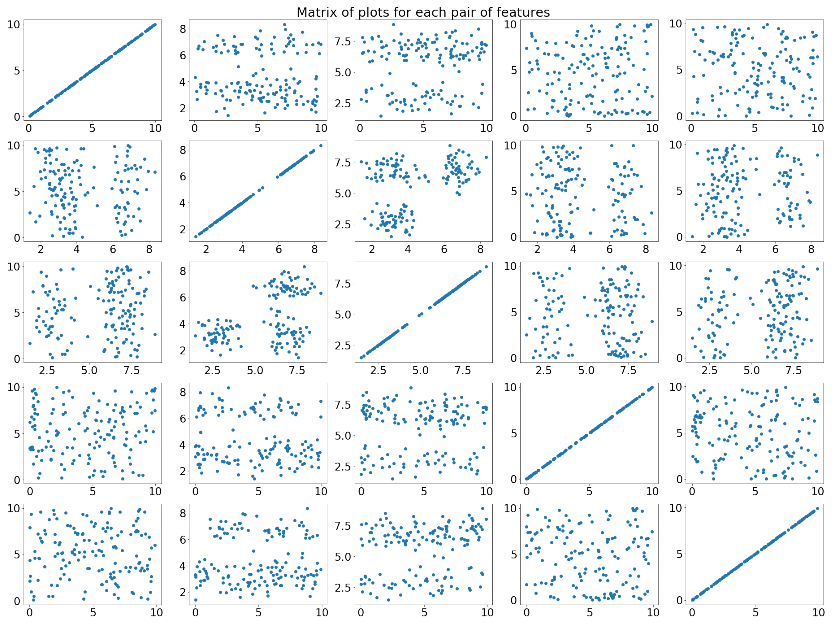


Figure 1: Displays each feature of the training data plotted against each other

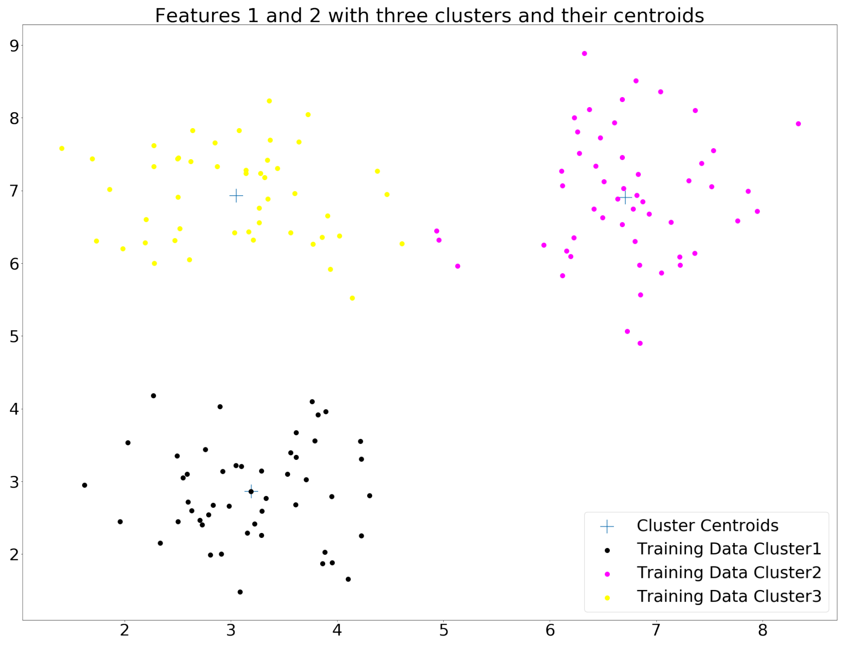


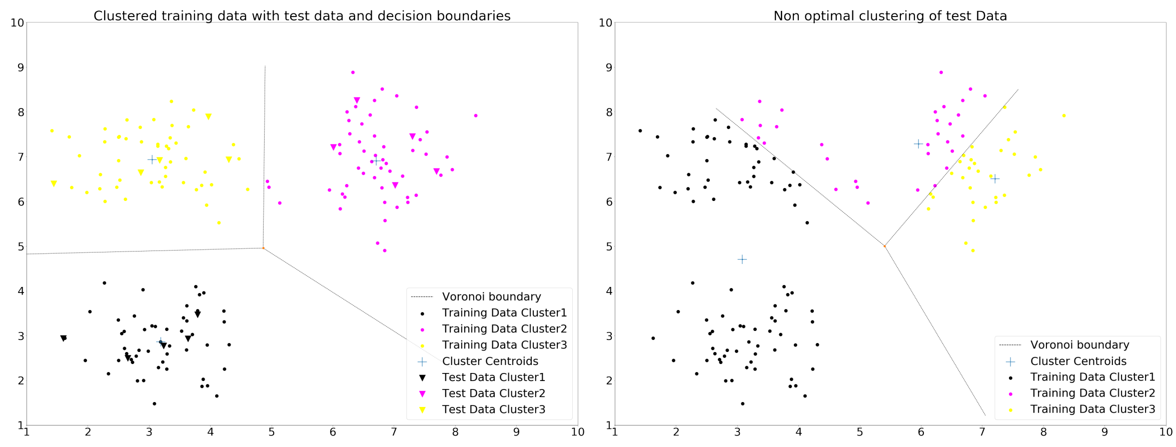
Figure 2: Displays our training data in three clusters and the centroid of each cluster

However, to show, the contrast of a non-optimal clustering we changed some of the initialization parameters of the K-Means algorithm to produce the result as shown in Figure 3b.

We manually inputted the initial cluster centres and deliberately made these non-optimal. We also ensured that this would be the only initial configuration of cluster centres the algorithm would use. Finally we reduced the number of iterations in the algorithm from 300 to 4 to ensure that the algorithm would not converge on an optimal solution.

The non-optimality of the clustering in Figure 3b can be seen when we compare the inertia values of the two clusterings. Where the inertia of a clustering represents the sum of all the distances between a data point and its cluster centre squared. Therefore, using the inertia we can easily calculate the standard deviation of the data using the formula below where n refers to the number of training data points:

$$StandardDeviation = \sqrt{\frac{inertia}{n-1}}$$



(a) Optimal clustering. (b) Non-optimal clustering.

Figure 3: Contrast between an optimal and non-optimal clustering of our training data.

The inertia of the optimal clustering is 150.18 compared with 513.13 in the non-optimal clustering which indicates that there is a signifcantly larger average distance between each point and its cluster centre in the non-optimal clustering.

4 Nearest Centroid Classification

Once we had put our training data into three distinct clusters through use of the K means function we then had to classify all our test data based upon which centroid was the closest. To do this we used the cdist function to find the euclidean distance between each data point and the three centroids. Following this we used the numpy.argmin() function to return a 0, 1 or 2 for each data point based on whether it was closest to cluster 1,2 or 3.

We realised that the information we had and the result we were looking for was similar to the problem of identifying the classes given a set of labels. So we went back in our program and generalised part of our question 2 by making a function that given a set of labels and some data points would put the data points into clusters. Then a further function to plot these clusters on our figure to avoid repeating similar code.

After we plotted the test data points on the figure we then plotted the decision boundary which is shown by the dotted lines in Figure 3a. The dotted lines represent all the points on the plot which are equidistant from two cluster centres. With the orange dot at the intersection of the three dotted lines representing the point which is equidistant from all three cluster centres.

We initially used the euclidean distance as our distance measure but after researching the documentation for cdist[2] we found that there are 21 other methods for calculating the distance between between two vectors. So we decided to test two other distance measures, the Canberra and Manhattan distances[6, 4]. The equation below will calculate the Manhattan distance between two vectors, p and q:

$$d_m(p,q) = \sum_{i=1}^{n} |p_i - q_i|$$

The following equation will produce the Canberra distance between two vectors, p and q:

$$d_c(p,q) = \sum_{i=1}^{n} \frac{|p_i - q_i|}{|p_i| + |q_i|}$$

However, as is evident in the diagram most of our test data points lie quite far from the nearest centroid decision boundary and so therefore changing between these different distance measures didn't change any classifications.

5 Maximum Likelihood Classifier

In order to produce a maximum likelihood classifier for our clusters we needed to model each cluster using a two dimensional normal distribution. We started by calculating the mean, a row vector of length 2 and the covariance, which was a 2x2 matrix. The next step for each cluster was to generate a large series of evenly spaced samples between 1 and 10 in both the x and y axis. Then we calculated the probability that each sample belonged to a particular normal distribution, using the probability density function that required the mean, covariance and a single sample as an input. The result of this was a 2x2 matrix that contained the probabilities that each sample existed within the two dimensional normal distribution of a particular cluster, we labelled this as the 'z' values for that cluster. The equation for the two dimensional normal distribution is written as:

$$f(x) = \frac{exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))}{2\pi|\Sigma|}$$

In this equation x is a 2 component column vector, which represents all the combinations of our evenly spaced samples for x and y that were generated earlier in the question. μ is the mean and Σ is the covariance, as described before. $|\Sigma|$ and Σ^{-1} are its determinant and inverse[3, pg.15-17]. The part of the equation that is raised to the exponent, $-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)$ is the formula used to calculate the Mahalanobis distance[5]. The Mahalanobis distance, can be used in multiple dimensions to calculate the distance a point is from the mean of a distribution.

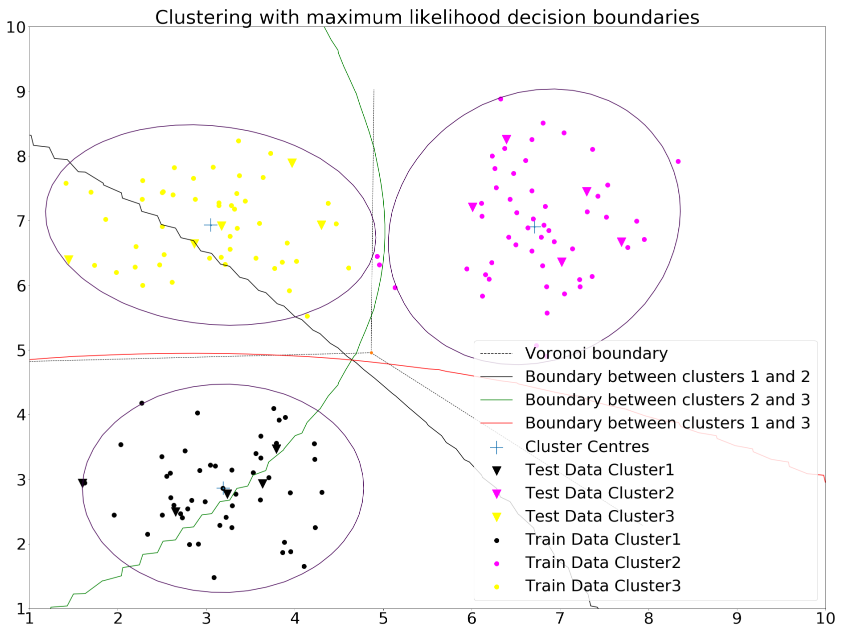


Figure 4: displays the maximum likelihood boundaries between clusters along with the nearest centroid boundaries

We then obtained the maximum likelihood decision boundary between each cluster by subtracting the 'z' values. This can been seen visibly in Figure 4, as the difference between the normal distributions were plotted as a contour, giving a non linear decision boundary between each cluster. The hyperbolic decision boundaries indicates that each cluster has different covariances.

In order to have our maximum likelihood decision boundary match our nearest centroid decision boundary all we had to do was set the covariance to be the identity matrix rather than generating it from the cluster data, this can be seen in Figure 5.

To compare between the non-linear and linear decision boundaries we plotted them on the same graph. When examining the data, it appears that the non-linear and linear decision boundaries disagree. Two points in cluster 2 of the training data lie on the cluster 3 side of the non-linear decision boundary.

The simpler nearest centroid classification seems to perform better than the more complex maximum likelihood classification, as the maximum likelihood classifier overfits the model to the data.

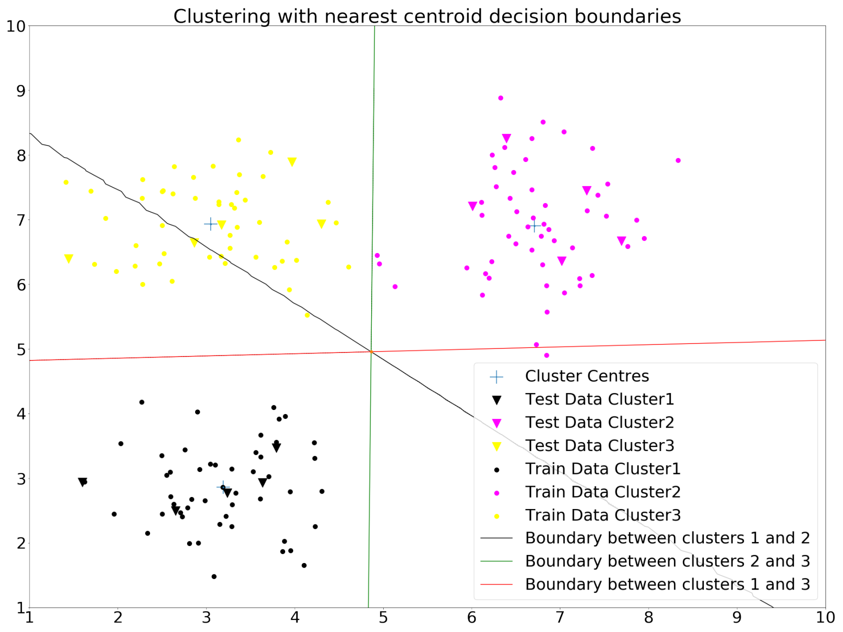
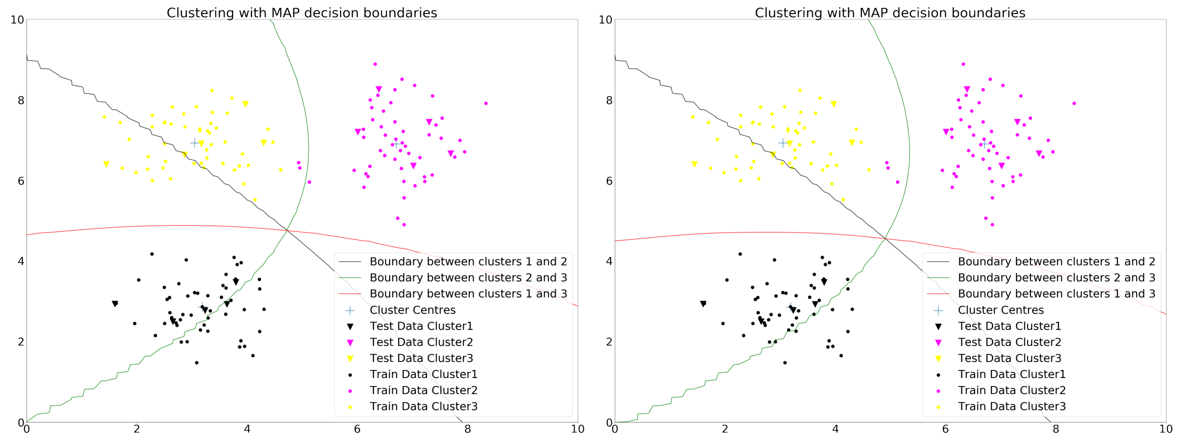


Figure 5
Displays the decision boundaries when the covariance matrix is equivalent to the identity matrix

6 Discussion of Results

On the final part of our assignment we produced a MAP estimation given a set of prior probabilities. By making one cluster twice as likely as another the impact on the decision boundaries wasn't too significant so we also decided to investigate the impact of making one cluster 10 times as likely as the other clusters with the results shown below in Figure 6.



(a) Cluster 3 is twice as likely. (b) Cluster 3 is ten times as likely.

Figure 6: MAP classifier based on prior probabilities .

As is evident in the diagram when we give cluster 3 a prior probability of twice the other cluster the shift in the decision boundary isn't too significant. However, when we make cluster 3 ten times more likely than the other clusters there is a significant shift in the decision boundary and one piece of training data that was previously in cluster 2 will now be in cluster 3.

When we modelled our clusters on a normal distribution wanted to see how many of our points were included within 95% of the probability mass for each cluster. This is demonstrated in Figure 7. Using the Chi squared distribution you can derive that this is equivalent to drawing an ellipsis with Mahalanobis distance of 6. By drawing these ellipsis we can see that cluster 2 has significantly more outliers in the data as there are 3 data points lying outside the 95% line compared with 0 for cluster 1. A future improvement of the maximum likelihood classification model could be to first check which training points lie outside of the 95% probability mass and attempt to make the model more robust to

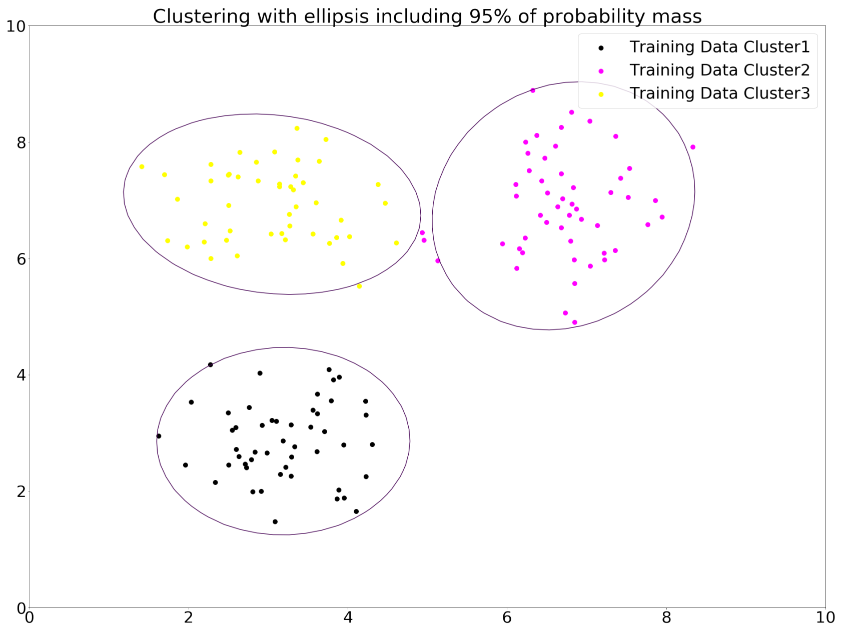


Figure 7: Shows an ellipsis containing 95% of the probability mass for each cluster

these outliers.

References

[1] *numpy.where()*, 2018. Available at https://docs.scipy.org/doc/numpy-1.14.0/reference/generated/numpy.where.html.

[2] *scipy.spatial.distance.cdist()*, 2018. Available at https://docs.scipy.org/doc/scipy-1.0.0/reference/generated/scipy.spatial.distance.cdist.html.

[3] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. Wiley-Blackwell, 2 edition, 2000.

[4] Xavier Décoret. Manhattan distance of a point and a line, 2002. Available at http://artis.imag.fr/~Xavier.Decoret/resources/maths/manhattan/html/.

[5] Stephanie Glen. What is the mahalanobis distance?, 2017. Available at http://www.statisticshowto.com/mahalanobis-distance/.

[6] Jan Schulz. Canberra distance, 2007. Available at http://www.code10.info/index.php?option=com_content&view=article&id=49%3Aarticle_canberra-distance&catid=38%3Acat_coding_algorithms_data-similarity&Itemid=57.

[7] Matthew Shardlow. An analysis of feature selection techniques. Report, University of Manchester, 2013. Available at http://syllabus.cs.manchester.ac.uk/pgt/2017/COMP61011/goodProjects/Shardlow.pdf.