**Signals, Patterns & Symbols: Coursework One**

Our coursework introduced us to the groundwork of data analysis, using clustering methods such as the k-means algorithm and classifying algorithms such as maximum likelihood and k-nearest neighbour.

Using the 150 data points provided for the 5 features we were given, we looked at how well KNN generalised as a classifier using the clustering and classification ability of k-means i.e. using the centroids obtained by k-means as data points. We will talk about why we used a cluster value of three when applying k-means to our data. We plotted a normal distribution with a specific Mahalanobis distance to act as a classifier and compared the generalising ability with a maximum likelihood classifier.

At the end of this process we compare differences between data sets given to us as a pair and look at how well each of the classifying techniques generalised on the test data given to us.

The five feature sets are compared to one another, shown in Figure 3, and 3a, using scatter plots to compare correlation between features, and histograms to describe the make up of the data. We know classes are groups that share similar features. We look for two features that show visually separate groupings of data when plotted against each other. Features 1 and 2, shown in row 1 column 2, and column 1 row 2 of Figure 3, along with features 1 and 4 in Figure 3a, clearly split the data points into 3 distinct groups, with each group appearing to be normally distributed.

The histograms for features 1 and 2 for one partner & 1 and 4 for the other suggest that each one separates the data points into two separate classes on their own. The histograms show an uneven distribution of data with most frequent points occurring towards the upper and lower values, and very low frequency in central values. Histograms for five features for both of us appear to show two normal distributions, which would correspond to the two classes the feature distinguishes.

For example, the histogram for feature one, shown in row 1 column 1 of Figure 3, implies two classes, one with low levels of the feature at around 2.5, and one with high levels of the feature at around 7.5, with a distinct lack of frequency for central values around 5. This is also the case for feature two of Figure 3, shown in row 2 column 2.

Let’s imagine the classes to be separated are fish, say Trout, Bass, and Pancake batfish. Feature one may be the width of the fish, and feature two the length. Feature one separates Pancake batfish from the other two classes because it is wider, but does not separate Bass and trout which have the same width. Feature 2 separates Bass as it is longer than the other two, but does not separate trout from pancake batfish, as they have the same length. When these two features are combined in the scatter plot in row 1 column 2, we see a clear separation between the 3 classes, based on the width and length.

We therefore selected features 1 and 4 as well as 1 and 2. We chose the number of classes to be 3.

We do not choose the features that generate roughly uniform histograms as they do not help to distinguish classes. For instance, in Figure 3, Features 3 and 4, both have roughly uniform distributions. Plotting the scatter graph of these two in figure 3, shown in row 3 column 4, displays an even spread of data point, with no clear groupings of data.

After choosing the number of classes equal to three, we ran k-means.

K-means works by randomly initialising a set of centroid locations. It then iterates through all data points and decides which centroid is the data point closest to. For each centroid it calculates the mean of its assigned group and moves there. The process restarts until final centroid locations are found, the process is analogous to balls rolling into a valley. K-means expects as parameter the number of clusters. K-means terminates when there is minimal change in centroid locations. We chose the number of clusters equal to three as mentioned in feature selection stage. The results can be seen in fig. 1 & 1a as the centroids as black x’s and the coloured x’s around the black x’s as the clustered points.

In order to create a situation whereby k-means incorrectly classifies clusters (shown in figure 2 & 2a) we identified two distinct weak points in the algorithm. One of them was to change the maximum number of iterations of the algorithm. The subjective threshold at which k-means halts is a weakness in the algorithm as if it is too big, there can be an incorrect classification as the means have not yet converged. The other method which is pictured in figure 2 & 2a is the result of initialising two of the centroids at a small distance from the data set i.e. the range of the data set in both the x and the y dimensions was approx. 10 so these two centroids were at (2,6) (1,5) respectively and so we placed the final centroid at a euclidean distance of (10000,10000). This meant that the algorithm moves the cluster placed far away with no members closer to one of the other clusters, after the other clusters have already found their local optima. For instance in both figures 2 & 2a, one of the clusters is halfway between the two clusters as it is located at the mean of its classification and closest to all points in its set. This meant that a local optima was found. The within cluster sums are now equal to the mean at the halfway point. The voronoi tessellation in the broken version of k-means reflects the centroid locations in the incorrect result.

We then used the k-nearest neighbour algorithm using the centroids given to classify any future test data points. The Voronoi tessellation plotted in figure 1 & 1a shows the decision boundaries of the k-nearest neighbour classifier with k = 1. The reason we do not re-use k-means once we have new test data is that we could end up overfitting. The cardinality of the test data was 10 percent of the training data, which seems to be reasonable as a way of measuring the success of our classifiers.

The k-nearest neighbour seemed to fit the data very well, not appearing to overfit or underfit the data given. On the test data it correctly classifies all data points, with a great deal of space between any data point and the decision boundary. It also only uses the centroids obtained by the training data as parameters, so it does not use the test data and therefore does not overfit. The decision boundary can be shown by the thin straight blue line which divides into 3 in the centre in Figure 1 & 1a.

Modelling the data in each class as being generated from a 2-D Normal Distribution, we estimated the means and covariance’s from the training data. To visualise a confidence interval and potentially a classifier that contains 95% of the data, we plotted an ellipse at a squared mahalanobis distance of 6 (or 5.99). Mahalanobis distance is a measure of the distance between a point P and a distribution D and takes into account correlation, it is here a 2 dimensional equivalent of saying that in 1 dimensional space, 95% of the Gaussian probability mass is within +/- 1.96 standard deviations from the mean.

All of the test data points were contained correctly within their appropriate ellipses, so the confidence interval has performed well and neither overfitted or underfitted on the test data. The selection of an interval containing 95% of the probability mass is somewhat arbitrary; we could have selected a 99% confidence interval, which would have fewer points beyond the boundaries. Selecting a confidence interval of 100% would be inappropriate, as this would then include every point of the data set, including all the points belonging to other classes and potentially overfitting the training data, however, we would need a larger test set to really answer that question. In the green cluster in figure 1 only one of the test points is outside the ellipse, however there seems to be a relatively high variance in this cluster and many training points are also outside the ellipse, which indicates the ellipse is generalising well.

Another technique we used to classify data was the maximum likelihood decision boundary. This involves taking each of the data points within a class and calculating their probability given the respective class covariances and class means. We then plot a decision boundary representing the line where the likelihoods of each class are equal i.e. one divided by the other gives one. This is a continuous line even with a finite set of discrete points as MatLab interpolates and finds the value where the decision boundary is exactly one.

In figure one, the decision boundary between the blue and green clusters is almost a perpendicular bisector of the line connecting the means. This is because they have roughly equal covariance matrices. We do not have any non contiguous decision boundaries even though the variances of the classes differ, as the diagram is not big enough. If we extended the diagram, there would most likely be a secondary point of intersection and therefore a secondary decision boundary. This decision boundary is very effective at classifying data points as all test points are included in the boundary.

Maximum likelihood will lead to a basic linear classifier whose decision boundary is orthogonal to the line connecting the means of data groups, if the features are uncorrelated and have the same variance. Therefore To achieve the same linear decision boundaries as nearest centroid classification, the covariance matrices would have to be made equal.  
  
Should one of the classes be known to be twice as likely to occur as another, we would calculate the maximum likelihood with this knowledge as a prior, in this case multiplying the class probability by this maximum a posteriori. This effect is shown in figure 4a, where class one is twice as likely, this causes the decision boundaries between cluster 1 and cluster’s 2 and 3 to change be tighter around cluster one, improving the accuracy of the ML classifier, meaning a misclassification of a data point as cluster one is less likely to occur.

In order to conclude, overall the data sets for both students are particularly similar in structure i.e. the number of classes and the structure of the features and how they separate the classes. The classification methods we used correctly classified all of the test data sets and so were equally proficient. It is interesting to observe that the non-parametric method, using K nearest centroids and Kmeans, performed just as well as the parametric maximum likelihood (ML) classifier, given the latter uses information about the underlying probability density function, a Gaussian model that is not available to the non-parametric K nearest centroids model. We would expect that the ML classifier would be a better classifier because of this extra information, the equality of effectiveness may be due to the relatively small data set used, and a larger data set might help distinguish these classifiers. Knowledge of a reliable prior to apply to the ML would improve its accuracy, and could also make it a more effective method of classification.

Alternatively, we could have employed other classification techniques, such as K nearest neighbours, with K equal to some number such as 10, classifying points based on the proximity to data points classified in the training set, as opposed to K nearest centroids and the Voronoi tessellation. This would improve accuracy, but would suffer from high space and time complexities, since we have to compute the distance between a given feature and samples in our training set each time we perform a classification.

All work is original and as such, we see no need to display additional sources.











