1 Introduction

This report aims to demonstrate and explain how the given data sets were clustered and classified into the given plots. The report will cover the various required methods to achieve this and incorporates two sets of data, each with 5 distinct attribute values. The two data sets will be referred to as Data E and Data L, and are each composed of training data and test data.

2 Feature selection

Using the 5 attribute values for each data set, 2D scatter plots were produced with an attribute on each axis, with the aim of identifying the pair of attributes which best separates the classes of the training data. The ideal feature to be observed in a suitable plot is a clear separation between different groups of data points. There were three characteristic plots that were observed:

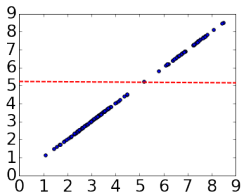

Figure 1: *A linear plot of an attribute against itself. Red line indicates separation*

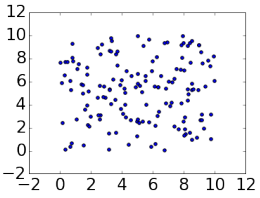

Figure 2: *A plot with no distinguishable separation between its data points*

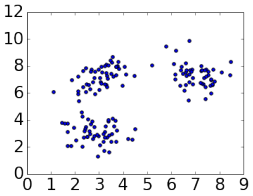

Figure 3: *A plot with clear separation between its data points*

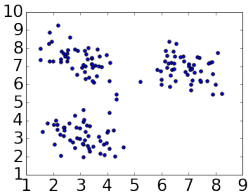

Figure 4: *Clear separation. The selected plot for Data E*

Although plotting an attribute against itself may appear unnecessary, it helps to reveal separation, if any, between data points within a single attribute, as seen in Figure 1. This quality means it is easy to identify attributes which are good candidates for selection. Figure 2 indicates a poor candidate as it is unclear how the data would be partitioned into groups. Figure 3 and Figure 4 show the selected attributes for Data L and Data E respectively as they present the clearest and strongest separation of all attribute pairs; shown by the 3 distinct clusters. It is important to note that for each cluster, the density of data points tends to be higher in the central area of each cluster. This type of density indicates that each cluster could be modeled by a normal distribution along each axis; this is a powerful trait as Maximum-Likelihood Classification revolves around this idea. Attributes 3 and 4 were chosen for Data L and attributes 4 and 5 were chosen for Data E.

3 Identifying the classes

Data L will use attribute 3 and 4 for the x and y-axis respectively and Data E will use attribute 4 and 5 for the x and y-axis respectively. With the attributes identified, it is necessary to define a set of classes based on the number of clusters. To identify classes in the training data we used the K-Means algorithm where k represents the number of clusters. In this case, the number of clusters is 3 for both data sets, and for simplicity we will use colours as class labels: red, green and blue. The K-Means algorithm can be broken down into several steps:

- Initialisation: Means, μ_c, are initialised for each of the k clusters. There are multiple methods for this, but the one used in this report is k-means++ which involves a heuristic to speed up convergence[1].
- Assignment: For the set of all data points, $d_i \in D$, and the set of all classes, $c \in C$, there exists a set of means, $\mu_c \in \mu$. All points are assigned to a class such that:

$$c \subset D$$

$$dist(d_i, \mu_c) = |d_i - \mu_c|$$

$$d_i \in \{c | argmin_c\{dist(d_i, \mu_c)\} \forall c \in C\}$$

- Adjustment: Once all points have been assigned, the mean of the classes must be adjusted accordingly such that:

$$\mu'_c = \sum_{d_i \in c} \frac{d_i}{|c|}$$

  Assignment and Adjustment are repeated over a number of iterations to reach appropriate approximations of a class' mean.

In short, given some initial mean for each class, each data point is assigned to the class with the minimum euclidean distance between the data point and that class' mean. Figure 5 and 6 show the resulting K-Means plot of Data L and Data E respectively, with the final mean of each class indicated as a cross. Visually, every point is assigned to the cluster that would be expected.

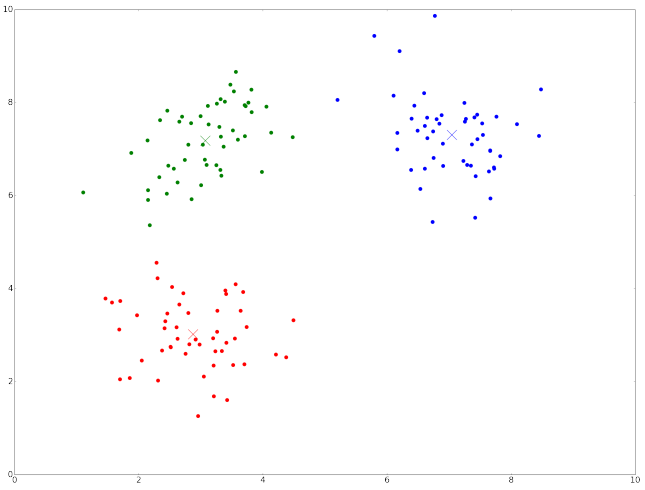

Figure 5: *K-Means clustering for Data L*

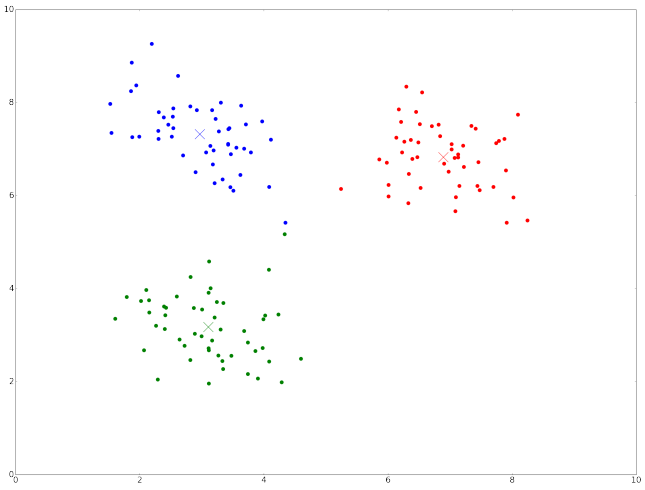

Figure 6: *K-Means clustering for Data E*

3.1 Suboptimal Clustering

The default Numpy function in fact applies this k-means algorithm ten times, with intelligent seeding for the means and 300 iterations of improvement. To find a non-optimal clustering for Data E, we reversed this, finding the worst fit out of 100 runs of the algorithm, and also changing the k-means++ to just random initialisation. This was not enough for Data L however, as the clustering was too tight (TODO?), so the 300 iterations had to be reduced to just 5 iterations of the k-means algorithm. Any more and the clustering would become optimal.

4 Nearest-centroid classification

Using the centroids (the means) obtained from k-means, the test data can be classified using a simple nearest neighbor approach where the closest centroid in terms of euclidean distance determines a data points' class. On Figure 7 and 8 the test data is plotted as diamonds and linear decision boundaries are displayed as dashed lines. The decision boundaries are essentially the loci of each centroid pair.

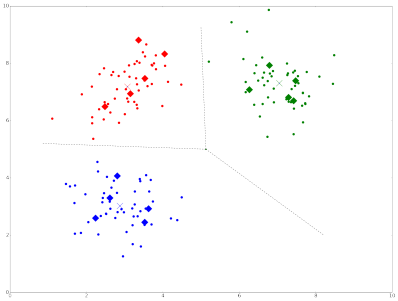

Figure 7: *Nearest-centroid classification for Data L*

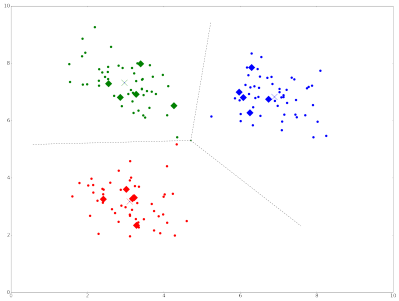

Figure 8: *Nearest-centroid classification for Data E*

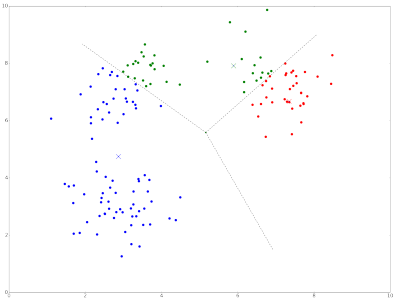

Figure 9: *Sub-optimal k-means clustering for Data L*

Nearest centroid classification is very dependent on the result of a clustering algorithm, such as the k-means algorithm, as it requires *distance* to be an accurate representation or measurement of a class. If it is not, then nearest centroid will produce inaccurate results. For example, the k-means algorithm normally requires numerous iterations to achieve a *reliable* result for typical data. Given a non-optimal k-means algorithm, such as one with a very small number of iterations or one which runs from scratch multiple times in order to find the poorest result, then the centroids returned will lead to poor classification. Figure 9 indicates the result of the latter, where a k-means algorithm runs from scratch to maximise distance as opposed to minimise distance. The decision boundaries intersect the visual clusters in Data L and prove to be suboptimal.

5 Maximum-likelihood classification

As mentioned earlier, the clusters mimic the shape of a Gaussian distribution. Each cluster can individually be viewed as a Gaussian distribution in two dimensions. Graphically, this represents an elliptical area of varying probability density. This view makes the assumption that all observations are completely independent. They are said to be independent and identically distributed. The probability density of a Multivariate Gaussian distribution can be modeled by the following:

$$pdf(d) = \frac{1}{\sqrt{|2\pi\Sigma|}} e^{-\frac{1}{2}(d-\mu)^T\Sigma^{-1}(d-\mu)}$$

Σ represents the covariance matrix for the distribution. To obtain a contour with 95% of the probability mass within the ellipse, it is given that for a bivariate distribution, the points on the ellipse satisfy: $(d-\mu)^T\Sigma^{-1}(d-\mu)/2 = 3$. This can be substituted into the equation above to give the following:

$$\frac{1}{\sqrt{|2\pi\Sigma|}} e^{-3}$$

which in turn allows calculation of the required points to plot the contour. Figure 10 and 11 show the contours plotted using this formula.

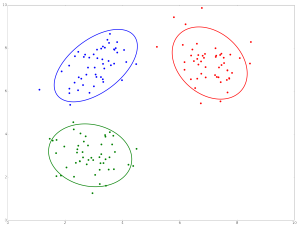

Figure 10: *95% probability mass contour for Data L*

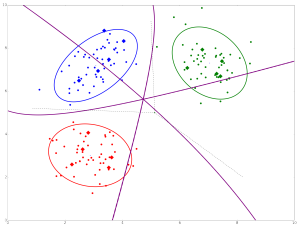

Figure 11: *Contour and decision boundaries for Data L*

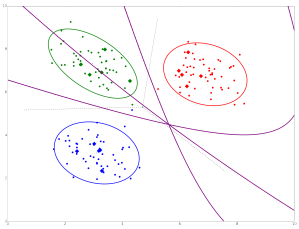

Figure 12: *Contour and decision boundaries for Data E.*

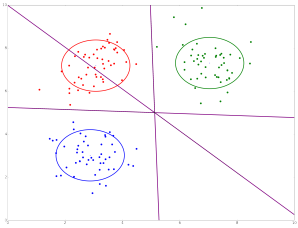

Figure 13: *Linear decision boundaries for Data L.*

Visually you can see that around 95% of data points are present within the contours. To plot the decision boundaries we effectively divided the probability density functions by each other in pairs; these boundaries are relatively similar to the linear boundaries given by Nearest-centroid classification. Figure 11 and 12 show these boundaries for Data L and Data E respectively; diamonds are plotted where test data has been classified using the decision boundaries, and the Voronoi plot has been included for comparison. Visually and comparatively, the classification results were the same as Nearest-centroid for the test data.

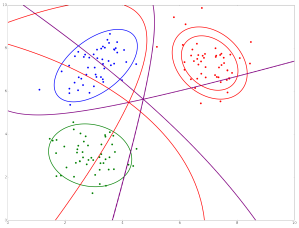

Figure 14: *Twice-as-likely weighted decision boundaries.*

It is possible to achieve linear decision boundaries, such as those given by Nearest-centroid, from Maximum Likelihood classification by using an identity matrix as the covariance matrix. This presents a scenario where the covariance is 0 and the random variables appear independent; the result is illustrated in figure 13. Furthermore, it is possible to adjust the decision boundaries to reflect a known change in probability: figure 14 illustrates the effect of making the red class twice as probable as the other two classes; this was achieved through multiplying that class's covariance matrix by 2. The contour of the bias class is enlarged and the decision boundaries are more heavily weighted towards the mean of the biased class for Data L.

6 Discussion of results

Both Data E and L classified as expected and had matching classifications from both methods. This is potentially due to the size of the test data in relation to the training data; if larger test data sets, or smaller training sets had been used then there would most likely be a greater mismatch in classification between MLE and Nearest Centroid. Data E had more spread that Data L, which is evident when looking at the training data near the decision boundaries, where some points are on different sides of the different boundaries.

7 Sources used

[1] `http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`

`http://matplotlib.org/examples/pylab_examples/`

`https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.multivariate_normal.html`

`https://www.python.org/doc/`