Symbols, Patterns and Signals Report

Eben Tuff (et15792) and Faizaan Sakib (ss16161)

March 19, 2018

1 Introduction

Given a set of training data consisting of 150 data points (each a feature vector with 5 features) and a testing data set with just 15 points (also with 5 features) the task was to compare 2 different methods of classifying data which help to identify relationships and patterns within the data. These feature vectors could, for example, represent fish with different species within. The 2 methods used were: nearest-centroid and maximum-likelihood classification. These methods were used to sufficiently *train* a classifier to correctly separate the data into classes with similar features, providing meaning to previously unlabeled data. Continuing with the fish analogy, this could be seen as classifying the data points into their respective species of fish.

2 Feature Selection

The data values had five different features, it was important to reduce this number, and subsequently the number of dimensions worked in, or else problems may have occurred. With multiple features and an unknown number of classes that the data points could belong to, it could make it harder to find valid relationships within the data as some of the features may be irrelevant, so their distance calculations could skew the classification results produced.

The first step was to look at the data. Each feature was plotted against all of the features, providing a 5x5 matrix of scatter graphs. These were then inspected, by eye, to decide which features should be used to provide the maximum amount of clusters. These clusters would go on to become the classes to which the data points would be classified.

5 of these graphs were redundant as the feature chosen was being compared to itself. Another 10 could be discounted as they were simply mirrors of the other 10. The remaining graphs had varying degrees of use towards the task. Some graphs, as visible in Figure 1, showed no clusters and were not considered useful towards the task. Other comparisons produced 2 clusters, either along the *x-axis* or *y-axis*, and seemed to be good solutions. The optimal feature pair was features 1 and 3 which produced 3 clusters when plotted against each other.

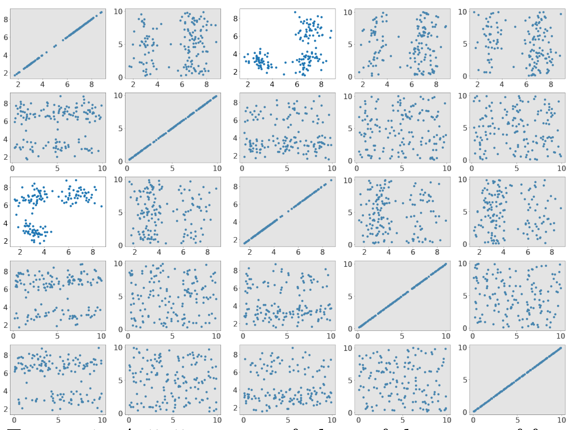


Figure 1: A 5x5 matrix of plots of the pairs of features. Graphs of features 1 and 3, plotted against each other, are highlighted

3 Class Identification

The next step was to label the training data in terms of classes. With the presence of 3 seemingly separated groups of data, the instance space could be fragmented into regions classifying the data within it. To do this, the K-Means algorithm was utilized to model the data set deterministically to set 3 clusters. This made the assumption that the relationships within the data were fixed, with no consideration given to any possible randomness or uncertainty.

The algorithm first initialises, in this case, 3 centroid-estimate vectors across the instance space. This is followed by a repeated loop of 2 instructions: (i) clusters are allocated by comparing the distances of each point to each of the centroids and assigning it to the closest one. (ii) each of the centroids is recalculated as the mean of all the instances assigned to the respective cluster, moving the centroids to a new location, paving the way for the next iteration of the loop. Instances may or may not still be assigned to the same cluster due to the new location of the centroids, so all the data points are re-evaluated.

The K-Means algorithm also seeks to minimise the sum of the squared distance magnitude of all the points to each of the clusters.

$$\Sigma_i ||x_i - \mu_k||^2 \tag{1}$$

This is known as the within-cluster scatter that is summed over all k clusters. The algorithm iterates and refines the positions of the centroids until the scatter is reduced as much as it can be to find the optimal configuration of clustering. The algorithm clustered the data points and assigned a class to each of them, colour-coded, as plotted in Figure 2, while also returning the scatter of each of the clusters and the cluster centroids which proved important in the next step.

4 Nearest-Centroid Classification

With the centres for each cluster realised, the test data was used to see if the classifier could correctly categorise any new data points.

For consistency, the same features from the training data had to be extracted from the test data. Then the Euclidean distance was calculated from each test data point to each of the 3 cluster centroids. With only 2 dimensions within the instance space, and not much importance given to the direction of the distances, the use of Euclidean distance would suffice. The class that each point would be assigned to was determined by which of the centroids it was closest to. Adding a Voronoi diagram, according to the cluster centres, showed that both the training and test data were divided as expected, according to the K-Means results.

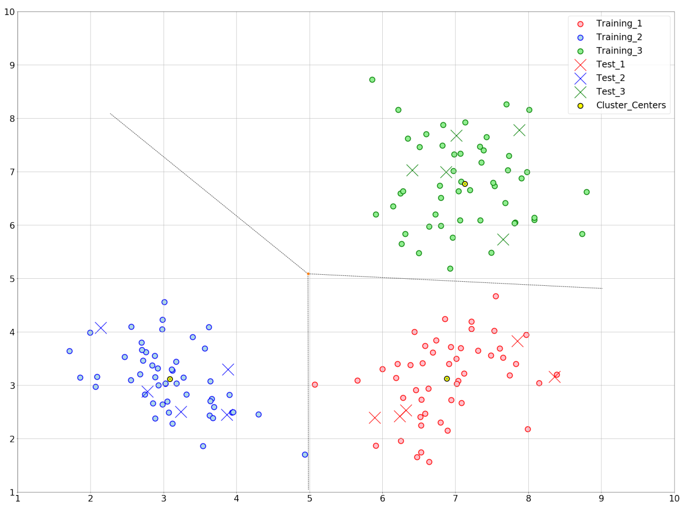


Figure 2: A graph showing the classified points of the training and test data sets, along with a Voronoi diagram, according to the cluster centroids produced by the K-Means algorithm.

Although the classifier created achieved the desired outcome, an important assumption is made. As the Voronoi shows, there is some area, within the instance space, which is empty due to the lack of training data. A new point in an area further away from the clusters, for example, on the top left of the instance space, would be classified since it would simply be compulsorily assigned to the nearest cluster centroid. However, this can be classed as extrapolation as there may be data the classifier has not been trained with yet which could indicate the possibility of the area in question to be classified in another cluster, or perhaps, one that has not even been known to exist yet.

The K-Means algorithm can be applied in a variety of different ways by setting constraints. One such constraint put on the K-Means algorithm, which was applied to the training data set, made the algorithm run for 300 iterations before halting. It could have stopped before this, subject to the convergence being realised before the 300th iteration. Another constraint in place stated that for the algorithm to have successfully converged, the change in all of the centroid values between the current and previous iteration would have to be less than 0.0001.

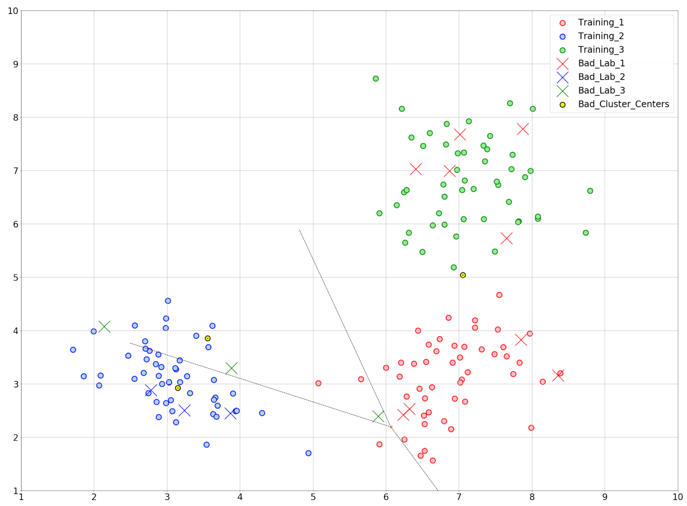


Figure 3: A graph showing the result of a sub-optimal clustering produced by an altered version of the K-Means algorithm.

However, the main characteristics of the K-Means algorithm used on this data set were two things. The method of centroid initialisation used and the number of times the algorithm is ran using different centroid seeds. These are what makes the K-Means algorithm so efficient and one of the fastest clustering algorithms available [1]. This can be proved by calculating new cluster centroids using the K-Means algorithm with initialisation parameters different to the ones used before.

Instead of using the K-Means++ initialisation technique, which is an optimised method of centroid initialisation [2], random initialisation was used instead. Furthermore, the algorithm was used on only a single set of centroid seeds it produced. As a result, the algorithm produced a configuration with abnormally high within-cluster scatter values and sub-optimal cluster centroids, which can be seen on Figure 2.

It is obvious to see that the data has been incorrectly classified, with the Voronoi finding itself incoherently placed compared to what it should be as seen before. This shows an importance in the initial cluster centres used in this algorithm, as with poor cluster centres at the start can lead the algorithm to converging to local minima rather than the global minima.

5 Maximum-Likelihood Classification

The second classification method used was maximum-likelihood classification (MLC). This method uses probabilities and ratios of maximum-likelihood estimates (MLE) to draw decision boundaries between data. To use this, the data had to be modeled as being generated from a 2-D normal distribution. This was achieved by disregarding the idea of the data being coordinates and, instead, imagining it in terms of probability densities.

First the mean (or, in terms of clusters, its centroid value) and covariance matrix (the spread and orientation) were calculated for each class, using the following equations respectively:

$$\boldsymbol{\mu} = \frac{1}{n}\sum_i \boldsymbol{v_i} \qquad \boldsymbol{\Sigma} = 1/N\sum_i (\boldsymbol{v_i} - \boldsymbol{\mu})(\boldsymbol{v_i} - \boldsymbol{\mu})^T$$

With these values, the distributions could then be visualized (as seen in Figure 4) by creating contour plots of each class's probability density function (PDF) - the likelihood that a point x would be drawn from the distribution.

$$p(x|\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}}\sqrt{det(\Sigma)}}exp(-\frac{(x-\mu)^T\Sigma^{-1}(x-\mu)}{2})$$

As the data is 2-dimensional, d is equal to 2.

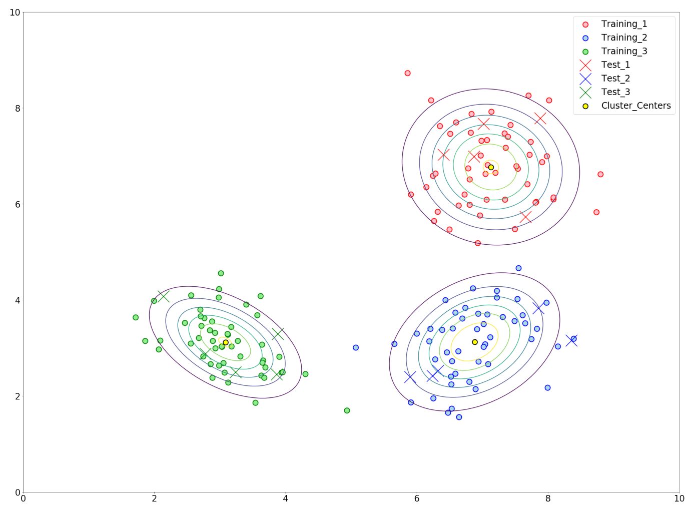


Figure 4: A visualization of the data modeled as being normally distributed.

Next, the goal was to classify the data with 95% confidence. This was visualized by creating a more concise plot with only 1 contour level encapsulating 95% of the probability mass (NB: this is *not* the same as containing 95% of the data points). Comparing this to nearest-centroid classification, this is similar to plotting the points, applying K-Means and classifying the data.

The sums of the squared gaussians follow a chi-squared distribution. This allowed the inverse of the chi-squared cumulative distribution to be used, with a percentage of 95% and 2 degrees of freedom, to obtain the distance that points must be away from the centroids of the clusters to be within the 95% threshold: 5.9915. What type of distance? Since this is not Euclidean space (i.e. an x-y *axis*), the 5.9915 cannot represent a Euclidean distance. The clusters, being in multivariate (or more specifically, bivariate) space, required distances to be measured as Mahalanobis distances, or in this specific case, squared Mahalanobis distances. The equation to find the Mahalanobis distance is $d_M = (x-\mu)^T\Sigma^{-1}(x-\mu)$.

Recall the PDF function used when visualizing the clusters:

$$p(x|\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}}\sqrt{det(\Sigma)}} exp(-\frac{(x-\mu)^T\Sigma^{-1}(x-\mu)}{2}) \tag{2}$$

where μ and Σ represent the clusters' means (centroids) and covariance matrices (spreads) respectively.

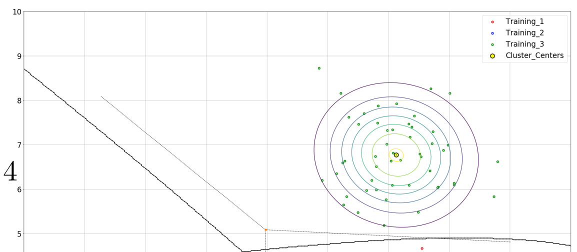
Notice, now, that in the exp part of the equation, the numerator is $(x-\mu)^T\Sigma^{-1}(x-\mu)$; the Mahalanobis distance. Since this must equal 5.9915, the value can be substituted back into (3):

$$p(x|\mu,\Sigma) = \frac{1}{2\pi\sqrt{det(\Sigma)}} exp(-2.99575) \tag{3}$$

Finally, substituting Σ (the μ becomes redundant as it's no longer in use on the right-hand side) for each of the 3 classes' covariance matrices yielded 3 probability values. These values represented the probability necessary to be on the 95% threshold.

With these 95 % confident classifications estimated, similarly to nearest-centroid classification, the next step was to plot decision boundaries. This was achieved by, plotting contours of the likelihood ratios. These are visible in Figure 6, below.

Unlike the Voronoi diagrams in nearest-centroid, the decision boundaries for MLC are curved. The lines plotted represent where the ratios between classes are equal;



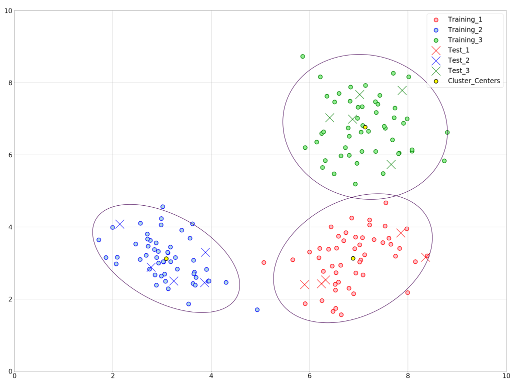


Figure 5: A graph showing cluster-ellipses that contained 95% of the probability mass.

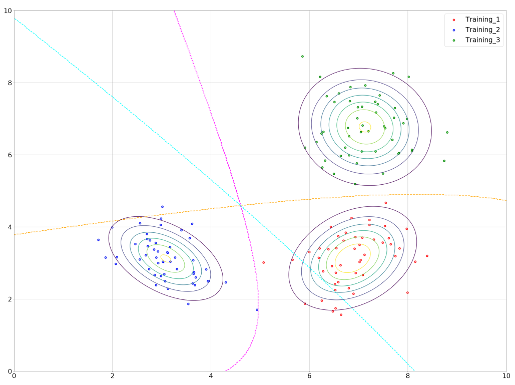


Figure 6: A graph showing rough MLE decision boundaries alongside PDF contours

in other words, where a point x is equally likely to be drawn from either distribution. With this, they follow the covariance matrix ellipses; the most prominent example being the magenta curve (in Figure 6) which follows Class 2's ellipse.

Notice that the cyan line appears linear. This is due to Class 2 and Class 3 having the same orientation. Intuitively, the decision boundary would be a translated (based on the width of the ellipses) equidistant line from the centroids and, hence, linear.

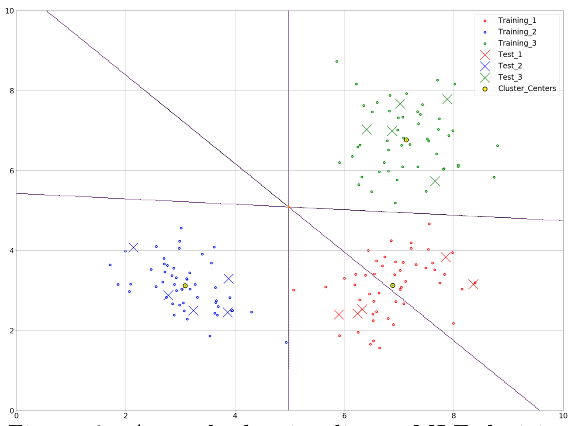


Figure 8: A graph showing linear MLE decision boundary estimates.

The K-Means Voronoi was generated by considering the point-to-centroid distance from all 3 classes, whereas the MLE boundaries only considered 2 classes at a time. With this, a new graph was created where plotted points represent where the ratio of 1 distribution's likelihoods are greater than the other 2.

The Voronoi was plotted on top of the this, as shown in Figure 7. It is evident that there is a significant variation in the 2. This could be due to MLE **overfitting** the boundaries (where they are plotted close to point-wise) which is evidenced by the highlighted point in Figure 7.

To alleviate this issue of over-fitting, the PDFs were recalculated with Σ set to the unit matrix, $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, likelihood ratios taken, as before, and the contours plotted, presenting the graph in Figure 8. The boundaries are now linear - but why? Recall earlier, in Figure 6, how the cyan boundary was linear due to 2 of the clusters having the same orientation. By replacing the clusters' covariance matrices with the identity matrix, the ellipses become regular circles, with equal variance. With this, all classes have the same orientation and thus the boundaries become linear.

The last step of using the MLC was, obviously, to add the testing data points to see if it worked. As the result was identical to that produced via nearest-cluster, the testing data was, indeed, correctly classified and is visible in Figure 8.

6 Discussion of Results

The data as a whole was classified as expected, besides one training point (highlighted in Figure 7), that was classified incorrectly into Class 3 instead of Class 2, under the assumption that each class had 50 points. As its distance from Class 3's centroid was the least, K-Means classified it incorrectly due to being a deterministic modeling algorithm which doesn't take outliers into account. As a result, both the nearest-centroid and MLE classifiers failed to correctly include the point into Class 2. This is most visible in Figure 7, where the MLE decision boundary has a significant bend to accommodate this outlier.

Looking at the test data, they were all assigned to the correct classes, well within the plotted decision boundaries. It is worth noting, though, that each of the test data points were relatively central to each of their respective clusters, meaning that the testing data didn't stress-test the classifiers to back-up the previous point.

In Figure 5, the 95% ellipses overlap. Any points within this region would have PDFs of 0.5 for Class 1 and Class 3. The yellow line shown in Figure 6 would bisect this region perfectly.

A final point to make is that the centre of the decision boundaries made from the likelihood ratios MLE is significantly different to that created with the unit covariance matrix and also generated by K-Means (Figure 7). As the data points in Class 2 have a higher density, and thus a smaller distribution, the standard deviation is smaller. This means that as you get further away from Class 2's centroid, the PDF will rapidly decrease. For the other 2, more spread out classes, however, their PDF remains larger per distance away. Hence the boundary has to be plotted closer to the smaller cluster, skewing the centre.

This skewing is not present in Figure 8 as all the covariance matrices, hence the distributions, were equal.

7 Sources Used

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

[2] David Arthur and Sergei Vassilvitskii. *k-means++ : The Advantages of Careful Seeding.* Stanford, 2006.