Lab 1: Clusters and Classification Boundaries

A Report Submitted in Partial Fulfillment
of the Requirements for SYDE 372

Manthan Shah, 20658832
Austin Atmaja, 20658965
David Zhang, 20672157
Moujan Saderi, 20654126
Heather D'Souza, 20671903

Faculty of Engineering
Department of Systems Design Engineering

February 12, 2019
Course Instructor: Prof. A. Wong

1 Introduction

Pattern recognition involves the age old problem of assigning unknown patterns to a known class. Logically a pattern should be assigned to the most similar class. The question then become what defines similarity and how can prototypes be chosen for each class to make the correct decision. To answer this question, five different classifiers are developed and tested on correlated, normally distributed data. The process involves generating clusters of this data, creating unit standard deviation contours, developing the classifiers, testing how the classifiers perform when given two to three classes, and conducting error analysis. The following includes detailed descriptions of each step of the process, the MATLAB code used to conduct each step, and the results of these findings.

2 Generating Clusters

Random samples were created using the `randn` function in MATLAB with a default mean of 0 and variance of 1. A cholesky factorization was then performed to diagonalize the covariance matrices using the `chol` function. Thereafter, a normal distribution was developed for each class by generating a matrix of means replicated N times (N = samples), and then summed with the random samples multiplied by the cholesky transform.

A scatter plot of the data was generated for all the classes, found in figures 1 and 2. The unit standard deviation contour ellipses were composed of square roots of the eigenvalues for the horizontal and vertical axis components respectively. The angle of the ellipses was determined by taking the inverse tangent of the eigenvectors.

2.1 Case 1

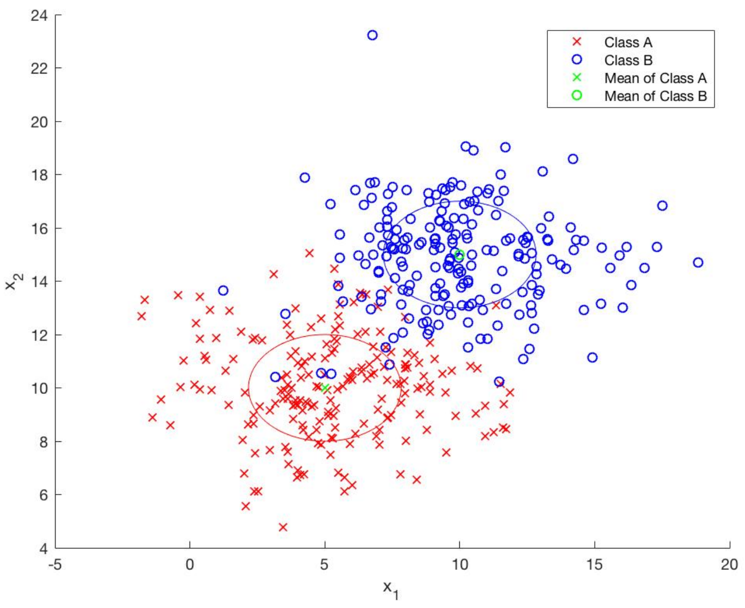

Figure 1: unit standard deviation contours for class A and class B

As class A and B are both uncorrelated based on the non-diagonal zero values on the covariance matrix, both unit standard deviation contours are un-rotated at 0 degrees. However, it is observed that the covariance matrix values along the diagonal are not the same, thus the contours are not a perfect circle - the width is greater than the height (a > c).

2.2 Case 2

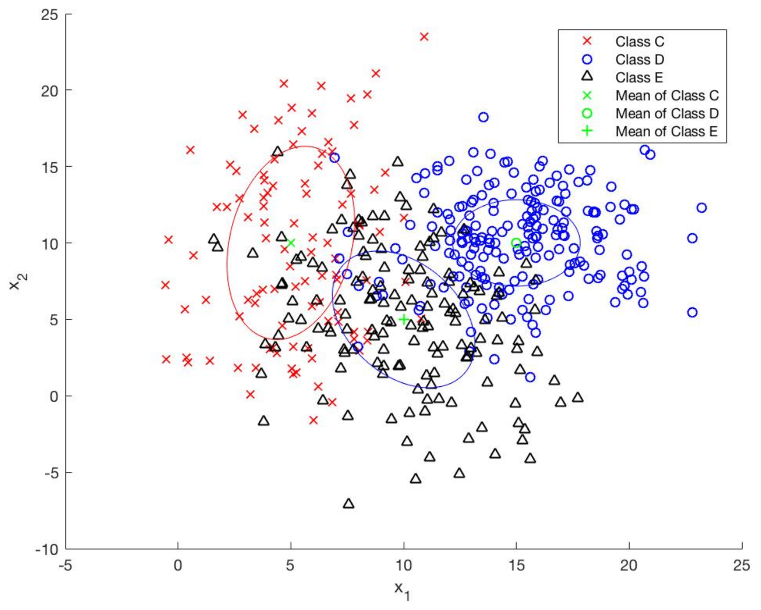

Figure 2: unit standard deviation contours for classes C, D, and E

The unit standard deviation contour for class C is rotated approximately 83 degrees counter-clockwise as the data is correlated - based on the non-diagonal values which are non-zero. The shape of the ellipse is not circular as the variances are not the same, indicated by the greater height parameter than the width (c > a).

The unit standard deviation contour for class D is horizontal at 0 degrees - this is due to uncorrelatedness in the data, marked by zero valued diagonal elements. The shape of the ellipse is not circular as the variances are not the same, indicated by the greater width than height (a > c).

The unit standard deviation contour for class E is rotated approximately 68 degrees clockwise as the data is correlated - based on the non-diagonal values which are non-zero. The shape of the ellipse is not circular as the variances are not the same, indicated by the greater width parameter than the height (a > c).

3 Classifiers

To develop decision boundaries for different types of classifiers, a matrix of points for x_1 and x_2 directions were generated ranging from the smallest value to the largest value based on the matrices of samples among the classes for each case. These are separated by a step size of 0.2. These points are then used as input for the `meshgrid` function, which outputs a 2D grid of points for use in each of the classifier equations. Each classifier function's output is a matrix of points for a contour plot defining the decision boundary. The decision boundaries applied on each case are found below in figures 3 and 4, where the MAP and GED boundaries lie on top of each other in figure 3.

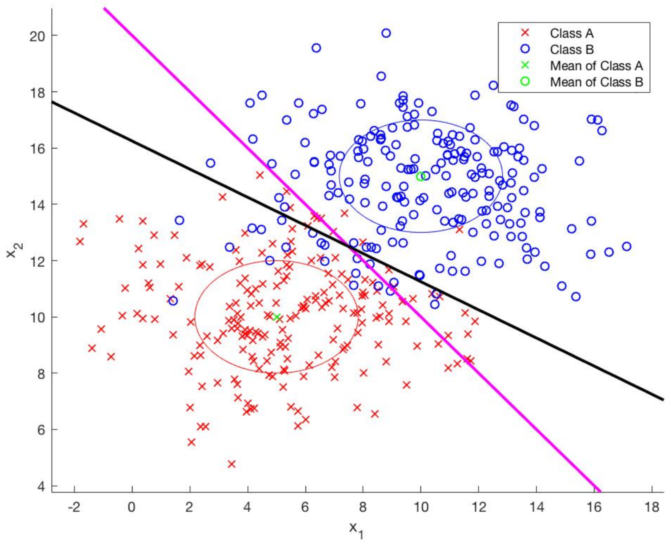

Figure 3: MED, GED, and MAP decision boundaries for classes A and B (case 1)

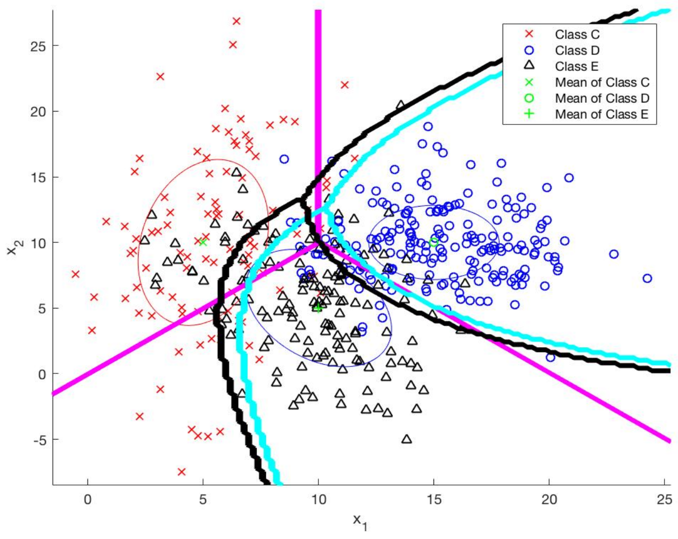

Figure 4: MED, GED, and MAP decision boundaries for classes C, D, and E (case 2)

3.1 Minimum Euclidean Distance (MED)

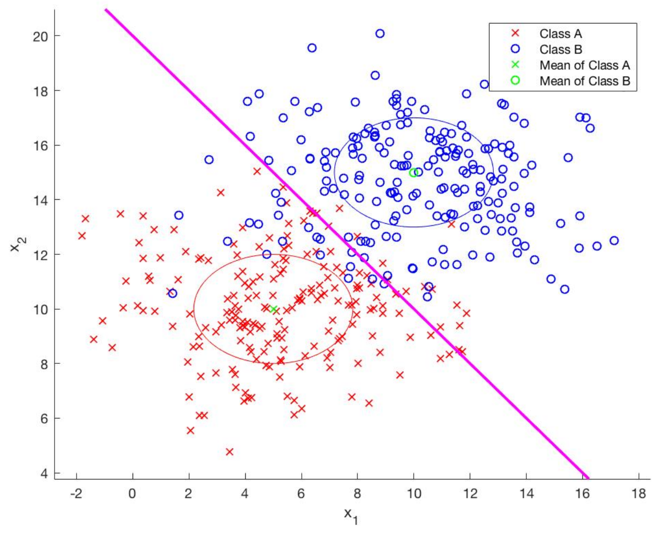

Figure 5: MED decision boundary applied on classes A and B clusters (case 1)

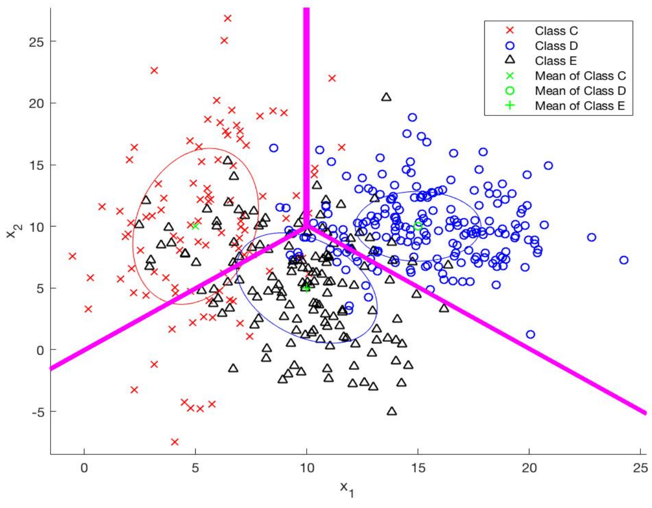

Figure 6: MED decision boundary applied on classes C, D, and E clusters (case 2)

The MED classifier is based on Euclidean distance, and is defined by the equation:

$$d_{MED}(\bar{x}, \bar{z_1}) = [(\bar{x} - \bar{z_1})^T (\bar{x} - \bar{z_1})]^{1/2}$$

where x represents a random point, and z_1 represents the prototype of class 1, which is the true mean of the class in this case.

The decision boundary for the MED classifier is equivalent to a matrix of points in the feature space that are equidistant to the prototypes of the two known classes (for the two class case). Thus, it can be simplified, and expressed as:

$$[(\bar{x} - \bar{z_1})^T(\bar{x} - \bar{z_1})]^{1/2} = [(\bar{x} - \bar{z_2})^T(\bar{x} - \bar{z_2})]^{1/2}$$

The MED decision boundary is generally expressed as a linear relationship between the feature spaces x_1 and x_2, making it the least robust of all the classifiers due to its curve fitting inabilities. Visually, in figures 5 and 6 it is clear that the classifier was less robust to noise and outliers, and poor at handling long, thin tendrils around the center of the scatter plot.

3.2 Generalized Euclidean Distance (GED)

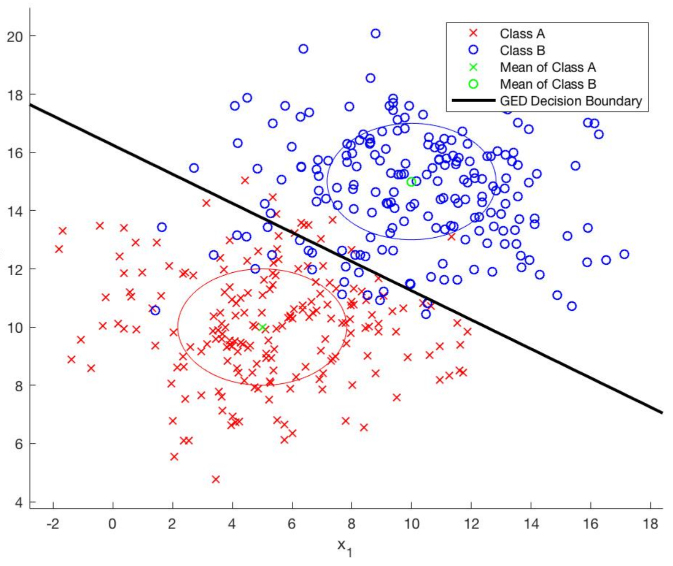

Figure 7: GED decision boundary applied on classes A and B clusters (case 1)

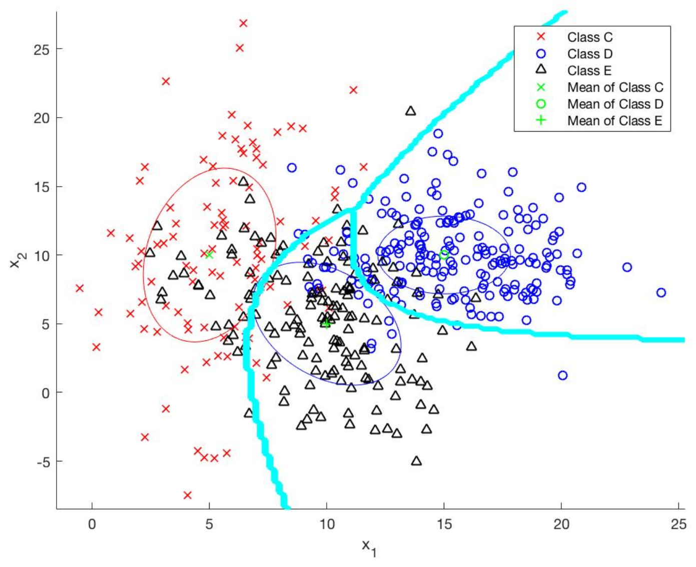

Figure 8: GED decision boundary applied on classes C, D, and E clusters (case 2)

The GED classifier is based on unit standard deviation as a distance metric, and takes into account a weighting factor for class distributions that do not follow unit variance or uncorrelatedness - conditions for application of Euclidean distance. The GED classifier's weighting matrix is the combination of the orthonormal covariance which after further simplifications is equivalent to the inverse covariance of the class. This weighting is applied to the fundamental MED classifier equation and the MED essentially becomes a special case of GED where the Euclidean distance conditions are assumed. The GED for a random pattern x, is defined as:

$$d_{GED}(\bar{x}, \bar{z_1}) = [(\bar{x} - \bar{z_1})^T S_1{}^{-1} (\bar{x} - \bar{z_1})]^{1/2}$$

where the x refers to a random arbitrary point in the feature space, z_1 represents the true mean prototype of a specific class, and S^{-1} represents the inverse covariance of a specific class. The decision boundary for the two class case was subsequently:

$$(\bar{x} - \bar{z_1})^T S_1{}^{-1} (\bar{x} - \bar{z_1}) = (\bar{x} - \bar{z_2})^T S_2{}^{-1} (\bar{x} - \bar{z_2})$$

This was implemented in MATLAB to generate the decision boundary in figures 4 and 5.

It is observed in figure 6 that the GED boundary occurs at the intersection of the standard deviation contours of the two classes A and B respectively. The boundary is linear because the covariance matrices of classes A and B are the same. In the three class case in figure 8, the decision boundary is non-linear, and developed at the intersections of the standard deviation contours of the classes.

The GED classifier was more robust than the MED classifier as it accounted for the individual variances of each class, and developed custom weighting matrices for each class. Thus, it had lower sensitivity to noise and outliers.

3.3 Maximum A Posteriori (MAP)

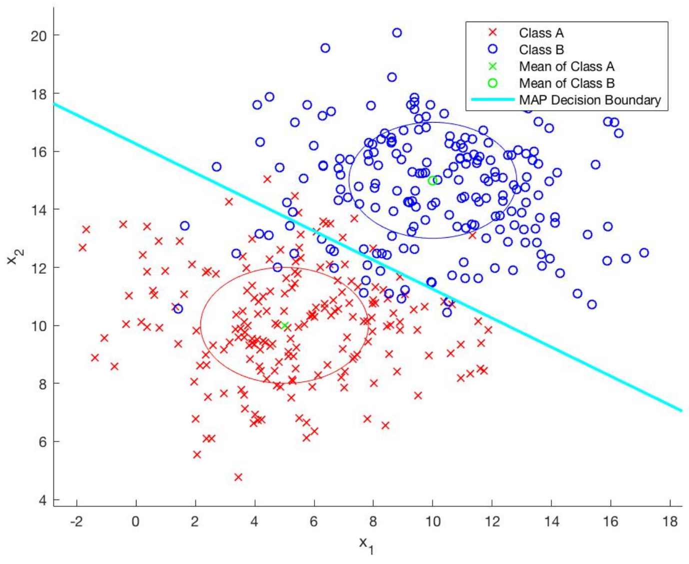

Figure 9: MAP decision boundary applied on classes A and B clusters (case 1)

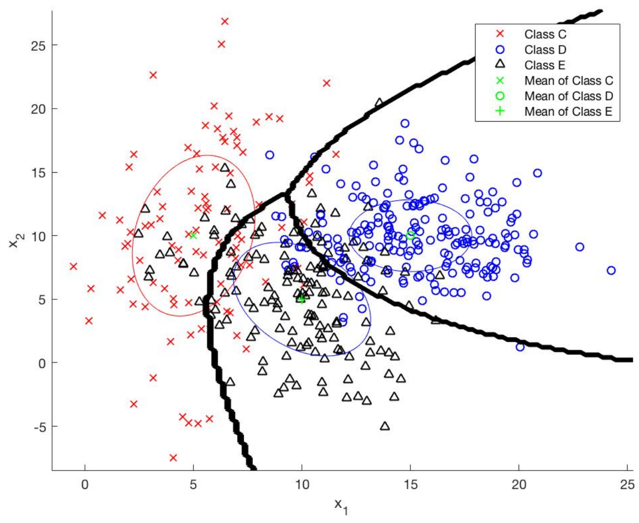

Figure 10: MAP decision boundary applied on classes C, D, and E clusters (case 2)

The MAP classifier considers the probabilistic behaviour of the classes to determine the likelihood of an observed pattern being associated with each class. Fundamentally, the MAP classifier is defined as:

$$P(A|x) <_B >_A P(B|x)$$

where P(A|x) is the posterior class probability for class A and P(B|x) is the posterior class probability for class B. All patterns with a higher posterior probability for A than B will be classified as A, and vice versa. Considering the classes took on normal distributions, and had posterior probabilities equivalent to the size of the class, following conversion into the log-likelihood form and simplification, the MAP decision boundary is expressed as:

$$\bar{x}^T Q_0 \bar{x} + Q_1 \bar{x} + Q_2 + 2Q_3 + Q_4 = 0$$

where

$$Q_0 = S_A{}^{-1} - S_B{}^{-1}$$

$$Q_1 = 2[\overline{m_B}^T S_B{}^{-1} - \overline{m_A}^T S_A{}^{-1}]$$

$$Q_2 = \overline{m_A}^T S_A{}^{-1} \overline{m_A} - \overline{m_B}^T S_B{}^{-1} \overline{m}_B$$

$$Q_3 = \ln\left[\frac{P(B)}{P(A)}\right]$$

$$Q_4 = \ln\left[\frac{|S_A|}{|S_B|}\right]$$

In case 1, the two class case, the MAP decision boundary was similar to the GED decision boundary as observed in figure 9. This is because the MAP decision boundary was simplified to the likelihood classifier form, due to equal posterior probabilities and equal volumes between the classes. In case 2 (figure 10), the MAP decision boundary is non-linear due to unequal posterior probabilities between the classes, and unequal covariances, thus unequal volumes.

3.4 Nearest Neighbor (NN)

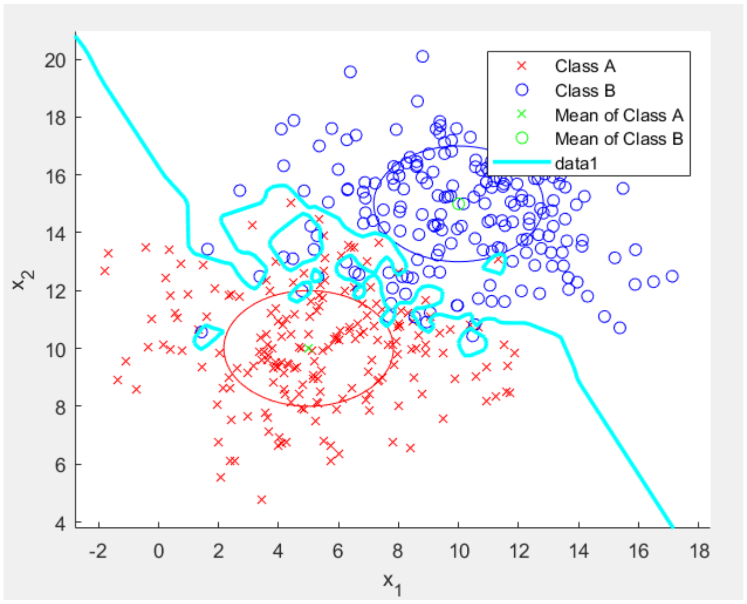

Figure 11: NN decision boundary applied on classes A and B clusters (case 1)

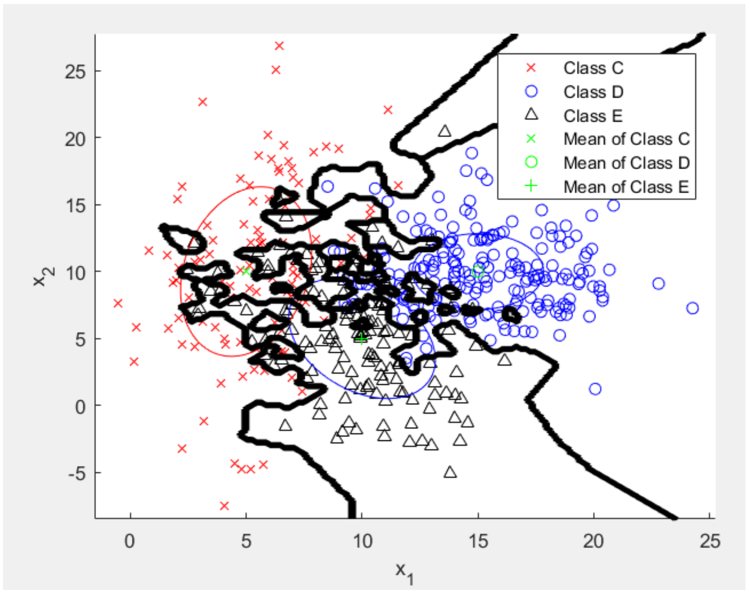

Figure 12: NN decision boundary applied on classes C, D, and E clusters (case 2)

To calculate the NN decision boundary, a prototype for each class was selected based on proximity to the current point. An Euclidean distance measurement was calculated for each member of a class, and the smallest distance in the class was continuously stored. After the entire class was iterated through, the point with the smallest distance was selected as the prototype. The current point was then classified as the class with the closest prototype.

When observing the NN decision boundary, it can be noted that the boundary fits the classes very tightly. This could be advantageous if the classes had long, tendril-like shapes. However, it can also be noted that areas surrounding outliers were classified as the same class as the outlier, potentially misclassifying any points in said region. This shows that the NN classifier is sensitive to noise and outliers.

3.5 K-Nearest Neighbors (KNN)

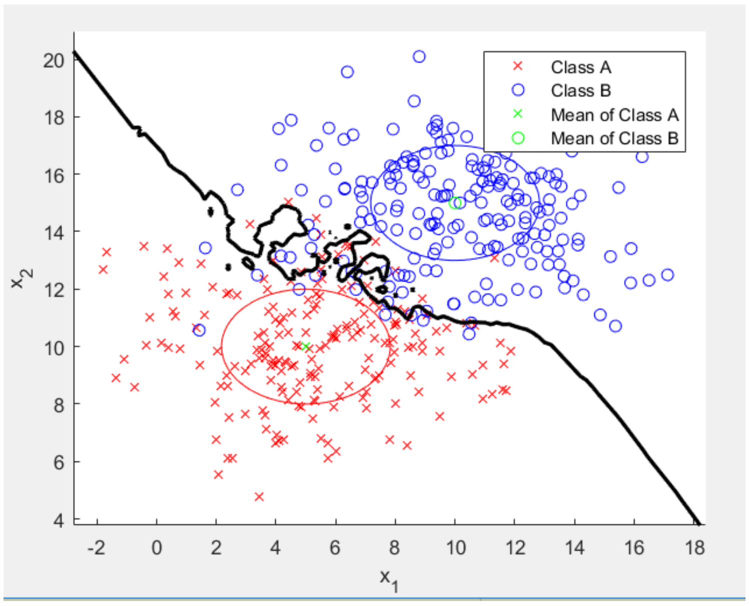

Figure 13: 5NN decision boundary applied on classes A and B clusters (case 1)

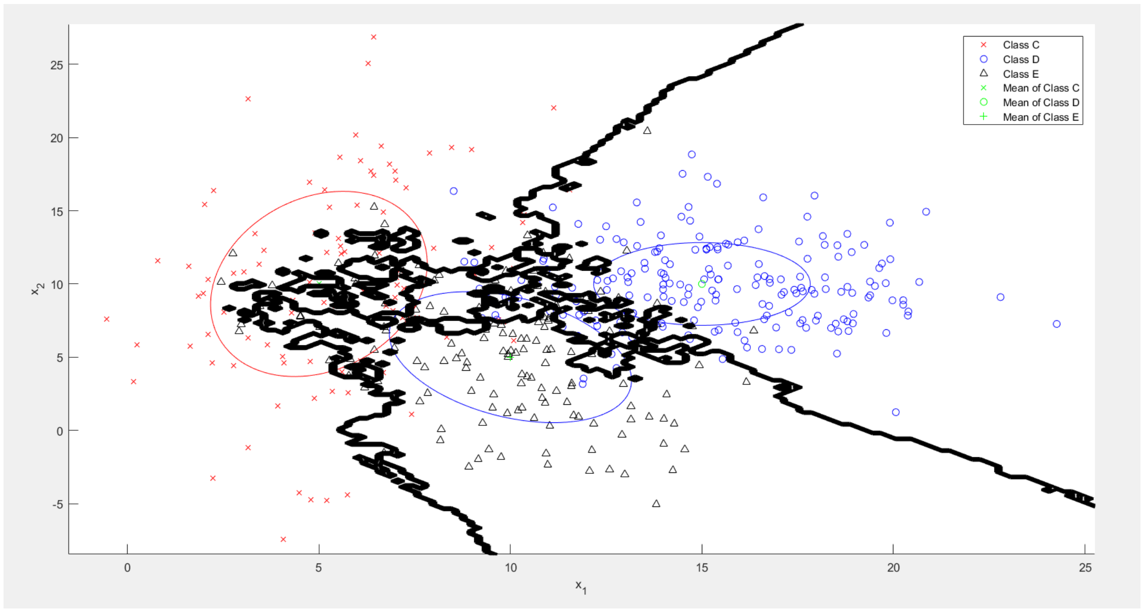

Figure 14: 5NN decision boundary applied on classes C, D, and E clusters (case 2)

The KNN classifier was implemented similarly to the NN classifier. For each class, the Euclidean distance of each point was calculated and stored in an array. After storing the distances for each point, the array was sorted from smallest to largest and the first K distances were selected. The average of the K distances was then calculated and selected as the prototype for the class. After doing this for each class, the point was classified as the same class as the closest prototype. In the examples shown in Figures 9 and 10, K was set to 5.

From observing the 5NN decision boundaries, it can be seen that the decision boundary still fits relatively closely to each class. The major distinction between the 5NN and NN classifiers was that the 5NN classifier better ignored outliers. For example, in Figure 9 the class B outlier at approximately (2, 11) in the class A region were ignored,

4 Error Analysis

4.1 Confusion Matrices

A confusion matrix summarizes the prediction results of a classification problem. Each column in the matrix represents an actual class, while each row represents a predicted class. The number contained at the (i, j)th index of the matrix gives the number of samples that were predicted as being part of class i, but actually belong to class j. This can be seen in the figure below.

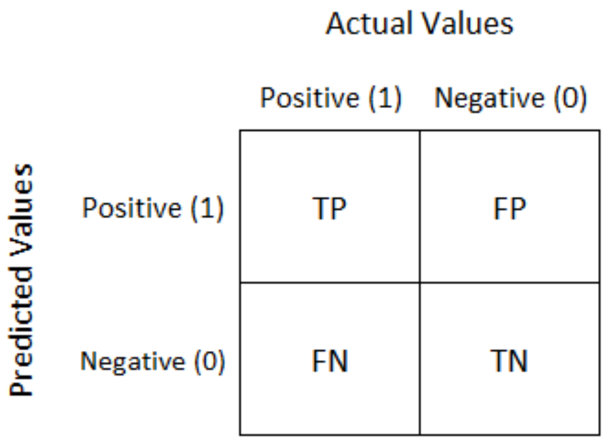

Figure 15: Example 2 class confusion matrix

Confusion matrices are useful tools for error analysis because they don't only reveal how many errors are being made, but the specific types of errors that classifiers are making. For instance, it is more useful to discover that class 4 is often being misclassified as class 2, as opposed to simply knowing that the classifier is making many errors.

To create a confusion matrix for a classifier, we must have a set of sample points with known class labels, as well as a matrix specifying the predicted output of a classifier at each point in the coordinate grid (output of the classifiers described above). We would then iterate through each sample point. We could find the predicted output of the classifier associated with each point by querying the points matrix. From this, we would know the predicted and actual labels associated with the sample point and be able to update the confusion matrix accordingly.

In general, it was found that the confusion matrices for case 2 had larger numbers at their non diagonal indices than their case 1 counterparts. This indicates that the classifiers perform worse when more classes are involved. This makes intuitive sense, considering the fact that a classifier would generally have a higher chance of picking the wrong option when more options are presented.

4.2.1 MED

```
   182    11
    18   189
```

Figure 16: Confusion Matrix for Case 1

```
    69     4    30
     2   172    17
    29    24   103
```

Figure 17: Confusion Matrix for Case 2

4.2.2 GED

```
   190    12
    10   188
```

Figure 18: Confusion Matrix for Case 1

```
    86     5    36
     0   168    14
    14    27   100
```

Figure 19: Confusion Matrix for Case 2

4.2.3 MAP

```
   190    12
    10   188
```

Figure 20: Confusion Matrix for Case 1

```
75     1    23
 1   180    22
24    19   105
```

Figure 21: Confusion Matrix for Case 2

4.2.4 NN

```
187    16
 13   184
```

Figure 22: Confusion Matrix for Case 1

```
69     6    29
 3   174    24
28    20    97
```

Figure 23: Confusion Matrix for Case 2

4.2.5 KNN

```
190    12
 10   188
```

Figure 24: Confusion Matrix for Case 1

```
75     3    14
 3   172    25
22    25   111
```

Figure 25: Confusion Matrix for Case 2

4.2 Experimental Errors

Once the confusion matrix has been determined, finding the experimental error rate, or the proportion of erroneous classifications is trivial. It can be found by dividing the sum of the non diagonal elements of the confusion matrix by the sum of all elements in the confusion matrix. The following table summarizes the experimental error rates for each classifier and case.

Table 1: Experimental Error Rates

Classifier Type	Case	Experimental Error Rate
MED	1	0.0725
MED	2	0.2356
GED	1	0.0550
GED	2	0.2133
MAP	1	0.0550
MAP	2	0.2000
NN	1	0.0725
NN	2	0.2444
KNN	1	0.0550
KNN	2	0.2044

Based on the above table, MAP has the lowest experimental error rate, slightly edging out KNN and GED. It's reasonable to expect that MAP would perform better, considering that it uses more information regarding how the data is distributed and prior classification history. However, since we are assuming that the data is normally distributed and the classes are balanced, it should not have much of an advantage over GED, as shown by the slim difference in their experimental error rates. In contrast, the NN classifier has the highest error rate, which is expected because such classifiers tend to heavily overfit to their training data.