Clusters and Classification Boundaries

SYDE 372 - Lab 1

February 12th, 2017

Group Members:	Krishn Ramesh - 20521942
	Brady Kieffer - 20517665
	Ramandeep Farmaha - 20516974
	Shubam Mehta - 20483061
Instructor:	Professor Wong

1 Introduction

Clustering and classifying data points is a fundamental skill in pattern recognition. Lab 1 provides a hands-on introduction to generating different kinds of decision boundaries, as well as evaluating and comparing their performance. Two cases with true statistics (mean, covariance matrices) for the classes in each case were provided. The initial setup work in generating the sampled data involved bivariate normal distributions subject to orthonormal transformations. Different distance measures were used to construct decision boundaries including MED, GED, MAP, NN and kNN. Lastly, the performance of each distance measure was analyzed and compared. Overall, Lab 1 provided valuable insight into the use of orthonormal transformations, decision boundaries and classification errors to solve practical problems.

2 Generating Clusters

Five classes were considered in this lab, split up into two cases. All the classes followed the following bivariate Gaussian distribution parameters:

CASE 1:

$$\text{Class A:}\quad N_A = 200 \quad \mu_A = \begin{bmatrix}5 & 10\end{bmatrix}^T \quad \Sigma_A = \begin{bmatrix}8 & 0\\ 0 & 4\end{bmatrix}$$

$$\text{Class B:}\quad N_B = 200 \quad \mu_B = \begin{bmatrix}10 & 15\end{bmatrix}^T \quad \Sigma_B = \begin{bmatrix}8 & 0\\ 0 & 4\end{bmatrix}$$

CASE 2:

$$\text{Class C:}\quad N_C = 100 \quad \mu_C = \begin{bmatrix}5 & 10\end{bmatrix}^T \quad \Sigma_C = \begin{bmatrix}8 & 4\\ 4 & 40\end{bmatrix}$$

$$\text{Class D:}\quad N_D = 200 \quad \mu_D = \begin{bmatrix}15 & 10\end{bmatrix}^T \quad \Sigma_D = \begin{bmatrix}8 & 0\\ 0 & 8\end{bmatrix}$$

$$\text{Class E:}\quad N_E = 150 \quad \mu_E = \begin{bmatrix}10 & 5\end{bmatrix}^T \quad \Sigma_D = \begin{bmatrix}10 & -5\\ -5 & 20\end{bmatrix}$$

The clusters were generated using the `randn` function provided by `MATLAB`. First, data was generated with a mean of zero and a variance of one. The provided covariance matrix for each class was diagonalized using Cholesky factorization provided within the `chol` function. The data was then transformed by first multiplying the sampled points by this diagonalized matrix and then adding the class mean to each row. The provided function `bivariate_normal` was used to generate samples for each of the classes provided in the lab. This produced data centered about the provided class mean with their respective covariances.

After generating sample data for each of the classes, they were plotted as a scatter plot with a unit standard deviation contour.

Figure 1: Plot of Case 1

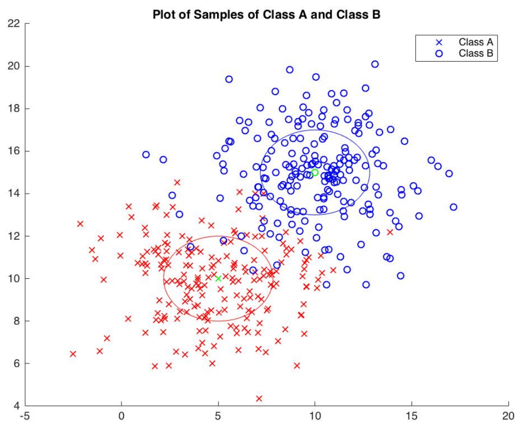
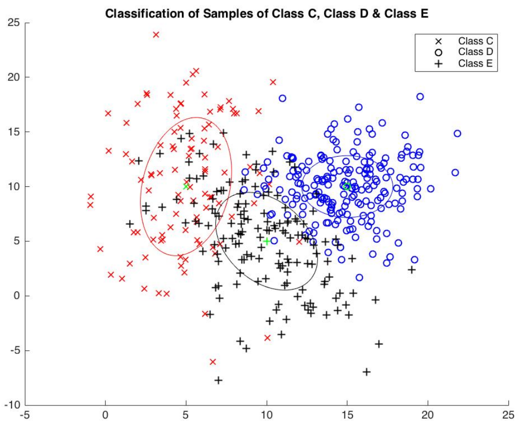


Figure 2: Plot of Case 2



The means of all the classes are plotted in green. The unit standard deviation contours were plotted using the `plot_ellipse` function. The lengths of the major and minor axes of the ellipse were determined by the square root of the 2 eigenvalues of each cluster. The angle of the ellipses was determined by taking the $arctan(\cdot)$ of the ratio of the eigenvector elements of each cluster.

Visually, the unit contour contains approximately 68% of the data, as the contour represents 1 standard deviation from the mean. The unit contour is centered at the mean of each class and follows the shape of the data; its major axis is aligned with the direction of greatest variance of the data, while its minor axis is aligned with the direction of least variance.

3 Classifiers and Error Analysis

All of the classifiers were approached numerically and implemented in `MATLAB`. A 2D grid of points was created for each case and run through all of the classifiers to classify each point in the grid. The decision boundaries were then plotted on top of the sample data using contour plots.

The errors for each classifier were also analyzed. The experimental error rate $P(\epsilon)$ for each case, was calculated as the number of samples classified wrong as a percentage of the total number of samples.

$$P(\epsilon) = \frac{\text{samples classified wrong}}{\text{total number of samples}}$$

The confusion matrix for each classifier was also produced to analyze its performance in further detail.

3.1 Minimum Euclidean Distance (MED)

The idea behind a Minimum Euclidean Distance classifier is to measure the Euclidean distance between the data point to be classified and the mean of all available classes. The data point then gets classified to the class with the lowest Euclidean Distance to its mean. The MED to each class mean was obtained using the following:

$$d_E(\bar{x}, \bar{\mu}_k) = [(\bar{x} - \bar{\mu}_k)^T(\bar{x} - \bar{\mu}_k)]^{1/2}$$

where $d_E(\bar{x}, \bar{\mu}_k)$ represents the MED for data point $\bar{x}$ to class mean $\bar{\mu}_k$.

The MED Classifier was implemented using `MATLAB`. The mathematical form used for implementation was:

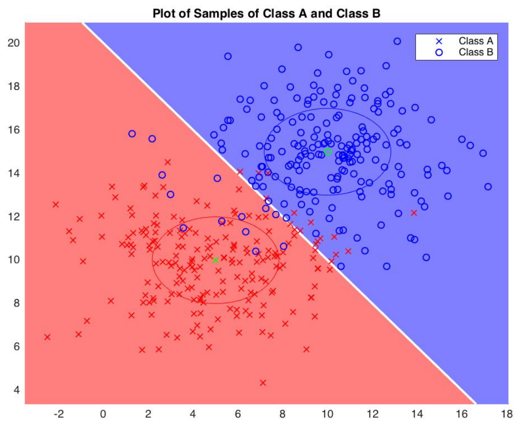
$$[(\bar{x} - \bar{\mu}_A)^T(\bar{x} - \bar{\mu}_A)]^{1/2} < [(\bar{x} - \bar{\mu}_B)^T(\bar{x} - \bar{\mu}_B)]^{1/2}$$

where:

- $\bar{x}$ was the data point to be classified.
- $\bar{\mu}_k$ was the mean of class k.

Once the MED was computed for each data point, it was classified to its respective class and the decision boundary separated each class. The function specified above to calculate the MED lies in the `MED.m` file. Running the function on the classes provided in Case 1 generate the figure shown below in Figure 3.

Figure 3: Plot of Case 1 with MED Classifier (white line)



The MED classifier results were as expected with the decision boundary being a diagonal line between both classes. The MED line was a perpendicular bisector of the line between the two means.

For case 1, the experimental error rate was:

$$P(\epsilon) = 0.0725$$

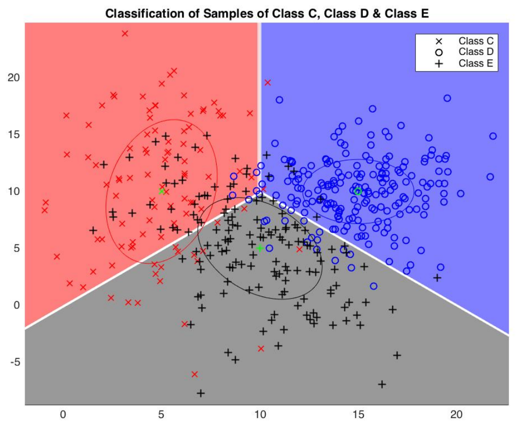
A confusion matrix of the results can be viewed in table 1 below.

	Predicted		
Actual	**A**	**B**	**Totals**
A	184	16	200
B	13	187	200
Totals	197	203	

Table 1: Confusion matrix for Case 1 with MED.

Case 2 consisted of 3 different classes and the same classifier function was used for each pair of classes. The result for case 2 is shown below in Figure 4.

Figure 4: Plot of Case 2 with MED Classifier (white lines)



The results were once again linear as expected but there were still some points past the decision boundaries due to the error. For case 2, the experimental error rate was:

$$P(\epsilon) = 0.1933$$

which as expected was higher than case 1 due to a higher number of classes. The confusion matrix for case 2 is shown below in table 2.

	Predicted			
Actual	**C**	**D**	**E**	**Totals**
C	80	2	18	100
D	4	177	19	200
E	37	7	106	150
Totals	121	186	143	

Table 2: Confusion matrix for Case 2 with MED.

3.2 Generalized Euclidean Distance (GED)

The GED classifier was implemented using `MATLAB`. The mathematical form used for implementation was:

$$(\bar{x} - \bar{\mu}_A)^T \Sigma_A^{-1} (\bar{x} - \bar{\mu}_A) < (\bar{x} - \bar{\mu}_B)^T \Sigma_B^{-1} (\bar{x} - \bar{\mu}_B)$$

Where:

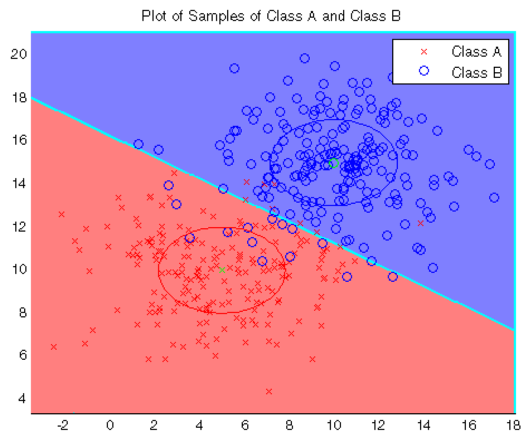
- $\bar{x}$ was the data to be classified.
- $\bar{\mu}_C$ was the class mean.
- Σ_C was the covariance matrix for the class.

Classifying whether a given point belonged to either class A or B was trivial by rearranging the inequality to obtain the following:

$$(\bar{x} - \bar{\mu}_A)^T \Sigma_A^{-1} (\bar{x} - \bar{\mu}_A) - (\bar{x} - \bar{\mu}_B)^T \Sigma_B^{-1} (\bar{x} - \bar{\mu}_B) < 0$$

If the difference was less than zero then the point was chosen to belong to class A. If it was greater than zero then class B was chosen. The decision boundary would then lie where the difference between the distances was zero. A function to calculate the GED between two classes was created and provided within the `GED.m` folder. Running `GED` on the classes provided within Case 1 generated the results within figure 5.

Figure 5: Plot of decision boundary for GED on Case 1.



The GED classifier behaved as expected with the decision boundary being drawn between the two standard deviation contours. It should also be noted that the decision boundary was a straight line due to the features being uncorrelated.

For Case 1 with the GED classifier the probability of error was:

$$P(\epsilon) = 0.0550$$

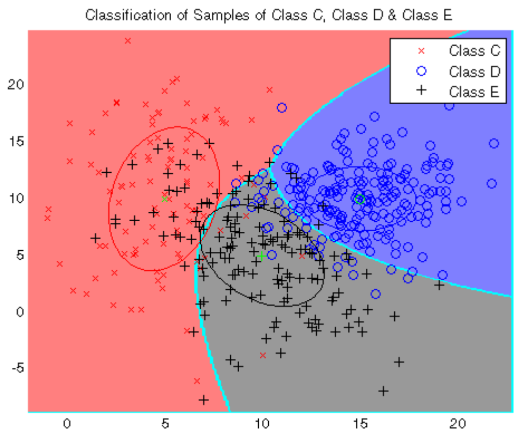
And a confusion matrix of the results can be viewed within table 3.

	Predicted		
Actual	**A**	**B**	**Totals**
A	192	8	200
B	14	186	200
Totals	206	194	

Table 3: Confusion matrix for Case 1 with GED.

The results of the GED classifier on Case 2 can be viewed within figure 6. Again the classifier behaved as expected with the decision boundaries being non-linear and passing through any intersection of the standard deviation contours.

Figure 6: Plot of decision boundary for GED on Case 2.



For Case 2 with the GED classifier the observed probability of error was:

$$P(\epsilon) = 0.1756$$

And a confusion matrix can be viewed within table 4

	Predicted			
Actual	**C**	**D**	**E**	**Totals**
C	91	0	9	100
D	3	169	28	200
E	35	4	111	150
Totals	129	173	148	

Table 4: Confusion matrix for Case 2 with GED.

3.3 Maximum A Posteriori (MAP)

The MAP classifier is a powerful classifier which assigns an unknown pattern as belonging to a class based on the highest *a posteriori* class probabilities i.e. $P(A|\bar{x})$. It makes the reasonable assumption that the class with the highest probability given the particular point $\bar{x}$ is the class that $\bar{x}$ belongs to. Mathematically:

$$\bar{x} \in A \quad \text{iff} \quad P(A|\bar{x}) > P(B|\bar{x})$$

This can be simplified using Bayes Rule, assuming Normal distributions for both classes and using log-likelihood form to obtain the decision boundary:

$$(\bar{x} - \bar{\mu}_B)^T \Sigma_B^{-1} (\bar{x} - \bar{\mu}_B) - (\bar{x} - \bar{\mu}_A)^T \Sigma_A^{-1} (\bar{x} - \bar{\mu}_A) = 2 \ln \frac{P(B)}{P(A)} + \ln \frac{|\Sigma_A|}{|\Sigma_B|}$$

When coding it up in `MATLAB`, the classifier was represented by:

$$\bar{x}^T Q_0 \bar{x} + Q_1 \bar{x} + Q2 + 2Q3 + Q4$$

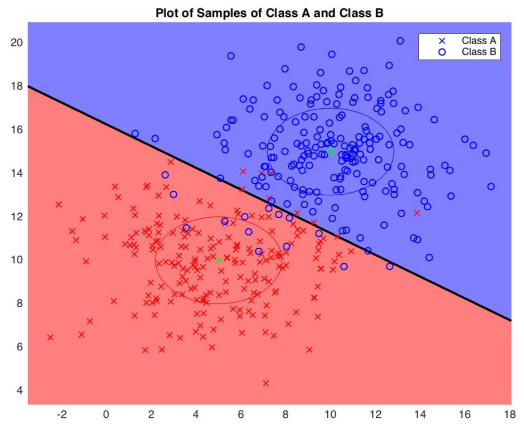
where:

$$Q_0 = \Sigma_A^{-1} - \Sigma_B^{-1}$$
$$Q_1 = 2[\bar{\mu}_B^T \Sigma_B^{-1} - \bar{\mu}_A^T \Sigma_A^{-1}]$$
$$Q_2 = \bar{\mu}_A^T \Sigma_A^{-1} \bar{\mu}_A - \bar{\mu}_B^T \Sigma_B^{-1} \bar{\mu}_B$$
$$Q_3 = \ln \frac{P(B)}{P(A)}$$
$$Q_4 = \ln \frac{|\Sigma_A|}{|\Sigma_B|}$$

The prior probabilities were proportional to the number of samples in each class using the true class statistics given in section 1. If the classifier value for a point was negative, it belonged to class A, otherwise it belonged to class B. Points with a MAP classifier value of 0 were on the decision boundary.

Running a MAP classifier through the sample data for case 1 (with class A and class B) produced a line separating the classes.

Figure 7: Plot of Case 1 with MAP Classifier (black line)



Although the MAP classification was not perfect, it is the optimal decision boundary considering the posterior probabilities. It is noteworthy that the MAP decision line is not perpendicular to the line connecting the means of the two classes.

The experimental error rate for case 1 was:

$$P(\epsilon) = 0.0550$$

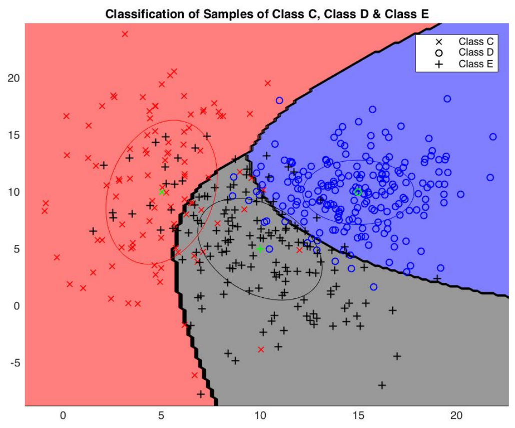
The confusion table for case 1 can be seen below.

	Predicted		
Actual	**A**	**B**	**Totals**
A	192	8	200
B	14	186	200
Totals	206	194	

Table 5: Confusion Table for Case 1 with MAP

Case 2 with 3 overlapping classes was more complicated, but the MAP classifier still handled it well.

Figure 8: Plot of Case 2 with MAP Classifier (black curve)



This time the decision boundary was no longer linear, but curved. The MAP classifier nicely separated all 3 classes with most of the samples being classified correctly.

The experimental error rate for case 2 was:

$$P(\epsilon) = 0.1511$$

This was higher than the error rate in case 1 but that was expected due to the more complex nature of this problem. There was a lot of overlap among the 3 classes, and they were definitely not linearly separable. Still, the MAP classifier did an admirable job.

The confusion table for case 2 goes over the classification in more detail.

	Predicted			
Actual	**C**	**D**	**E**	**Totals**
C	84	1	15	100
D	1	182	17	200
E	21	13	116	150
Totals	106	196	148	

Table 6: Confusion matrix for Case 2 with MAP

3.4 Nearest Neighbor (NN) and k-Nearest Neighbor (kNN)

The Nearest Neighbor and k-Nearest Neighbor classifiers differ from the rest of the classifiers discussed in this report primarily because they utilize all elements of sample classes, rather than the mean and covariance.

The basic idea of the NN classifier is that for a given point P, the Euclidean distances between P and all points within sample classes A and B are computed. Point P is then assigned to the class that contains the closest point to P. This can be summarized in the following mathematical expression:

$$z_k(x) = \bar{x}_k : d_E(\bar{x}, \bar{x}_k) = min_i d_E(\bar{x}, \bar{x}_i) \forall \bar{x}_i \in c_k \tag{1}$$

Where:

- $z_k(x)$ is the potential class that the point belongs to

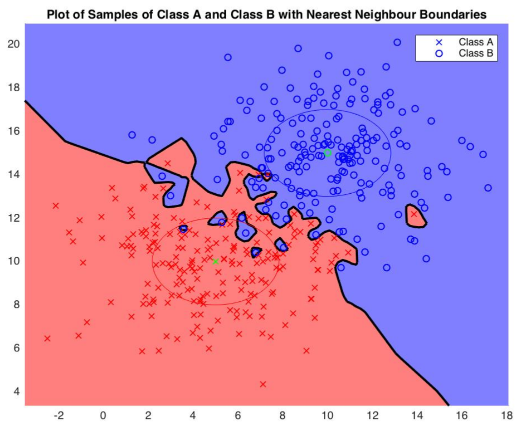
- $\bar{x_k}$ is the closest sample to the point $\bar{x}$
- c_k is the sample class

This can be translated to determining the decision boundary between two classes in the following method:

1. Iterate over each point in the generated 2D grid
2. For each point, compute the distances between all sample points in both classes and determine the nearest neighbor for both classes
3. Assign the value of that point to the difference of the two distances

The method above generates a contour map that indicates the regions for either class. If the value of a point is negative, it belongs to class A and class B if it is positive. If the value of a point is 0, this indicates that no decision can be made about which class that point belongs to, i.e. it forms the boundary of the two classes.

Figure 9: Plot of Case 1 with NN Classifier (Black Curves)



Unlike the previous classifiers, the NN decision boundary is much more precise, and accounts for the outliers within the two sample classes. However, it's interesting to note that the boundary itself is not contiguous; there are smaller island-like sub-boundaries that form around the extreme outliers in either sample class.

In order to conduct an error analysis for the NN and kNN approaches, testing data needs to be generated to offset NN's reliance to overfit on training data. The testing data is created by using a new seed in the randomizing function and applying it to the `bivariate_normal` function.

The experimental error rate for case 1 with the testing data was:

$$P(\epsilon) = 0.0950 \tag{2}$$

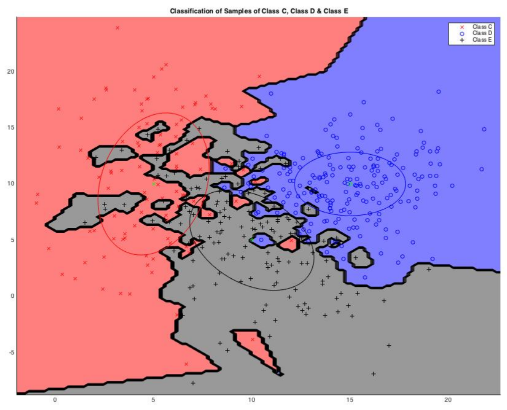
The confusion matrix for Case 1 is summarized below:

	Predicted		
Actual	**A**	**B**	**Totals**
A	177	23	200
B	15	185	200
Totals	192	208	

Table 7: Confusion Table for Case 1 with NN

Case 2 has 3 classes instead of 2, with decision boundaries computed between all three pairs of classes. Similar to Case 1, the NN classifier is much more precise than the other classifiers, however, there still are islands of sub-boundaries.

Figure 10: Plot of Case 2 with NN Classifier (Black Curves)



The experimental error rate was again determined in a similar fashion to Case 1: separate testing data was created using the same parameters as the training data but with a different random seed. The error rate was:

$$P(\epsilon) = 0.2400 \tag{3}$$

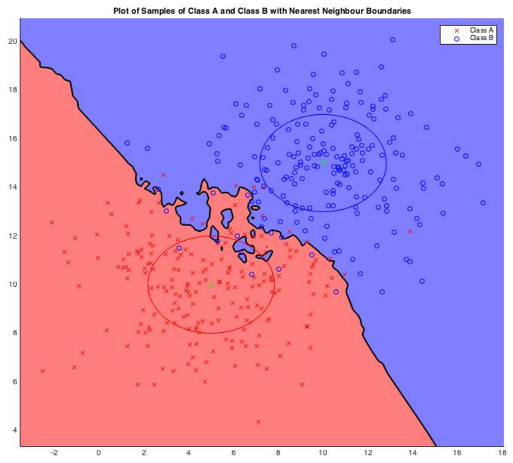
The confusion matrix for Case 2 is shown below:

	Predicted			
Actual	**C**	**D**	**E**	**Totals**
C	64	5	31	100
D	1	176	23	200
E	23	25	102	150
Totals	88	206	156	

Table 8: Confusion matrix for Case 2 with NN

The kNN approach takes the same underlying implementation, but instead of using the single nearest neighbors for each class, it takes the k nearest neighbors and generates a prototype based on the sample mean of the nearest neighbors. For the purposes of this lab, $k = 5$. These mean neighbors for each class are used then used to determine the decision boundary.The kNN classifier attempts to mitigate the overfitting in the NN classifier through the mean neighbor approach. Below is the decision boundary for Case 1 generated by the kNN classifier:

Figure 11: Plot of Case 1 with kNN Classifier (Black Curves)



Comparing Figures 9 and 11, it is clear that the kNN classifier reduces the overfitting and smooths out the resulting decision boundary. Using mean sampling, many of the sub-boundaries are also eliminated. The experimental error rate for the kNN classifier is shown below:

$$P(\epsilon) = 0.0750 \tag{4}$$

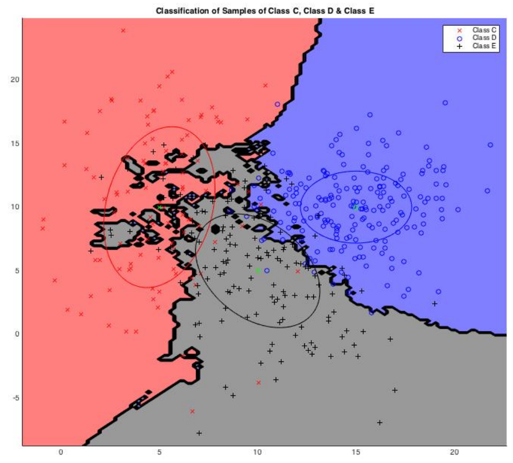
The decrease in error rate compared to the NN classifier can be attributed to the decrease in overfitting; the decision boundary is much more generalized and can account for variations in the sample classes. The confusion matrix for Case 1 is shown below:

	Predicted		
Actual	**A**	**B**	**Totals**
A	179	21	200
B	9	191	200
Totals	188	212	

Table 9: Confusion Table for Case 1 with kNN

Similar to Case 1, the kNN decision boundary for Case 2 is more generalized and has less islands or sub boundaries (see Figures 10 and 12).

Figure 12: Plot of Case 2 with kNN Classifier (Black Curves)



The experimental error rate is again less than the NN classifier equivalent:

$$P(\epsilon) = 0.2156 \tag{5}$$

The confusion matrix provides a more detailed view of the classification and resulting errors:

	Predicted			
Actual	**C**	**D**	**E**	**Totals**
C	65	4	31	100
D	2	182	16	200
E	23	21	106	150
Totals	90	207	153	

Table 10: Confusion matrix for Case 2 with kNN

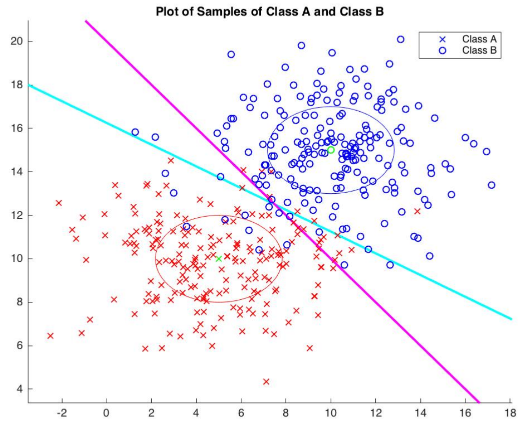
4 Performance Comparison of Classifiers

Each classifier was, to a degree, able to correctly assign classes. The decision boundaries of each classifier overlaid on one another show the differences (or lack thereof) in their predictions. They also show the strengths and weaknesses of each classifier and demonstrate the power of a classifier when it is able to leverage all of the information stored within a class prototype.

4.1 Case 1

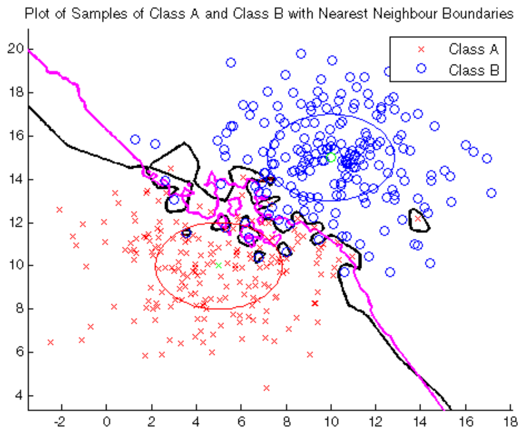
The decision boundaries for each distance based classifier can be viewed together in figure 13. As is expected the MED classifier simply drew a line that is a perpendicular bisector to the line between each class prototype. However, both the GED and MAP classifiers gave identical decision boundaries. This is due to the fact that each class prototype contained the same covariance matrix and number of samples (i.e. same prior probabilities) meaning the MAP classifier simplified to the GED classifier. If more unit contours were plotted for each class, the GED/MAP line would go through the intersection of the unit contours of the two classes.

Figure 13: Plot of MED (magenta), GED (cyan) and MAP (black) decision boundaries for Case 1.



As mentioned in Section 3.4, the NN and kNN classifiers create much more precise decision boundaries compared to the previous methods. However, the NN classifier leads to overfitting, where extreme values within the sample classes are accounted for and have their own sub-boundaries. This can be seen in Figure 14, where there are distinct black sub-boundaries separate from the primary boundaries. The use of sample mean in the kNN classifier mitigates this, reflected in the much smoother magenta decision boundary, which has very few sub-boundaries.

Figure 14: Plot of NN (black) and kNN (magenta) decision boundaries for Case 1.



Overall for case 1, the MAP/GED classifiers had the lowest error rate of 0.0550. For a simple case with classes with equal variances, the simpler GED classifier performs just as well as MAP.

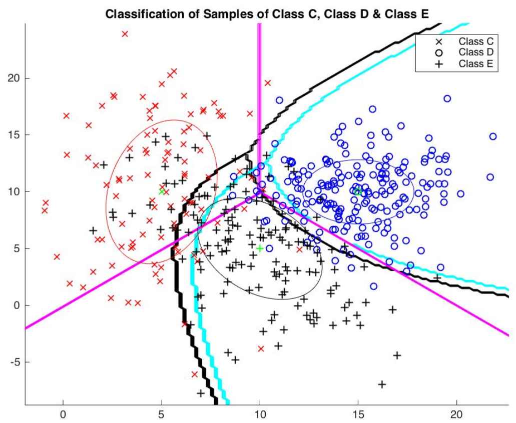
4.2 Case 2

The decision boundaries for each distance based classifier can be viewed in figure 15. Here the MED classifier performed poorly as each decision boundary was simply a line between the class prototypes. This behaviour was expected because the MED classifier simply measures the Euclidean distance between a point and the class prototype. The GED classifier behaved as expected by creating a non-linear decision boundary. This boundary passed through any intersection of the standard deviation contours and nicely between classes D

and E. Finally, as expected, the MAP classifier performed the best. This was due to the fact that the MAP classifier was able to take into account every feature of the class prototype the prior knowledge (number of samples) and the correlations of each class. This allowed the MAP classifier to perform optimally on the given data.

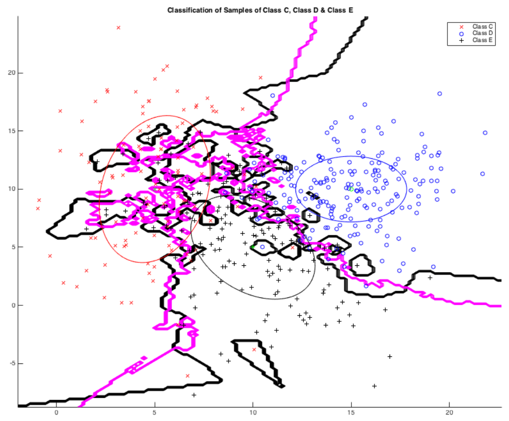
A noteworthy difference between the GED and MAP non-linear boundaries was how the MAP boundary between class C and E shifted left closer to class C. While the GED boundary is bound to the unit standard deviation contour, the MAP boundary is not, suggesting that it can better deal with overlapping classes.

Figure 15: Plot of MED (magenta), GED (cyan) and MAP (black) decision boundaries for Case 2



Similar to Case 1, the decision boundary in Figure 16 drawn by the kNN implementation (magenta) is much smoother with less sub-boundaries due to the decrease in overfitting when compared to the NN classifier (black). This decrease in overfitting for the training data reflects in a better model; when the kNN classifier is applied to testing data, its resulting experimental error rate is less than its NN counterpart.

Figure 16: Plot of NN (black) and kNN (magenta) decision boundaries for Case 2



Overall for case 2, the MAP classifier had the lowest error rate of 0.1511 as it was the most powerful classifier and took advantage of all the information available including the variance, priors and volume of data. The NN and kNN approaches succumbed to overfitting, which led to high error rates.

From the confusion matrix of case 2 of all classifiers, it is apparent that class E was often predicted incorrectly as class C or D, whereas class C and D were rarely predicted as each other. This is mostly due to cluster C and D being relatively far apart while cluster E sat right in between C and D and overlapped them significantly. Another interesting note from the confusion matrix was that the class with the most samples was always predicted more accurately.

5 Conclusion

One of the key takeaways from this lab was that the MAP classifier performed the best out of all classifiers tested against sample data. The MAP classifier, unlike the other classifiers discussed in this lab which rely on the Euclidean distance metric and variance, is also able to take the priors and volumes of each class' data into account when drawing the decision boundaries, thus improving its performance.

While the NN and kNN generally have the most precise decision boundaries for a given set of training data, they have a much higher error rate when applied to testing data due to their reliance to overfitting the training data.