SYDE 372 Lab1 Report

Austin Bianchini - 20506675
Nakash Ali Babwany - 20586425
Rashmi Umashankar - 20562741
Abilas Sathiyanesan - 20567746

February 12, 2019

1 Introduction

The purpose of this lab is to investigate three areas: calculating orthonormal transformations, creating decisions boundaries using different classifiers and assessing classification error.

Five classes with the following bivariate Gaussian distribution parameters were considered.

CASE 1:

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

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

CASE 2:

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

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

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

Both distance based and probability based measures are used for classification. The following is a list of classifiers that are dealt with in this lab.

- Minimum Euclidean Distance (MED)

- Generalized Euclidean Distance (GED)
- Maximum A Posteriori (MAP)
- Nearest Neighbour (NN)
- k-Nearest Neighbours (kNN)

2 Generating Clusters

The clusters were generated using the `random.multivariate_normal` function provided by NumPy which is a scientific computing package for Python. This function draws random samples from a multivariate normal distribution specified by its mean and covariance matrix. To draw the unit standard deviation ellipses for the different set of class samples produced, the eigenvalues and eigenvectors were computed using the `linalg.eig` function provided by NumPy. The direction of the principal axes of the ellipse are provided by the eigenvectors and their lengths are specified by the square root of the corresponding eigenvalues.

Figure 1 and Figure 2 show the resulting class samples and unit standard deviation contours for each of the five classes. The mean of each class is plotted in green.

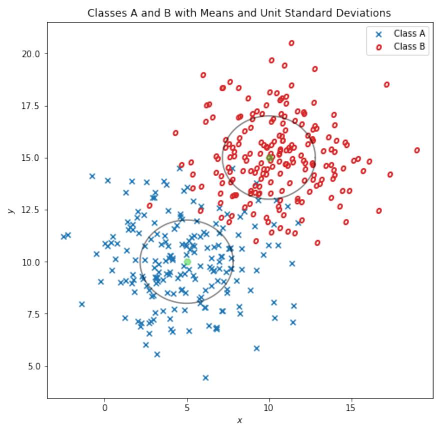


Figure 1: Case 1

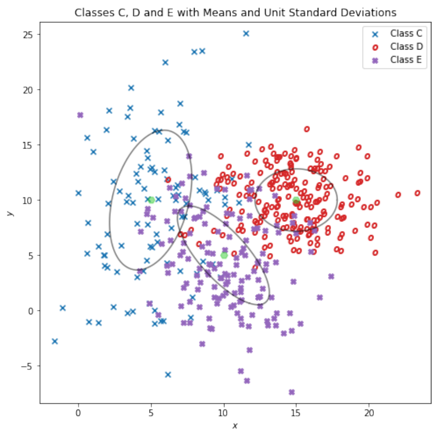


Figure 2: Case 2

The standard deviation is a valuable measure of how far values in a class distribution vary from the mean. For classes A and B, the covariance matrix has zero non-diagonal elements. Hence, the eigendecomposition reveals that the eigenvectors are merely the basis of the feature space and hence, the ellipses are not rotated. In case 2, classes C and E have non-zero non-diagonal elements implying that the features are correlated. The ellipses are oriented in the direction of the greatest variance in the data. Hence, class C is rotated clockwise and class E is rotated anticlockwise.

3 Classifiers

To facilitate ease of computation, our general approach to the problem was numerical in nature. A 2D mesh of the class samples was created and each point was classified individually based on the decision rules for different classifiers.

3.1 Minimum Euclidean Distance (MED)

In MED, the similarity of a pattern with a class is quantified by measuring the Euclidean distance betweem the pattern and the prototype of the class. It is an example of a distance based measure for pattern classification.

Definition: $\underline{x} \in c_k$ iff $d_E(\underline{x}, \underline{x}_k) < d_E(\underline{x}, \underline{x}_l)$ for all $l \neq k$.
That is, $\underline{x}$ is classified as an element of class k if and only if the distance between $\underline{x}$ and the prototype of $\underline{c}_k$ is less than the distance between $\underline{x}$ and any other class prototype.

Our implementation involved measuring the Euclidean distance between each point in the grid and the prototype of each class. In this case, the prototype was taken to be the mean of the class distribution. The distance between the point $\underline{x}$ and the mean $\underline{\mu}_k$ was computed by measuring the norm of the vector given by $(\underline{x} - \underline{\mu}_k)$. This was done using the `linalg.norm` function provided by NumPy that computes the norm of the vector passed as a parameter. For each point, the MED was calculated by comparing the euclidean distances between the point and the different class means. Based on the MED, the point was classified to the corresponding class distribution. The decision boundaries were drawn and filled using the `contourf()` function provided by the `matplotlib` package for Python.

Figure 3 and Figure 4 show the MED decision boundaries for Cases 1 and 2. As expected, in Case 1, the MED boundary is a straight line perpendicular to the line segment joining the means of the two classes. Similarly in case 2, the decision boundary is equidistant to each class.

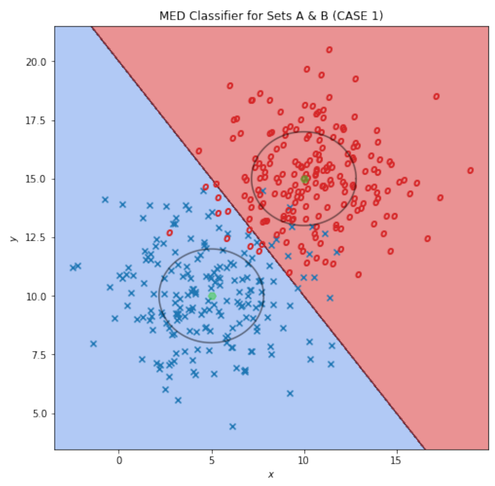


Figure 3: MED Case 1

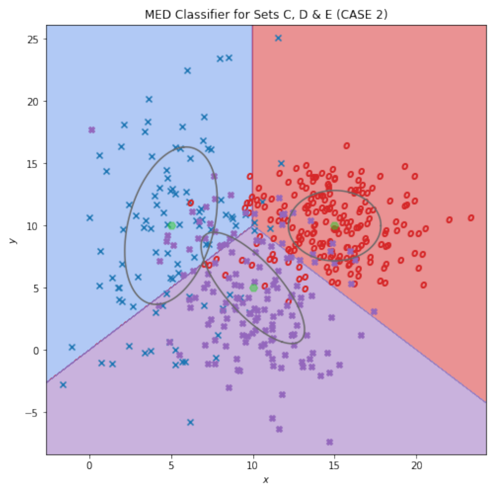


Figure 4: MED Case 2

3.2 Generalized Euclidean Distance (GED)

The GED metric is given by

$$d(\underline{x}, \underline{z}) = [(\underline{x} - \underline{z})^T S^{-1} (\underline{x} - \underline{z})]^{\frac{1}{2}} \tag{1}$$

where $\underline{x}$ is the pattern to be classified, S refers to the covariance of the class and $\underline{z}$ refers to its prototype.

In our implementation, since the mean was chosen to be the prototype, the GED was calculated as follows,

$$d(\underline{x}, \underline{\mu}) = [(\underline{x} - \underline{\mu})^T S^{-1} (\underline{x} - \underline{\mu})]^{\frac{1}{2}} \tag{2}$$

For each point in the 2D grid, the GED between the point and the different class prototypes was computed. If the distance between the point and class A prototype was shorter than that of class B, the point was assigned to class A. Points for which these distances are equal would be part of the decision boundary. The implementation of the 3 class case is similar with the GED being calculated between every point and prototypes for classes C,D and E.

Conceptually, the decision boundary can be found by the intersection of the corresponding equidistance contours between the classes. Since in case 1 the covariance matrices are equal, we expect the decision boundary to be a hyperplane that passes through the midpoint of the means which is what is observed in Figure 5. In case 2, the covariance matrices are different and hence, the decision boundary is non linear in nature as can be seen in Figure 6.

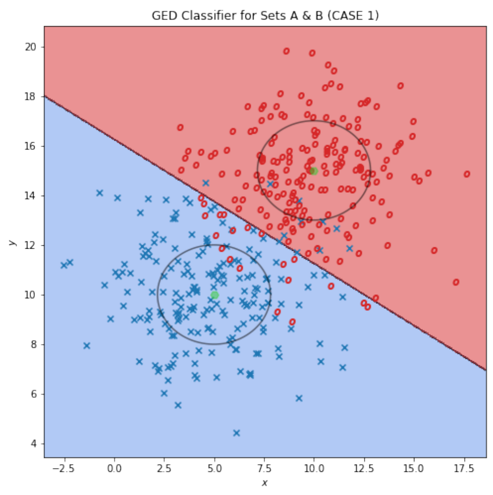


Figure 5: GED Case 1

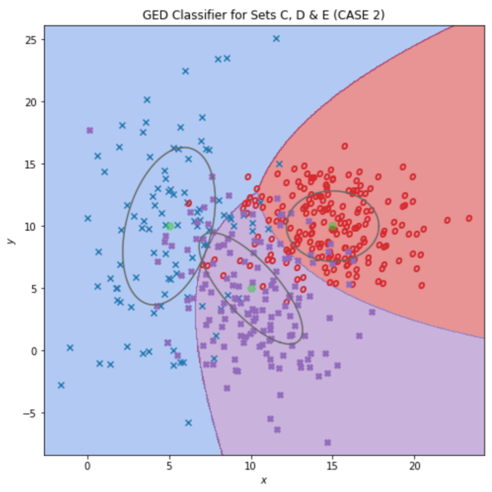


Figure 6: GED Case 2

3.3 Maximum A Posteriori (MAP)

The Maximum A Posteriori (MAP) classifier is an example of a probability measure for classification. It relies on the *a posteriori* probabilities $P(A|\underline{x})$ and $P(B|\underline{x})$ given a pattern $\underline{x}$. The main idea behind MAP is that given a pattern $\underline{x}$, it is most similar to the class that has a greater *a posteriori* probability i.e

$$\underline{x} \in A \text{ iff } P(A|\underline{x}) > P(B|\underline{x}) \tag{3}$$

Since we're dealing with normal distributions, the MAP decision boundary between two classes A and B can be simplified as follows:

$$(\underline{x}-\mu_B)^T \Sigma_B^{-1} (\underline{x}-\mu_B) - (\underline{x}-\mu_A)^T \Sigma_A^{-1} (\underline{x}-\mu_A) \underset{B}{\overset{A}{\gtrless}} 2\left[ln \frac{P(B)}{P(A)}\right] + \left[\frac{|\Sigma_A|}{|\Sigma_B|}\right] \tag{4}$$

In our code, this was implemented by computing the following,

$$\underline{x}^T Q_0 \underline{x} + Q_1 \underline{x} + Q_2 + 2Q_3 + Q_4 \tag{5}$$

where

$$Q_0 = S_A^{-1} - S_B^{-1} \tag{6}$$

$$Q_1 = 2[\underline{m}_B^T S_B^{-1} - \underline{m}_A^T S_A^{-1}] \tag{7}$$

$$Q_2 = \underline{m}_A^T S_A^{-1} \underline{m}_A - \underline{m}_B^T S_B^{-1} \underline{m}_B \tag{8}$$

$$Q_3 = \left[ln \frac{P(B)}{P(A)}\right] \tag{9}$$

$$Q_4 = \left[\frac{|S_A|}{|S_B|}\right] \tag{10}$$

In the 2 class case, the value of eqn. 5 is computed for each point. The *a priori* class probabilities were set to be proportional to the number of samples in each class. If the computed value is less than 0, then the point is assigned to class A and if it is greater than 0, it is assigned to class B. For the 3 class case, a similar approach was taken where the MAP decision boundary was calculated for 2 classes at a time and classifying the point was then carried out by implementing a simple logic.

Since case 1 represents a simple, symmetric case of equally likely, equi-variance class, the decision boundary is a line that passes through the mid point of the two sample means. This can be seen in Figure 7.

As expected, the decision boundary is non-linear and can be observed in Figure 8.

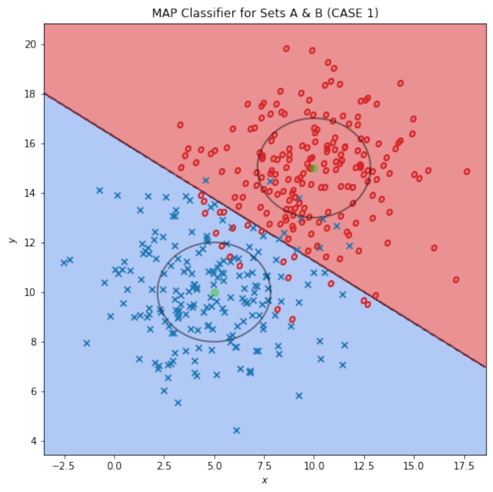


Figure 7: MAP Case 1

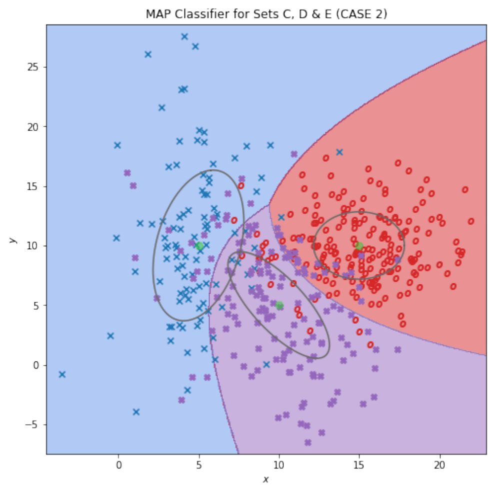


Figure 8: MAP Case 2

3.4 Comparison of MED, GED and MAP

The composite plot illustrating the MED, GED and MAP classifiers for the two different cases are shown in Figures 9 and 10.

For case 1, MED gives the simplest of decision boundaries which is just a straight line perpendicular to the line connecting the two means of Classes A and B. As seen in Figure 9, the decision boundaries obtained by the GED and MAP classifiers are identical. This can be explained by referring to eqn.4. The RHS of the equation is referred to as the threshold and is responsible for biasing the decision rule to favour the more likely class according to the prior probabilities and the determinant of the covariance matrix. Since, for case 1 the prior probabilities and the determinants of the covariance matrices are equal, the threshold term reduces to zero. Hence, the MAP rule for normal distribution reduces to the GED classifier and they are equivalent. This is clearly observed in Fig 9.

For case 2, the MED decision boundary is equidistant from the sample means of classes C, D and E. The GED performs better compared to MED and partitions the feature space better. While considering MAP, it is important to note that Case 2 contains classes that have different covariance matrices and different prior probabilties. Hence, this is a case of a non-zero threshold. In such cases, the GED classifier is sub-optimum and as expected, the MAP performs the better. Looking at Figure 10, we can see that it has a similar non-linear decision boundary but also fewer points that are erroneously classified. This is because MAP takes into account more class-specific information than the other two classifiers in the form of prior probabilities and volume of the equidistance contours.

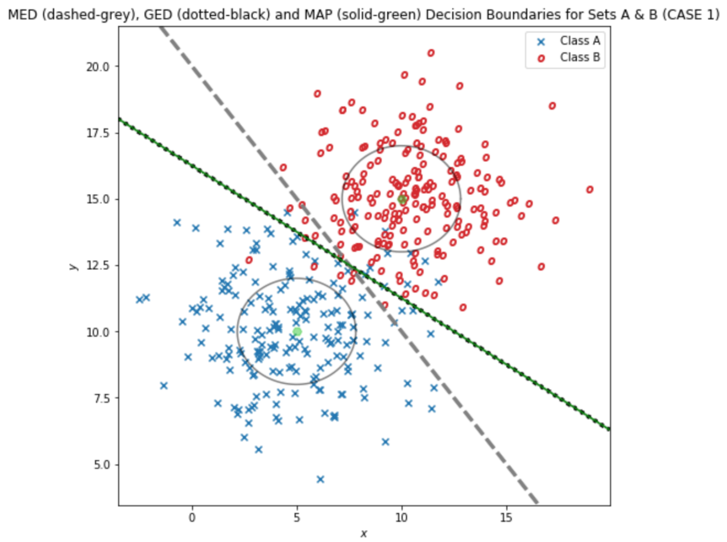


Figure 9: Comparison of MED, GED and MAP - Case 1

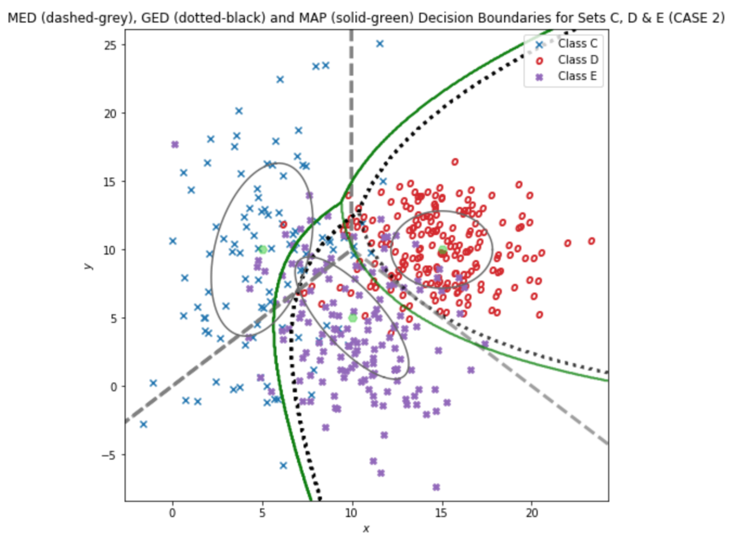


Figure 10: Comparison of MED, GED and MAP - Case 2

3.5 Nearest Neighbor (NN)

The NN algorithm classifies a pattern by calculating the Euclidean distance between the pattern and each labeled sample in all existing classes. It then chooses the class that contains the labeled sample that provided the minimum Euclidean distance between the pattern and the sample.

Our approach was to calculate the Euclidean distance between each labeled sample using the `linalg.norm` function for every point in the 2D grid. For each point, the class associated with the shortest Euclidean distance between a sample and the point is assigned to it.

Figure 11 and Figure 12 show the decision boundaries for NN for cases 1 and 2 respectively. NN algorithms are known to be highly sensitive to noise and outliers and overfits the training set.

3.6 k-Nearest Neighbor (kNN)

The kNN algorithm works similar to the NN algorithm, but it finds the 'k' number of labeled samples with the minimum euclidean distances for each class. It then averages these 'k' values for each class which is then used as the prototype for the respective classes.

Figures 13 and 14 show the decision boundary for kNN for cases 1 and 2 respectively. For our kNN, where k = 5, it performs better when it comes to noise and outliers as compared to NN. The data is not overfitted here for some of the larger outliers.

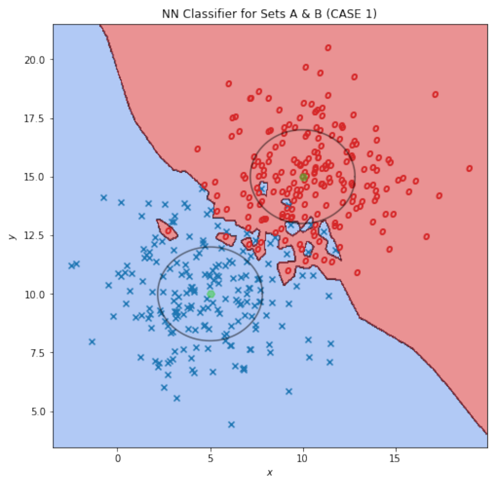


Figure 11: NN Case 1

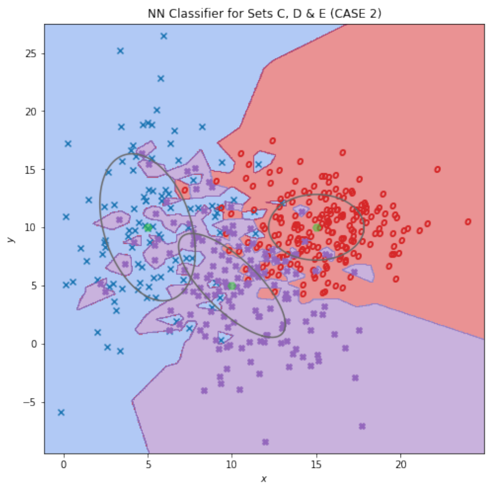


Figure 12: NN Case 2

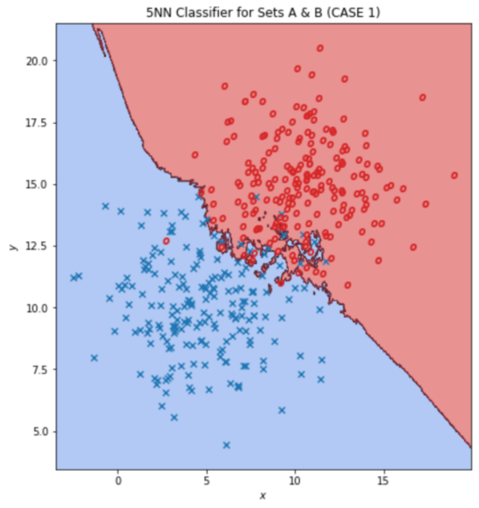


Figure 13: kNN Case 1

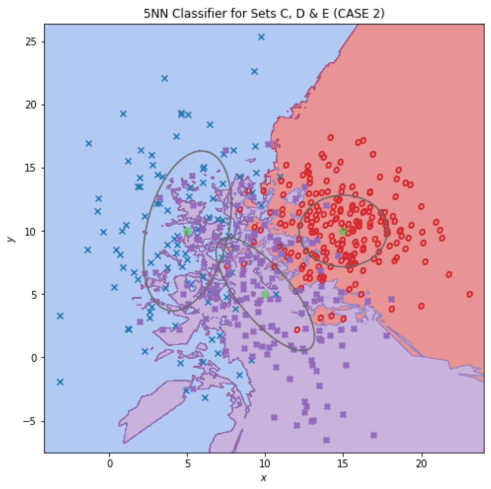


Figure 14: kNN Case 2

3.7 Comparison of NN and kNN

The composite plot illustrating the distance based classifiers (NN and kNN) for the two different cases are shown in Figures 15 and 16.

For case 1 we can immediately see that the NN algorithm has overfitted on all outliers. kNN performs similarly, however the boundary tends to be smoother in most cases and kNN does not overfit on some of the larger outliers. This can be explained by our value of 'k' which we have chosen to be 5. This makes it so our

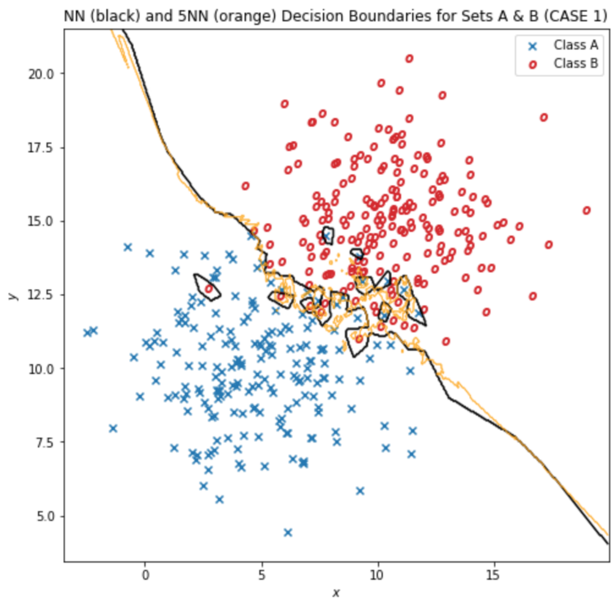


Figure 15: Comparison of NN and kNN classifiers - Case 1

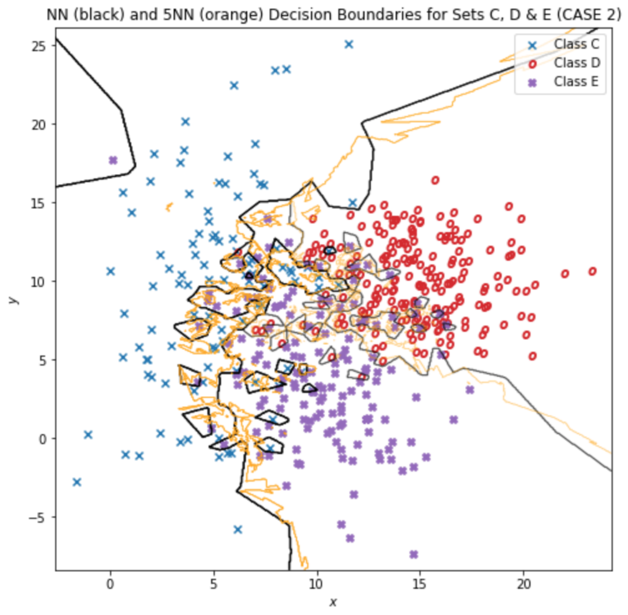


Figure 16: Comparison of NN and kNN classifiers - Case 2

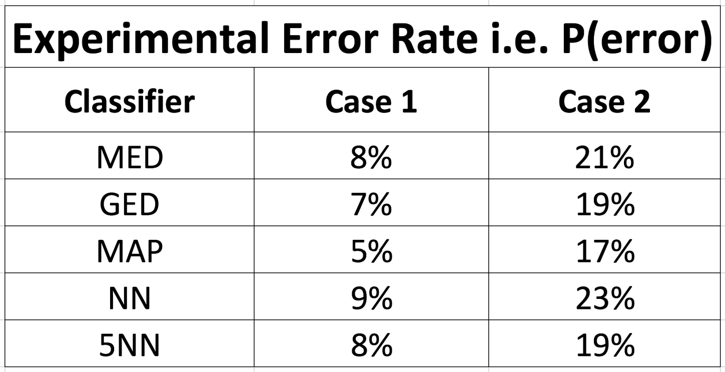
kNN algorithm takes the average of the 5 closest labeled samples by euclidean distance. NN however just takes the closest labeled sample by euclidean distance with no averaging involved. This is what causes the overfitting on even the large outliers that kNN can avoid.

For case 2 we see some of the same patterns exhibited in case 1 but to an even larger degree. kNN's boundary seems to follow the rough outline of NN's boundary but still tends to be smoother. There are cases where it deviates quite a bit however. NN still overfits all outliers and this time it's shown very clearly as there is a part of the boundary that completely breaks off to the other end of the plot. kNN has cases of overfitting as well, but not to the degree of our NN algorithm. This again can be explained by our value chosen for 'k' which is 5. The higher the value of 'k', the less cases of overfitting we are likely to see. As long as 'k' is an integer greater than 1 it will overfit less than the NN algorithm. At 1, since it is averaging just 1 labeled sample it will perform the exact same as the NN algorithm.

4 Error Analysis

The experimental error rate for each of the classifiers was calculated using the following formula, and then converted into a percentage:

$$\text{Error Rate} = \frac{\text{Number of wrongly classified samples}}{\text{Total number of samples}} \tag{11}$$



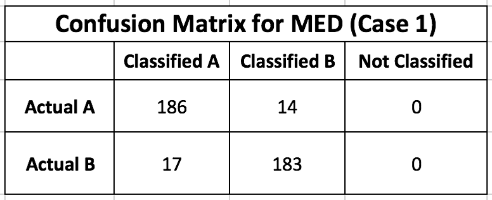
Experimental Error Rate i.e. P(error)

Classifier	Case 1	Case 2
MED	8%	21%
GED	7%	19%
MAP	5%	17%
NN	9%	23%
5NN	8%	19%

Figure 17: Experimental Error Rate

From Figure 17 above, we can see that the experimental error was the smallest for the MAP classifier. This supports the theory because MAP uses the knowledge of the prior probabilities and also the knowledge of the Gaussian distribution. The highest error in Figure 17 is that of the Nearest Neighbor classifier, closely followed by the MED classifier. This can be explained by the fact that the NN classifier over-fits the data and the fact that the MED classifier only uses mean prototypes. The fusion of these two, i.e. the 5NN classifier, thus performs better than both of them.

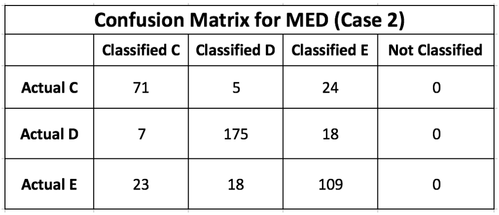
Another important observation from Figure 17 is that the error rates for Case 2 are much higher than Case 1 for all the classifiers. This can be explained by the fact that Case 2 has an extra class to classify and that two of its classes have correlations between the features. All of these reasons contribute towards the higher complexity of Case 2, thus increasing its error rate.



Confusion Matrix for MED (Case 1)

	Classified A	Classified B	Not Classified
Actual A	186	14	0
Actual B	17	183	0

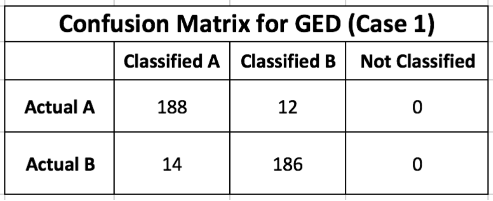
Figure 18: MED Case 1



Confusion Matrix for MED (Case 2)

	Classified C	Classified D	Classified E	Not Classified
Actual C	71	5	24	0
Actual D	7	175	18	0
Actual E	23	18	109	0

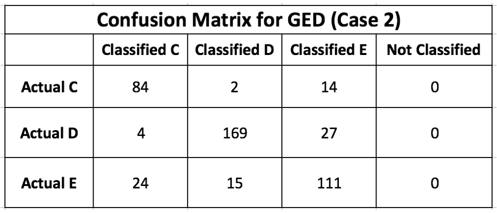
Figure 19: MED Case 2



Confusion Matrix for GED (Case 1)

	Classified A	Classified B	Not Classified
Actual A	188	12	0
Actual B	14	186	0

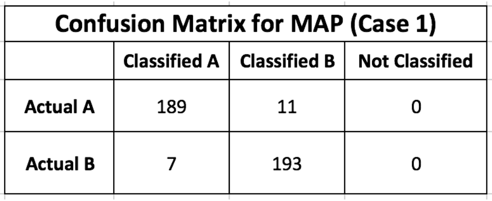
Figure 20: GED Case 1



Confusion Matrix for GED (Case 2)

	Classified C	Classified D	Classified E	Not Classified
Actual C	84	2	14	0
Actual D	4	169	27	0
Actual E	24	15	111	0

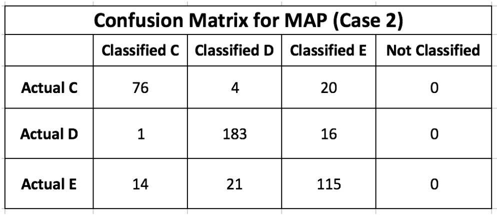
Figure 21: GED Case 2



Confusion Matrix for MAP (Case 1)

	Classified A	Classified B	Not Classified
Actual A	189	11	0
Actual B	7	193	0

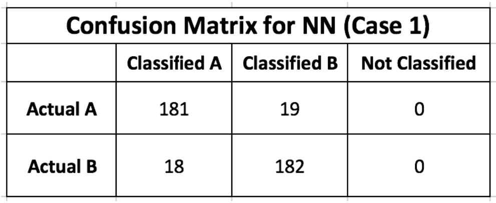
Figure 22: MAP Case 1



Confusion Matrix for MAP (Case 2)

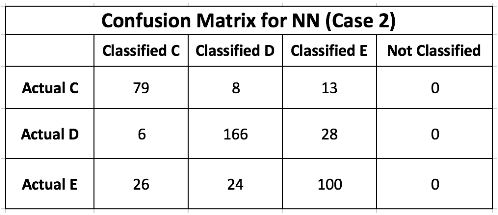
	Classified C	Classified D	Classified E	Not Classified
Actual C	76	4	20	0
Actual D	1	183	16	0
Actual E	14	21	115	0

Figure 23: MAP Case 2



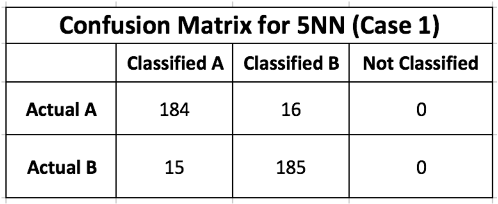
Confusion Matrix for NN (Case 1)			
	Classified A	Classified B	Not Classified
Actual A	181	19	0
Actual B	18	182	0

Figure 24: NN Case 1



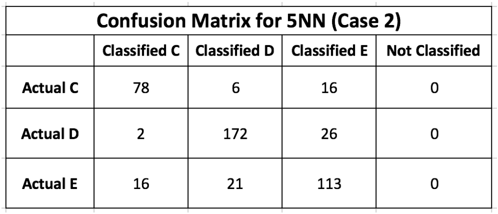
Confusion Matrix for NN (Case 2)				
	Classified C	Classified D	Classified E	Not Classified
Actual C	79	8	13	0
Actual D	6	166	28	0
Actual E	26	24	100	0

Figure 25: NN Case 2



Confusion Matrix for 5NN (Case 1)			
	Classified A	Classified B	Not Classified
Actual A	184	16	0
Actual B	15	185	0

Figure 26: 5NN Case 1



Confusion Matrix for 5NN (Case 2)				
	Classified C	Classified D	Classified E	Not Classified
Actual C	78	6	16	0
Actual D	2	172	26	0
Actual E	16	21	113	0

Figure 27: 5NN Case 2

The confusion matrices above provide a deeper look into the error rates from the previous section. Firstly, it can be seen that the high error rates for Case 2 could have been caused by Class E because it is often classified incorrectly. Moreover, the NN and 5NN classifiers seem to find it difficult to differentiate between Class E and Class D since a lot of Class D samples get classified as Class E and vice versa, which also adds to the higher error rate. The matrices also show, from the last column in each matrix, that none of the classifiers were ever in a situation where they did not make a decision, which could be because of the experimental nature of the classification, as opposed to doing it analytically; the probability of two floats being equal is just very low.