SYDE 572 - Lab 1

Feb 26th, 2021

Group Members:
Christina Ji - 20774617
Soha Kheiri - 20784724
Naomi Kothiyal - 20780817
Sonali Patel - 20755220

Instructor:
Professor Alex Wong

1. Introduction

The purpose of Lab 1 was to investigate three related areas: calculating orthonormal transformations, creating decision boundaries, and assessing classification error. The classifiers investigated were Minimum Euclidean Distance (MED), Generalized Euclidean Distance (GED), Maximum A Posteriori (MAP), Nearest Neighbour (NN), and k-Nearest Neighbours (kNN) with k = 5. Distance metrics used included Euclidean and the generalized Euclidean distances. Additionally, an evaluation and comparison of each classifier's prediction error was conducted to determine which performs best and which performs poorest.

2. Generating Clusters

2.1

Generating clusters for each of the given classes was done using our `generate_data` function. Within this function, MATLAB's `randn` function was used to create normally distributed, uncorrelated, unit variant data. Using the `chol` function to calculate the Cholesky factorization from the given class variance, these values were multiplied to calculate the orthonormal transform of the data and applied to achieve the desired variance and correlation. The class mean was then added to these values to offset our data as desired.

2.2

The samples and unit standard deviation contours of each class can be seen in the figures below. Figure 1 displays classes A and B, while Figure 2 displays classes C, D, and E. Visually, the elliptic unit contours relate to the cluster data by representing the cluster's variance and correlation. For these bivariate distributions, the major axis of the unit contour corresponds to the variance of the first feature, while the minor axis corresponds to the variance of the second feature. The rotation of the contour also demonstrates the correlation of the data. If the contour is not rotated from the horizontal axis, there is no correlation between features. If the contour is rotated clockwise or counterclockwise from the horizontal, the features are negatively or positively correlated, respectively. The centres of these contours are positioned at the true mean of the cluster data. The samples lying on the unit contour are thus known to be 1 standard deviation from the true mean.

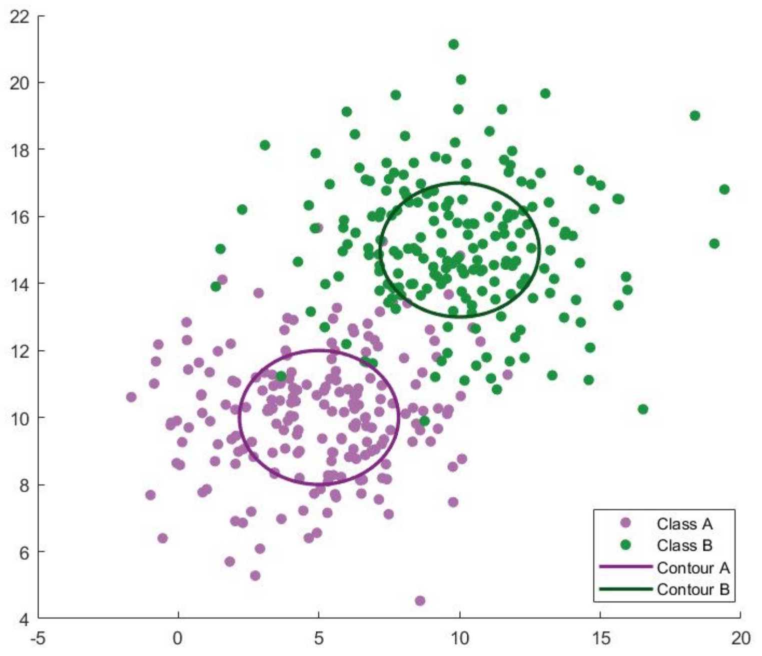

Figure 1. Classes A and B and Respective Unit Standard Deviation Contours

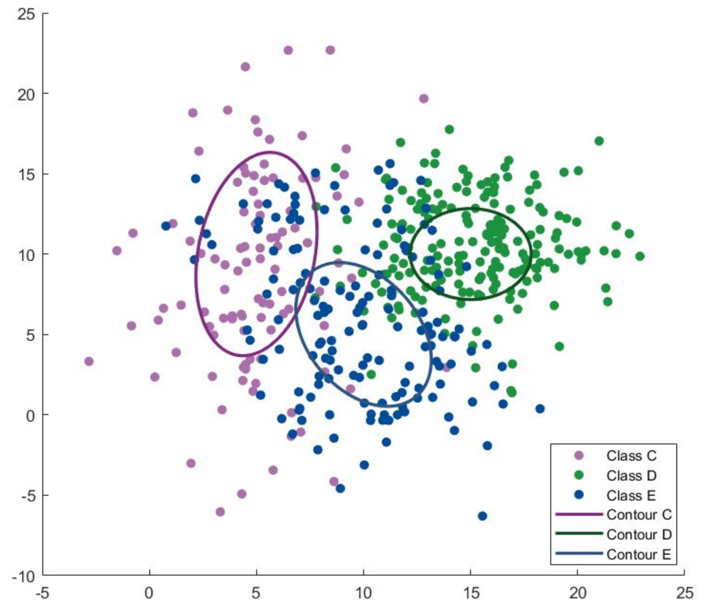

Figure 2. Classes C, D, and E and Respective Unit Standard Deviation Contours

3. Classifiers

To build each of our classifiers, a 2D grid was generated using MATLAB's `meshgrid` function. To generate the points, the grid ranged from the minimum and maximum values of each axis for each of the two cases. Through iteration, all points in this grid were labelled by each classifier as belonging to one of the two classes. Using this classified grid of points, a contour plot was made separating the classes based on their respective classifier decision boundaries.

For Case 1, the decision boundary was found simply between classes A and B, where the point classification was binary and so only one contour was used at level 0. Case 2, contained three classes and required additional analysis. Initially, for each classifier, three boundaries were formed; between classes C and D, C and E, and finally D and E. This resulted in three sets of classified grids (gridCD, gridCE, and gridDE). Our `determine_class` function was used to merge the three decision boundaries into a single grid, by ensuring each point was only a member of a single class. This function assigned a value of -1 to all points that were classified as Class C in both gridCD and gridCE. Similar evaluations were made for Class D, where a value of 0 was assigned to each point that was classified as D in both gridCD and gridDE. Lastly, all points classified as Class E between classes gridCE and gridDE were assigned a value of 1. Using this newly created grid that classified the points to one of three classes, a contour plot at levels -1, 0, and 1 formed the decision boundary.

Calculations of experimental error in this report are as follows:

$$P(\varepsilon) = \frac{Misclassified\ samples}{Total\ Samples}$$

Where *misclassified samples* refer to the number of known test samples assigned to the incorrect class, and *total samples* is the total number of all class samples.

3.1 Minimum Euclidean Distance (MED)

The first classifier examined was the Minimum Euclidean Distance (MED) classifier. MED classifies a point to the class that has the shortest Euclidean distance to the class prototype. In this scenario, the true class means were used as prototypes. Our `get_distance` function was used to calculate Euclidean distance.

$$d_E(\bar{x},\bar{z}) = \left[(\bar{x}-\bar{z})^T(\bar{x}-\bar{z})\right]^{1/2}$$

The above equation calculated the Euclidean distance between $\bar{x}$, an unclassified data point, and $\bar{z}$, the class prototype. To classify an unknown data point $\bar{x}$ between classes 1 and 2 using MED, the following comparison is made.

$$\bar{x} \in c_1 \ iff \ \left[(\bar{x}-\bar{z}_1)^T(\bar{x}-\bar{z}_1)\right]^{1/2} < \left[(\bar{x}-\bar{z}_2)^T(\bar{x}-\bar{z}_2)\right]^{1/2}$$

In the above equation, $\bar{x}$ represents the unknown data point, and $\overline{z_1}$ and $\overline{z_2}$ represent the prototypes of classes 1 and 2, respectively. The point is classified as class 1 if the distance between the point and $\overline{z_1}$ is shorter than the distance between the point and $\overline{z_2}$. Our `get_MED` function thus calculated this as follows:

$$\left[(\bar{x}-\overline{z_1})^T(\bar{x}-\overline{z_1})\right]^{1/2} - \left[(\bar{x}-\overline{z_2})^T(\bar{x}-\overline{z_2})\right]^{1/2} < 0$$

If the distance between $\bar{x}$ and the mean of class 1 is less than that of class 2, the above relation holds and $\bar{x}$ is classified as class 1, otherwise, it is classified as class 2. Using this function, this comparison was done on each point of the 2D grid. For Case 1 this created a decision boundary where the distance of an unknown point to each class mean was equal, meaning the difference between these distances is 0. In the case of 3 classes, the decision boundary was formed if any of the following holds:

$$d_E(\bar{x},\bar{z}_C) = d_E(\bar{x},\bar{z}_D) < d_E(\bar{x},\bar{z}_E) \quad \text{OR}$$
$$d_E(\bar{x},\bar{z}_C) = d_E(\bar{x},\bar{z}_E) < d_E(\bar{x},\bar{z}_D) \quad \text{OR}$$
$$d_E(\bar{x},\bar{z}_D) = d_E(\bar{x},\bar{z}_E) < d_E(\bar{x},\bar{z}_C)$$

For Case 1, the MED decision boundary can be seen in Figure 3 below. As expected, the decision boundary is linear and passes through the midpoint of the class means. This line is perpendicular to the line connecting these means and has a negative slope.

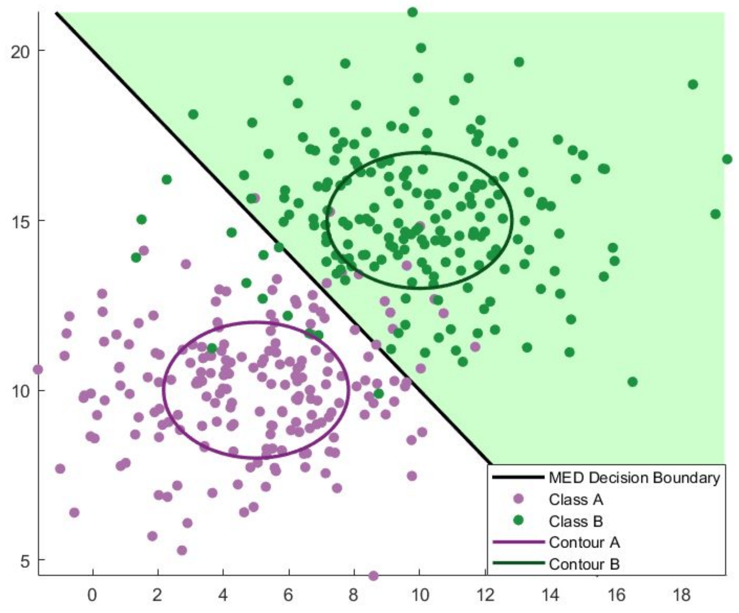

Figure 3. MED Decision Boundary for Case 1 (Classes A and B)

For Case 2, the MED decision boundary can be seen in Figure 4 below for classes C, D, and E. Again, the decision boundaries separating the three classes are linear and located at the perpendicular bisector between the line connecting adjacent class means. The decision boundaries are as follows: vertical separating classes C and D, positively sloped separating classes C and E, and negatively sloped separating classes D and E. The three segments of the decision boundary intersect at the midpoint between the sample means of all three classes.

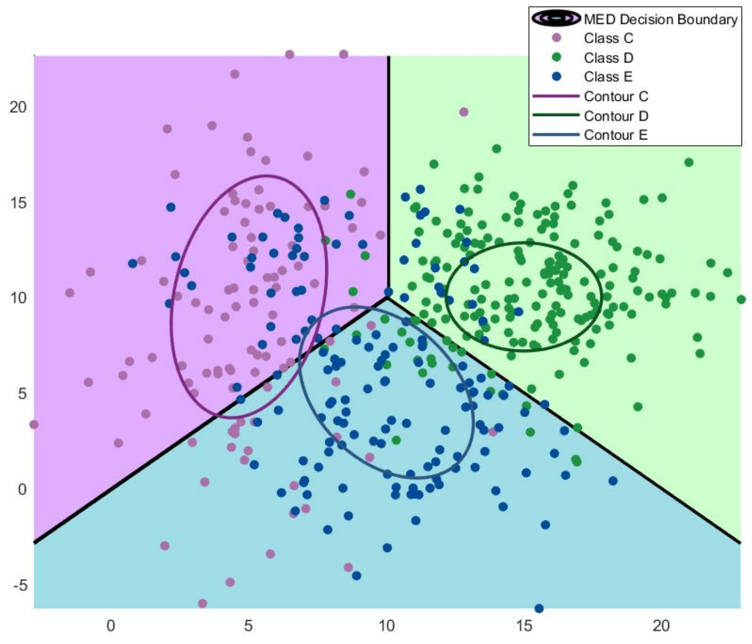

Figure 4. MED Decision Boundary for Case 2 (Classes C, D, and E)

The MED experimental error for Case 1 was calculated as $P(\varepsilon)_{MED,\ Case\ 1} = 0.0700$. The confusion matrix for Case 1 can also be seen below.

		Predicted Class	
		A	B
Actual Class	A	185	15
	B	13	187

The MED experimental error for Case 2 was calculated as $P(\varepsilon)_{MED,\ Case\ 2} = 0.2444$. This error is a significant increase from the 2 class case, which is expected due to the increased complexity of an additional class including the fact that classes C and E have correlated data and all classes overlap to some extent. The MED confusion matrix for Case 2 can be seen below.

		Predicted Class		
		C	D	E
Actual Class	C	74	1	25
	D	4	170	26
	E	33	21	96

3.2 Generalized Euclidean Distance (GED)

The second classifier studied was the Generalized Euclidean Distance (GED)/MinimumIntra-Class Distance (MICD) Classifier. In terms of the distance metrics, the goal of the MICD classifier is to minimize the distance within the class. We can judge the intra-class distance using the mean squared distance within the class. We can relate the MICD classifier to the GED because based on the criterion of the minimum mean squared distance within classes, the GED is the MICD metric. Our `get_MICD_distance` calculated this Generalized Euclidean distance using the following equation:

$$d_G(\overline{x},\overline{z}) = \left[(\overline{x}-\overline{z})^T \Sigma^{-1} (\overline{x}-\overline{z})\right]^{1/2}$$

MICD assigns class membership in a two-class scenario based on the following decision rule:

$$\overline{x} \in A \text{ iff } (\overline{x}-\overline{m}_A)^T \Sigma_A^{-1} (\overline{x}-\overline{m}_A) < (\overline{x}-\overline{m}_B)^T \Sigma_B^{-1} (\overline{x}-\overline{m}_B)$$

In the above rule, $\overline{x}$ represents the unknown pattern, $\overline{m_A}$ and $\overline{m_B}$ represent the mean of classes A and B respectively, and Σ_A and Σ_B represent the covariance matrices of classes A and B, respectively. Thus, each class has its own metric determined by its covariance matrix. For example, Σ_A^{-1} is used to transform $\overline{x}$ to compute the distance metric between $\overline{x}$ and $\overline{m_A}$

For our two-class case, to classify whether each point in our grid belonged to class A or B, the inequality was rearranged and used in our `get_GED` functions:

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

If the difference was greater than zero (i.e. the distance to the mean of class A in the transformed feature space was greater than that of class B), then the point would belong to class B. If the difference was less than zero then the point would belong to class A. The decision boundary of MICD lies on the intersections of the equidistant contours around the classes. The equidistant contour for two classes can be defined as:

$$(\overline{x} - \overline{m}_A)^T \Sigma_A^{-1} (\overline{x} - \overline{m}_A) = (\overline{x} - \overline{m}_B)^T \Sigma_B^{-1} (\overline{x} - \overline{m}_B)$$

For the three-class case, a similar analysis was done as previously described in MED to assign each gridpoint to a single class.

For Case 1, the MICD decision boundary can be seen below in Figure 5 for classes A and B. Since their covariances are equal ($\sum_A = \sum_B$). T), the decision boundary is linear and passes through the midpoint between the means of each class, with a slope that is influenced by $\sum$.

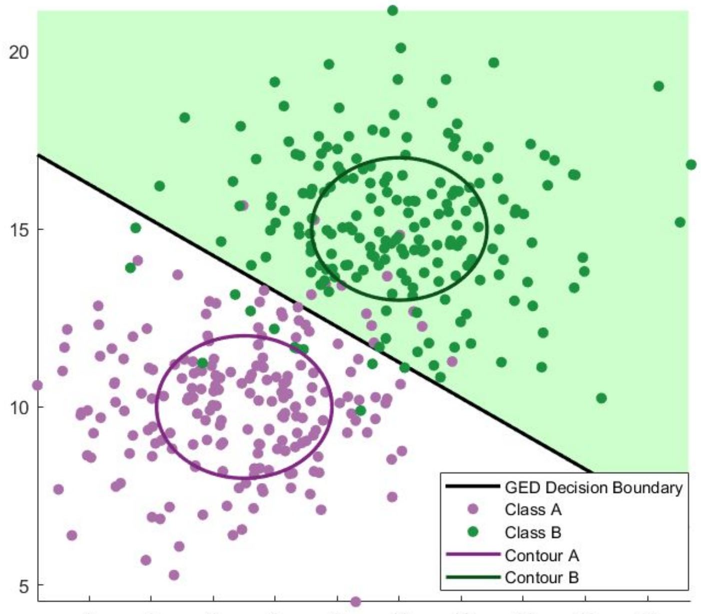

Figure 5. GED Decision Boundary for Case 1 (Classes A and B)

For Case 2, the MICD decision boundary can be seen below in Figure 6 for classes C, D, and E. The decision boundary is non-linear because the covariance matrices are different for each class.

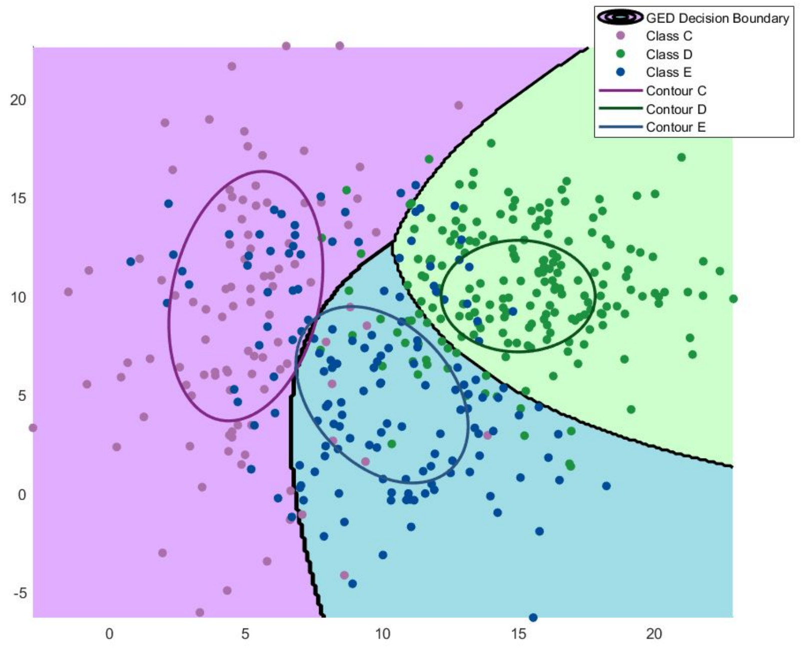

Figure 6. GED Decision Boundary for Case 2 (Classes C, D, and E)

The calculated error for the GED classifier for Case 1 is $P(\varepsilon)_{GED,\ Case\ 1} = 0.0600$. The confusion matrix for Case 1 is seen below.

		Predicted Class	
		A	B
Actual Class	A	187	13
	B	11	189

The calculated error for the GED classifier for Case 2 is $P(\varepsilon)_{GED,\ Case\ 2} = 0.2244$. The confusion matrix for Case 2 is seen below.

		Predicted Class		
		C	D	E
Actual Class	C	91	0	9
	D	6	164	30
	E	40	16	94

3.3 Maximum A Posteriori (MAP)

The third classifier examined was the Maximum A Posteriori (MAP) classifier. MAP is able to assign an unknown pattern to a class based on the highest *a posteriori class* probability. The posteriori class probabilities are defined as $P(A|\bar{x})$ and $P(B|\bar{x})$, and reference the probability of a class membership given a set of measurements. A pattern with a higher posterior probability for A than for B will be classified as belonging to class A, and vice versa. Under this assumption, the MAP classifier can choose the most probable class. Mathematically:

$$\bar{x} \in C1 \; iff \; P(C1|\bar{x}) > P(C2|\bar{x})$$
$$\bar{x} \in C2 \; iff \; P(C1|\bar{x}) < P(C2|\bar{x})$$

Assuming both classes follow a Gaussian distribution, and simplification using Bayes Rule and log-likelihood, we obtain the final comparison used in our `get_MAP` function:

$$\bar{x} \in C1 \; iff \; (\bar{x} - \overline{\mu_{C2}})\Sigma_{C2}^{-1}(\bar{x} - \overline{\mu_{C2}})^T - (\bar{x} - \overline{\mu_{C1}})\Sigma_{C1}^{-1}(\bar{x} - \overline{\mu_{C1}})^T - log\frac{|\Sigma_{C1}|}{|\Sigma_{C2}|} - 2log\frac{P(C2)}{P(C1)} > 0$$

From this simplification, we can see the MAP classifier is just a GED/MICD classifier with the addition of two trailing terms. Thus, MAP is able to minimize intra-class distance as well as incorporate full probabilistic behaviour of the classes by considering prior class knowledge and correlations of the classes. To incorporate this behaviour, our classifier utilizes the `get_MICD_distance` function and then subtracts the prior ratio and covariance ratios to determine whether data point $\bar{x}$ belongs to class 1 or class 2. If the classifier returned a positive value for the data point, the point belonged to Class 1. If the classifier returned a negative value for the data point, it belonged to Class 2. Using this function, a matrix was formed classifying each point in our 2D grid as either class 1 or class 2. The decision boundary lies on the points where $P(C1|\bar{x}) = P(C2|\bar{x})$, in which the MAP classifier would return 0.

The prior probabilities for each class were the ratio between the number of samples in each class and the total number of samples, given by the class statistics.
The prior probabilities were calculated as follows:

$$P(A) = n\_A / (n\_A + n\_B) = 200/(200+200) = 1/2$$
$$P(B) = n\_B / (n\_A + n\_B) = 200/(200+200) = 1/2$$
$$P(C) = n\_C / (n\_C + n\_D + n\_E) = 100/(100+200+150) = 2/9$$
$$P(D) = n\_D / (n\_C + n\_D + n\_E) = 200/(100+200+150) = 4/9$$
$$P(E) = n\_E / (n\_C + n\_D + n\_E) = 150/(100+200+150) = 1/3$$

The linear decision boundary produced using a MAP classifier for Case 1 (Class A and Class B) can be seen in Figure 7 below. By inspection, we can see the boundary is not the perpendicular bisector between the class prototypes (class means) like with an MED classifier, because it accounts for the covariances and posteriori probabilities of each class occurring.

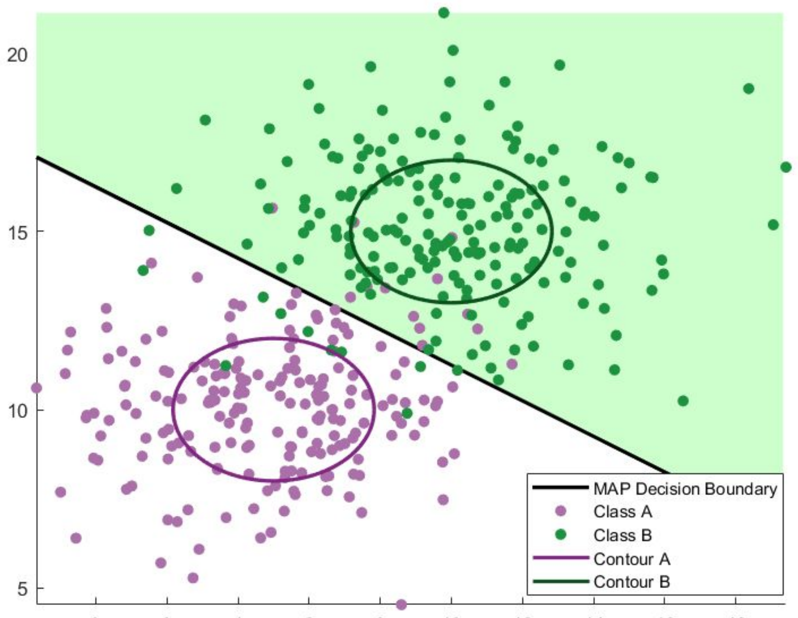

Figure 7. MAP Decision Boundary for Case 1 (Class A and Class B)

The decision boundary produced using a MAP classifier for Case 2 (classes C, D, and E) can be seen in Figure 8 below. Unlike the two-class case, the overlapping three-class classification is more complicated. The decision boundary is no longer linear but instead curves, to take into account the different prior class probabilities and volume distributions.

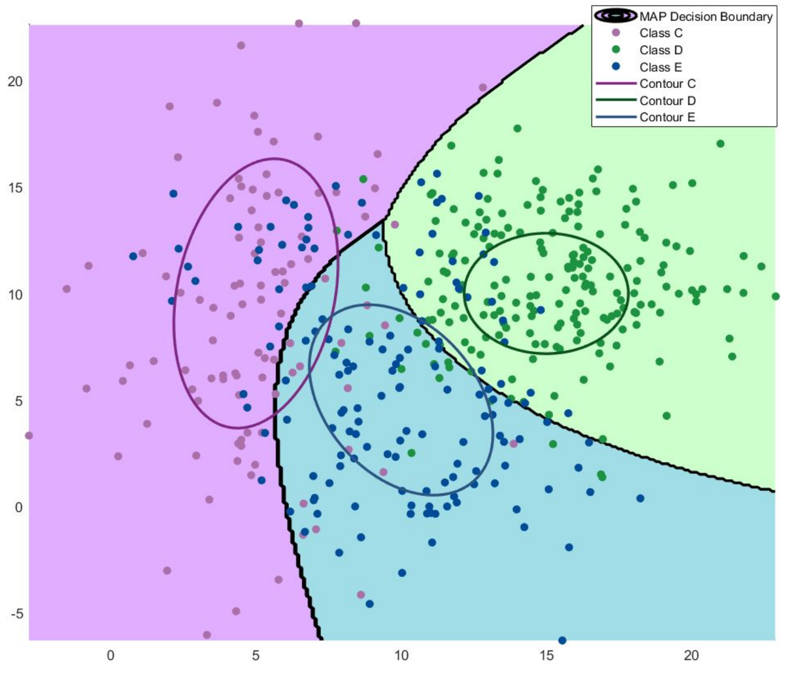

Figure 8: MAP decision boundary for Case 2 (Classes C, D and E)

The MAP experimental error for Case 1 was calculated to be $P(\varepsilon)_{MAP,\ Case\ 1} = 0.0600$. The confusion matrix can be seen below.

		Predicted Class	
		A	B
Actual Class	A	187	13
	B	11	189

The MAP experimental error for Case 2 was calculated to be $P(\varepsilon)_{MAP,\ Case\ 2} = 0.2089$. The confusion matrix can be seen below.

		Predicted Class		
		C	D	E
Actual Class	C	83	1	16
	D	2	179	19
	E	29	27	94

The MAP experimental error for Case 2 was 248% higher than that of Case 1. This significant increase in error was expected due to the complexity of adding an additional class, having more overlap between the classes, as well as Class C and Class E having correlated data.

3.4 Nearest Neighbour (NN)

The Nearest Neighbour (NN) classifier defines class prototypes as the class sample with the minimum Euclidean distance to a given point x of an unknown class. Euclidean distances between x and class prototypes are then evaluated, and x is assigned to the same class as the nearest class prototype. In mathematical terms, this is expressed as:

$$z_k(x) = \bar{x}_k \ \ such\ that \ \ d_E(\bar{x}, \bar{x}_k) = min_i d_E(\bar{x}, \bar{x}_i) \quad \forall\ \bar{x}_i \in c_k$$

Where z_k is the class prototype, chosen based on the position of x. $d_E(\bar{x}, \bar{x}_k)$ is the Euclidean distance between point x and x_k, the sample of class k with the minimum distance to x. As seen above, this distance is defined as the minimum distance between x and a sample point x_i belonging to class k.

Our NN algorithm identified the class prototype in the following steps:

1. Iterate through all samples of a given class
2. For each sample i, calculate the Euclidean distance between the given x and point x_i
3. If the distance to the current sample point is smaller than that of a sample already visited, store this class sample as the class prototype, noting its distance for future comparisons.

For the two-class case, the same process was completed for the second class, resulting in two class prototypes and their respective Euclidean distances. This was expanded into the three-class case, where we repeated the same process for the third class and determined three class prototypes. Point x was then assigned to the class with the minimum Euclidean distance to the class prototype. Since each gridpoint would have individualized class prototypes based on its position in the feature space, class prototypes were calculated for each gridpoint.

The decision boundary was determined to be at points where the Euclidean distance between a given point x and two class prototypes is equal. The equation defining the NN decision boundary for each case is shown below.

Case 1:

$$d_E(\bar{x}, \bar{z}_A) = d_E(\bar{x}, \bar{z}_B)$$

Case 2:

$$\begin{aligned} d_E(\bar{x}, \bar{z}_C) = d_E(\bar{x}, \bar{z}_D) < d_E(\bar{x}, \bar{z}_E) \quad &\text{OR} \\ d_E(\bar{x}, \bar{z}_C) = d_E(\bar{x}, \bar{z}_E) < d_E(\bar{x}, \bar{z}_D) \quad &\text{OR} \\ d_E(\bar{x}, \bar{z}_D) = d_E(\bar{x}, \bar{z}_E) < d_E(\bar{x}, \bar{z}_C) \quad & \end{aligned}$$

Where $d_E(\bar{x}, \bar{z}_k)$ is the distance between the gridpoint x and the class prototype of class k.

The resulting NN decision boundaries for the two and three-class cases are shown below in Figures 9 and 10, respectively. It can be seen that the decision boundaries for both cases are much more precise than the MED, GED and MAP decision boundaries shown previously. This is influenced by the fact that class prototypes are recalculated for each individual point in the feature space to create a precise boundary, as opposed to using sample statistics such as the sample mean or covariance to increase a classifier's generalizability. This also causes the NN classifier to be very sensitive to noise. The runtime for our NN algorithm was significantly slower than our MED, GED and MAP algorithms due to the increased computation.

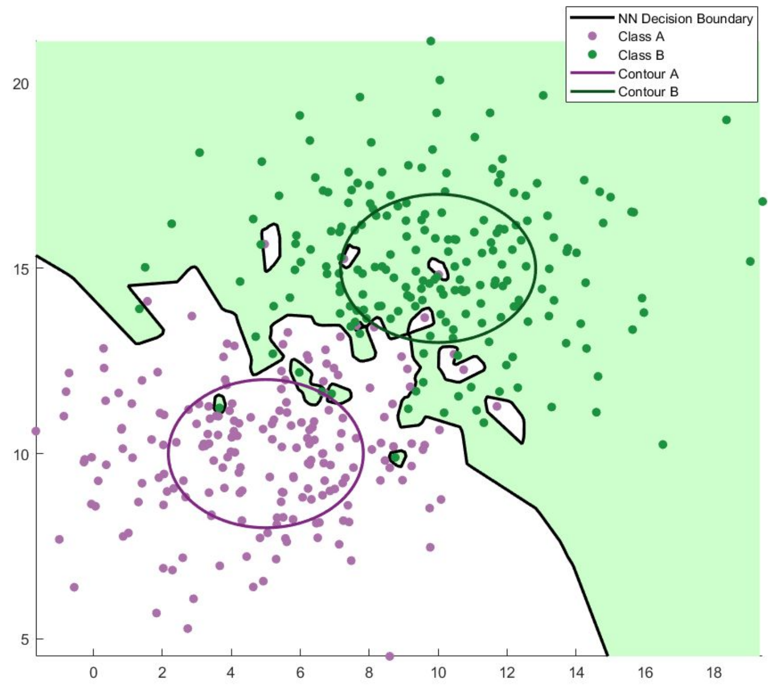

Figure 9. NN Decision Boundary for Case 1 (Clases A and B)

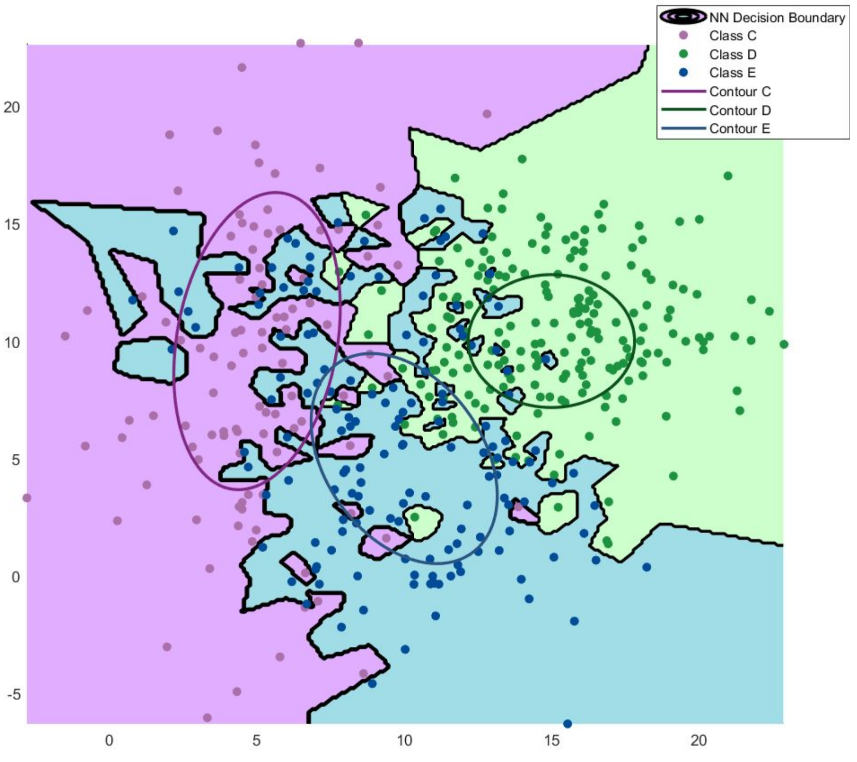

Figure 10. NN Decision Boundary for Case 2 (Classes C, D, and E)

Once decision boundaries were created, a second set of class samples were created for testing. When training, the prediction accuracy was 100%; this is because each class sample considered itself as the class prototype, and thus was always assigned to the correct class. The accuracy was slightly lower in testing, which resulted in the following errors for Case 1:

$$P_A(\varepsilon) = 0.08$$
$$P_B(\varepsilon) = 0.085$$

The overall prediction error of the NN classifier was $P(\varepsilon)_{NN,\ Case\ 1} = 0.0875$. The confusion matrix for Case 1 is presented below.

		Predicted Class	
		A	B
Actual Class	A	182	18
	B	17	183

The NN classification error for each class of the three-class case was found to be:

$$P_C(\varepsilon) = 0.28$$
$$P_D(\varepsilon) = 0.18$$
$$P_E(\varepsilon) = 0.407$$

The overall prediction error for case 2 was $P(\varepsilon)_{NN,\ Case\ 1} = 0.2778$. The confusion matrix for Case 2 is presented below.

		Predicted Class		
		C	D	E
Actual Class	C	72	2	26
	D	6	164	30
	E	33	28	89

3.5 kNN Classifier

The k-Nearest Neighbours (kNN) classifier computes the distance between a given point x and all class samples, and defines class prototypes as the sample mean of the k samples with shortest Euclidean distances. Euclidean distances between x and class prototypes are then evaluated and x is assigned to the same class as the nearest class prototype.

Our kNN algorithm used k = 5 and identified the class prototype in the following steps below:

1. Initialize a matrix to store the 5 class samples with minimum distance to x.
2. Iterate through all samples of a given class.
    a. For each sample point x_i, calculate the Euclidean distance to the given point x.
    b. If the distance to the current sample point is smaller than the largest distance of those stored in our matrix (from previously visited samples), replace the sample.
3. The previous step results in a matrix of the 5 samples closest to point x. The class prototype is then defined as the sample mean of these 5 samples.

This process was completed for the rest of the classes in the feature space, point x was then assigned to the class with the minimum distance to the class prototype. Since class prototypes will vary depending on the position of x in the feature space, class prototypes were calculated for each point in our user-defined grid.

The decision boundary was determined to be at points in the feature space where the Euclidean distance between a given point x and all class prototypes is equal. The equation defining the kNN decision boundary for each case is:

$$d_E(\bar{x}, \bar{z}_A) = d_E(\bar{x}, \bar{z}_B)$$

$$\begin{aligned} d_E(\bar{x}, \bar{z}_C) &= d_E(\bar{x}, \bar{z}_D) < d_E(\bar{x}, \bar{z}_E) \quad \text{OR} \\ d_E(\bar{x}, \bar{z}_C) &= d_E(\bar{x}, \bar{z}_E) < d_E(\bar{x}, \bar{z}_D) \quad \text{OR} \\ d_E(\bar{x}, \bar{z}_D) &= d_E(\bar{x}, \bar{z}_E) < d_E(\bar{x}, \bar{z}_C) \end{aligned}$$

Where $d_E(\bar{x}, \bar{z}_k)$ is the distance between the gridpoint x and the class prototype of class k.

The resulting kNN decision boundaries for the two and three-class cases are shown below in Figures 11 and 12, respectively. It can be seen that the decision boundaries for both cases are much more precise than the MED, GED and MAP decision boundaries, but slightly more generalized than the NN. This latter observation is because the class prototype is defined as a sample mean of the 5 nearest neighbours rather than a single sample. This allows for more generalizability of the algorithm and is reflected in the prediction accuracy. The runtime for our kNN algorithm was slower, but comparable to the NN algorithm due to increased computation and memory requirements.

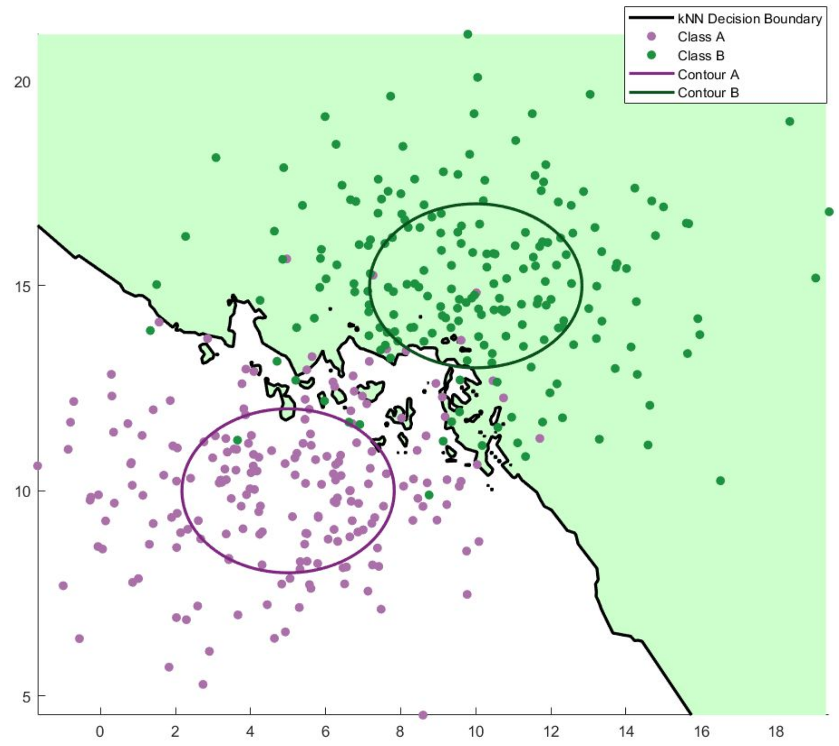

Figure 11. kNN Decision Boundary for Case 1 (Clases A and B)

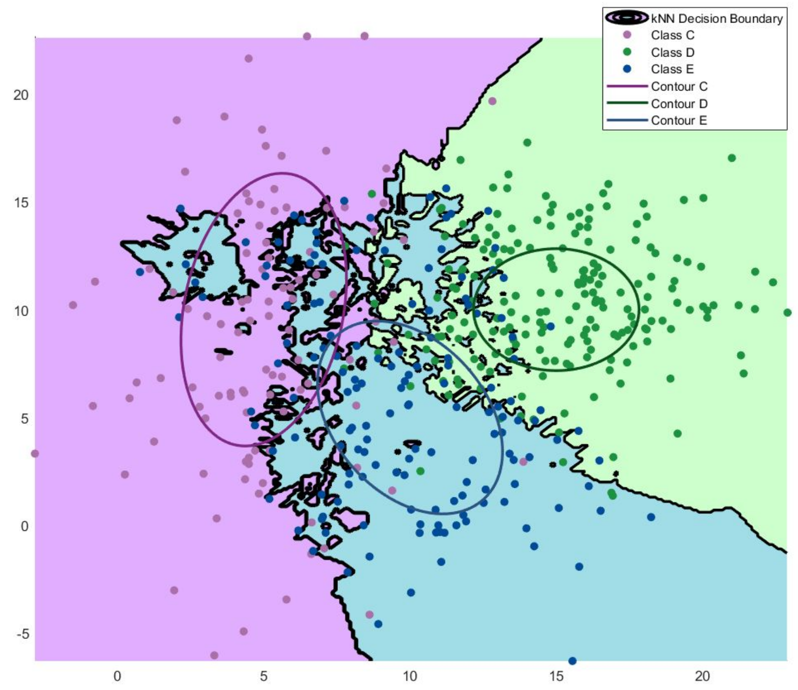

Figure 12: kNN Decision Boundary for Case 2 (Classes C, D, and E)

Once decision boundaries were created, a second set of class samples were created for testing. Confusion matrices for the two and three-class cases are presented below. The kNN classification error for both classes of the two-class case was found to be

$$P_A(\varepsilon) = 0.045$$
$$P_B(\varepsilon) = 0.075$$

The overall prediction error of the 5NN classifier was $P(\varepsilon)_{kNN,\, Case\ 1} = 0.0600$. The confusion matrix for this case is presented below. in the two-class case and 0.2022 in the three-class case.

		Predicted Class	
		A	B
Actual Class	A	191	9
	B	15	185

The kNN classification error for each class of the three-class case was found to be:

$$P_C(\varepsilon) = 0.26$$
$$P_D(\varepsilon) = 0.135$$
$$P_E(\varepsilon) = 0.253$$

The overall prediction error of the 5NN classifier was $P(\varepsilon)_{kNN,\, Case\ 2} = 0.2022$. The confusion matrix for this case is presented below.

		Predicted Class		
		C	D	E
Actual Class	C	74	1	25
	D	2	173	25
	E	22	16	112

Classifier Comparisons

Case 1

Figure 13 below illustrates Case 1 (with classes A and B), plotted with the MED, GED, and MAP decision boundaries. Important to note in this figure is that the GED and MAP decision boundaries are equivalent. This is because both classes have the same number of samples (N = 200), leading to equal prior probabilities (P(A) = P(B) = 0.5), and equal covariance matrices. Visually, all decision boundaries are linear, however, the MED has a steeper slope than the GED/MAP boundaries.

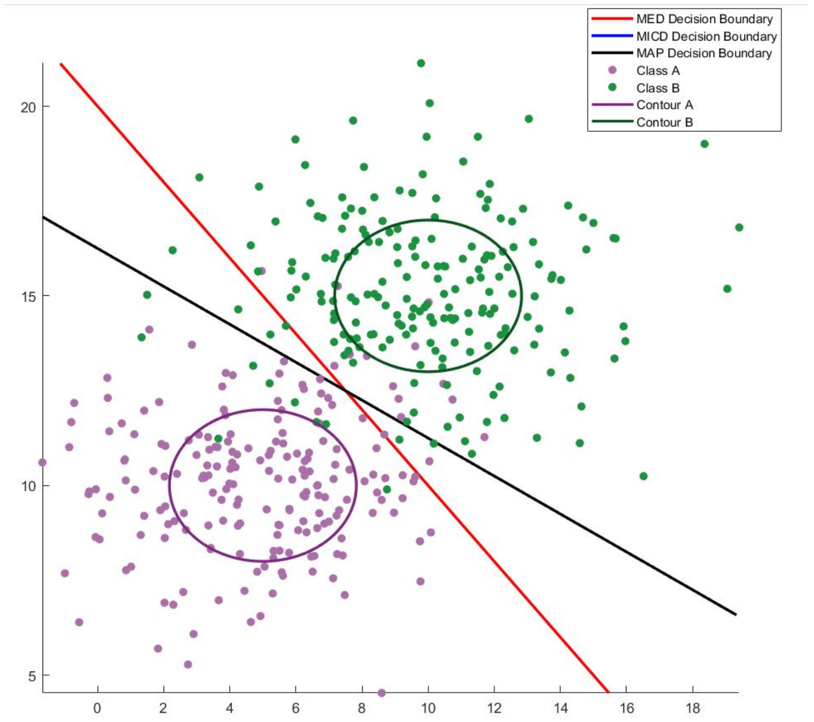

Figure 13. MED, GED, and MAP Decision Boundaries for Case 1 (Classes A and B)

$$P(\varepsilon)_{MED} = 0.0700$$
$$P(\varepsilon)_{GED} = 0.0600$$
$$P(\varepsilon)_{MAP} = 0.0600$$

Comparing the classifier experimental errors, GED and MAP perform identically and are both more accurate than MED. This is attributed to their accountancy of cluster variance.

Figure 14 illustrates Case 1 with classes A and B, plotted with the NN and kNN decision boundaries. As explained in Section 3.4, both NN and kNN created decision boundaries that were much more precise than MED, GED and MAP. However, we can see that the decision boundary created with kNN is slightly more generalized because each class prototype is defined as a sample of the 5 nearest neighbours, rather than a single nearest sample. This leads to overfitting by the NN classifier and makes it more sensitive to noise and outliers, which can be observed through the sectioned-off boundaries created around all of the Class A samples. Because our kNN is created using the sample mean of the 5 nearest neighbours, the overfitting effect is mitigated to some extent, resulting in fewer sub-boundaries. The effect of overfitting on training data and having a decreased ability to generalize also explains why the experimental error for NN (0.0875) was higher than for KNN (0.0600) when tested on new data samples. From the confusion matrices, we can see the number of true positive and true negative classifications increased for the KNN classifier and the number of false positives and false negatives decreased, which is in line with the decreased experimental error.

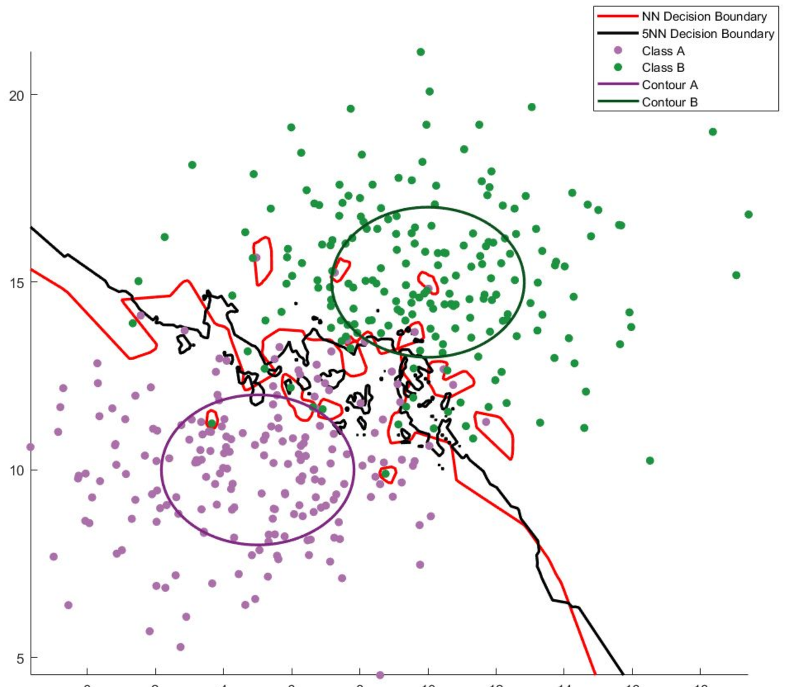

Figure 14. NN and KNN Decision Boundaries for Case 1 (Classes A and B)

Case 2

Figure 15 illustrates Case 2 with classes C, D, and E, plotted with the MED, GED, and MAP decision boundaries. This three-class case introduces more complexity to each of these classifiers. Visually, MED remains linear as expected, while GED and MAP are both nonlinear. The shapes of the GED and MAP classifiers are similar, and appear to more accurately separate the three classes. This is because they utilize more statistical information to separate the classes than MED, such as accounting for the cluster variances. Additionally, the slight difference in the GED and MAP classifiers is accredited to MAP's consideration of each class' prior probabilities and volume cases. We also note that MAP is shifted more towards class C when compared to GED due to the fact that class C has the smallest prior probability, and largest volume case, thus biasing the decision boundary towards it.

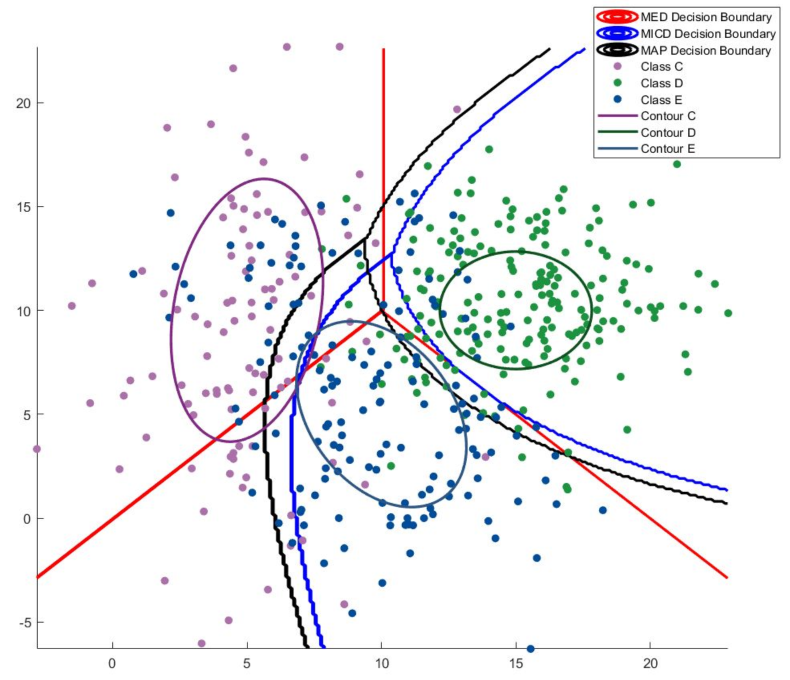

Figure 15. MED, GED, and MAP Decision Boundaries for Case 3 (Classes C,D and E)

$$P(\varepsilon)_{MED} = 0.2444$$
$$P(\varepsilon)_{GED} = 0.2244$$
$$P(\varepsilon)_{MAP} = 0.2089$$

Comparing the classifier experimental errors, MAP performs best, followed by GED, with MED performing poorest. The additional statistics considered in GED and MAP allow for their improved performance.

Figure 16 illustrates Case 2 with classes C, D and E, plotted with the NN and kNN decision boundaries. Again, we can see decision boundaries that are more precise and less general than those created by MED, GED and MAP classifiers. We also observe that the decision boundary created with kNN is slightly more generalized, with fewer sub-boundaries because of the mitigated overfitting effect from using the 5 nearest neighbours instead of the single nearest class sample as the class prototype. This explains why the experimental error for NN (0.2778) was higher than for KN (0.2022) when tested on new data samples. From the confusion matrices, we can see the number of true positive and true negative classifications increased for the KNN classifier and the number of false positives and false negatives decreased, which is in line with the decreased experimental error.

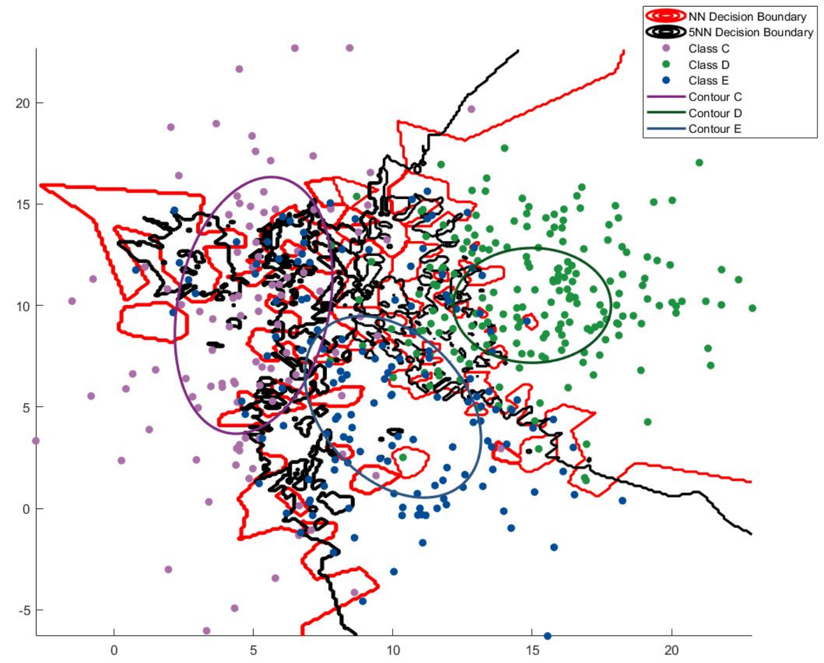

Figure 16. NN and KNN Decision Boundaries for Case 2 (Classes C, D and E)

By examining each confusion matrix in Case 2 a pattern can be seen for the distinguishability of the classes. When looking only at classes C and D, it is clear that all classifiers are able to fairly accurately differentiate between these classes. Very few data points belonging to class C are misclassified as class D, and vice versa. The worst-case scenario appears using NN where 6 samples of class D are misclassified as class C. This is because these two classes are well separated, which is visually evident as shown previously. Another pattern to note is that classes C and D are more frequently misclassified as class E, and vice versa. By looking only at class E, all classifiers misclassify samples as Class C more often than Class D. This can be attested to the fact that classes C and D are relatively well separated, whereas Class E is situated in between them and contributes to most of the overlap between the classes.

Conclusion

Overall, the KNN classifier performed the best out of all the classifiers, followed closely by MAP. For three classes, the decision boundary created by KNN had the lowest experimental error (0.2022) with MAP closely trailing at 0.2089. For two classes, both classifiers had experimental errors of 0.0600. This was expected because the nearest neighbour classifiers have more precise decision boundaries, at the cost of overfitting. Although our KNN was still overfitting to some extent, it was not as extreme as NN and was able to generalize the decision boundary better for the training set of data points. This overfitting of training data explains why NN performed the worst out of all classifiers. MAP was expected to perform well because of its ability to use the full probabilistic behaviour of each class by considering each class's prior probabilities and volumes along with Euclidean distance and covariance metrics.