



University of Waterloo

Department of Mechanical and Mechatronics Engineering

Lab 1: Clusters and Classification Boundaries

SYDE 372

Pattern Recognition

Prepared by -

Abhinav Grover - 20557610

Syeda Zainab - 20550724

Lalit Lal - 20572296

1.0 Introduction

The purpose of this lab was to use different classification methods to classify sets of data and to assess the errors associated with the different methods. Since different types of classifiers work well with different types of data, an understanding of the data itself allows for better judgement on the type of classifier to be used.

In this lab, the data is randomly generated using given true means and covariances. Thus, though the data points themselves will be randomized, the overall shape and structure is pre-defined. MATLAB was used for generation, classification and error analysis of datasets in this lab.

2.0 Generating Clusters

2.1 Implementation

The detailed specifications of each of the classes were given in the Lab outline. Using that data alongside the randn function in matlab, random points with a Gaussian distribution were created for each of the classes. These points will later be used in the classification as well as error analysis and from the 'data' for this lab.

2.2 Results

There were two cases for which the above implementation was run; Case 1 and Case 2. The same cases will be used in future classification and error analysis section as well.

Case 1:

Case 1 consisted of two classes; A and B with the following specifications.

Class A:

$$\begin{aligned}N_A &= 200 \\ \mu_A &= [5 \ 10]^T \\ \Sigma_A &= \begin{bmatrix} 8 & 0 \\ 0 & 4 \end{bmatrix}\end{aligned}$$

Class B:

$$\begin{aligned}N_B &= 200 \\ \mu_B &= [10 \ 15]^T \\ \Sigma_B &= \begin{bmatrix} 8 & 0 \\ 0 & 4 \end{bmatrix}\end{aligned}$$

The above specifications were used with the MATLAB randn function to create random data points with a Gaussian distribution. The data points were plotted for each class and the unit standard deviation contours were plotted on top.

From the covariance matrices, it can be seen that in both the classes, the data is uncorrelated and has the same variances. As such, it is expected there will be no transformation on the unit standard deviation contours. It is further expected that the contour plots will be ellipses rather than circles due to the diagonal elements being different values. Since the means for the classes

are different, it is expected that the ellipses representing the unit standard deviation will, atleast, not sit directly on top of each other.

Figure 2.0 shows the plotted data for each class and its unit standard deviation contour. Each axis of the figure represents a feature.

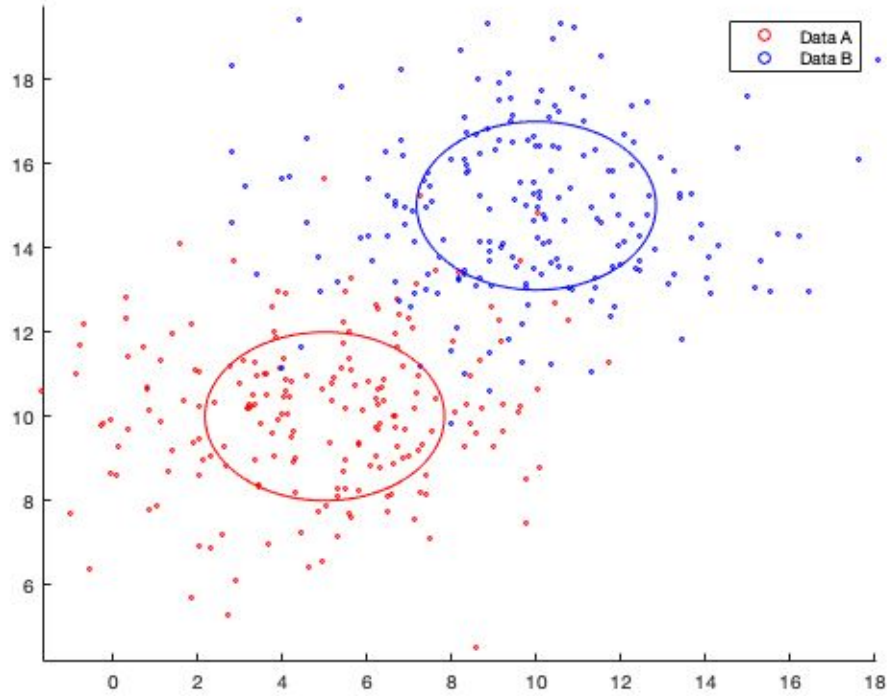


Figure 2.0: Class 1 data and unit standard deviation contour plot

As seen from the above figure, the unit contour is consistent with the fact that the two classes have the same covariance. It is also consistent with the fact that the two features are uncorrelated in both classes, as predicted by the observation of all non-diagonal elements in the covariance matrices being zero.

Case 2:

Case 2 consisted of three classes; C, D and E with the following specifications.

Class C:

$$N_C = 100$$

$$\mu_C = [5 \ 10]^T$$

$$\Sigma_C = \begin{bmatrix} 8 & 4 \\ 4 & 40 \end{bmatrix}$$

Class D:

$$N_D = 200$$

$$\mu_D = [10 \ 15]^T$$

$$\Sigma_D = \begin{bmatrix} 8 & 0 \\ 0 & 8 \end{bmatrix}$$

Class E:

$$N_E = 150$$

$$\mu_E = [10 \ 5]^T$$

$$\Sigma_E = [10 \ -5; \ -5 \ 20]$$

From the above specifications, it is expected that the unit standard deviation contour for class C will be an ellipse, due to the non-equal diagonal elements, with a counter-clockwise tilt due to the positive non-diagonal elements. Class E is expected to be an ellipse with a clockwise tilt and class D will present a perfect circular contour with no transformation due to equal diagonal values in the covariance matrix and all non-diagonal values being zero.

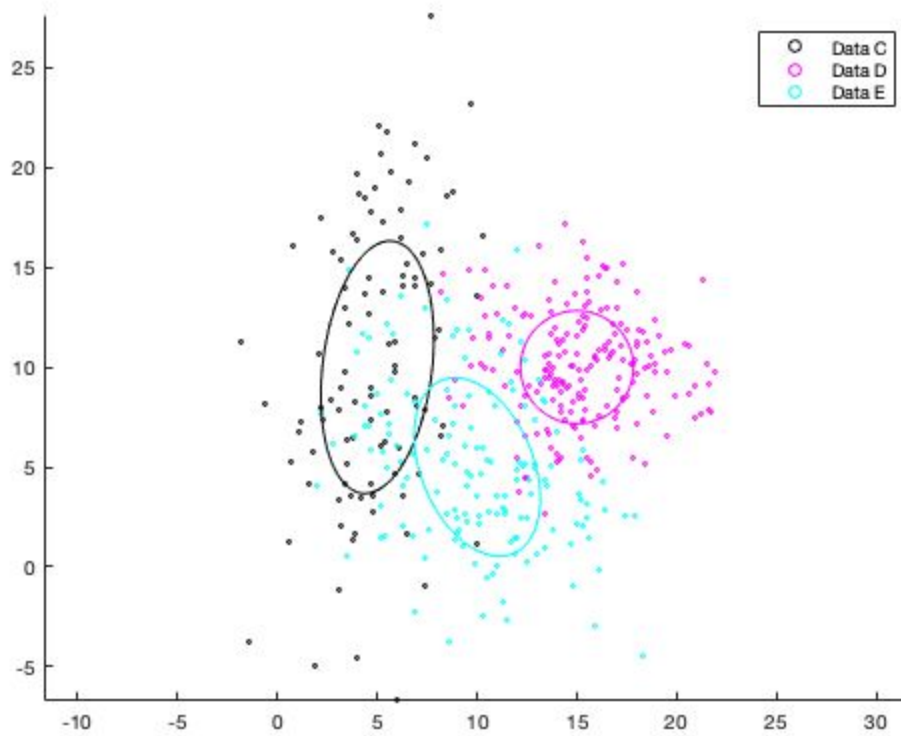


Figure 2.1: Class 2 data and unit standard deviation contour plot

As expected, class C, shown in black, shows both a correlation between the two dimensions (tilted ellipse) and a high variance in the y direction (elongated ellipse). This is consistent with the covariance matrix.

Class D shows no correlation and the same covariance in both dimensions and is represented by a circle which is consistent with the predictions.

Class E shows a clockwise tilt which is consistent with the negative non-diagonal values in its covariance matrix. It also has a high variance in the y direction, similar to class C, as represented by the elongated ellipse.

3.0 Classifiers

3.1 Implementation

Five classifiers were implemented for this lab - MED, GED, MAP, NN, and 5NN. In order to find the classification boundary iteratively, the sample space was quantized into blocks and treated as a matrix.

The implementation for MED, GED and MAP classifiers was very similar with slight changes to the code in order to choose the appropriate one. The implementation then proceeded as follows:

1. For each class in the classification process, create a matrix which spans the quantized sample space
2. Fill the matrix with the distance from each matrix point to the class prototype. Use MED, GED or MAP rules given below to calculate the prototype.
3. Create a new matrix, the same size as the original matrices
4. Fill the indices of the new matrix with integer labels (1, 2, etc) representing the class for which the distance in that particular index is the smallest.
5. If there are two classes which have the same distance for a particular index, give preference to the first class.

The MED, GED and MAP implementations are discussed below.

MED

Given an MED prototype:

$$\left[(X - \mu_A)^T (X - \mu_A) \right]^{\frac{1}{2}} \leq \sum_B^A \left[(X - \mu_B)^T (X - \mu_B) \right]^{\frac{1}{2}}$$

For Class k ,

$$d_k(X, \mu_k) = \left[(X - \mu_k)^T (X - \mu_k) \right]^{\frac{1}{2}}$$

Now,

$$X \in k \quad \text{iff} \quad d_k = \min(d_1, d_2, \dots, d_N)$$

where N is the number of classes

GED

Given the GED prototype:

$$\left[(X - \mu_A)^T \Sigma_A^{-1} (X - \mu_A) \right]^{\frac{1}{2}} \gtrless_B^A \left[(X - \mu_B)^T \Sigma_B^{-1} (X - \mu_B) \right]^{\frac{1}{2}}$$

For class k ,

$$d_k(X, \mu_k) = \left[(X - \mu_k)^T \Sigma_k^{-1} (X - \mu_k) \right]^{\frac{1}{2}}$$

Now,

$$X \in k \text{ iff } d_k = \min(d_1, d_2, \dots, d_N)$$

where N is the number of classes.

MAP

Given the MAP prototype:

$$(X - \mu_B)^T \Sigma_B^{-1} (X - \mu_B) - (X - \mu_A)^T \Sigma_A^{-1} (X - \mu_A) \gtrless_B^A 2 \ln \frac{P(B)}{P(A)} + \ln \frac{|\Sigma_A|}{|\Sigma_B|}$$

For class k ,

$$d_k(X, \mu_k) = (X - \mu_k)^T \Sigma_k^{-1} (X - \mu_k) - 2 \ln(P(k)) + \ln |\Sigma_k|$$

Now,

$$X \in k \text{ iff } d_k = \min(d_1, d_2, \dots, d_N)$$

where N is the number of classes

NN & KNN

The nearest neighbor and k-nearest neighbor classifiers are very similar, where $K = 1$ for the NN case. The implementation is as follows:

1. Create a grid space that spans all data points in each class. Iterate through each grid cell, where each cell is a sample. The chosen grid is from -10 to 25 on both x and y axis, with a step size of 0.1.
2. For each data point in each cluster, determine the euclidean distance to the sample point. Store these distances in a vector, one vector per class.
3. Sort each of the distance vectors in ascending order to determine the K nearest points to the sample. Calculate the mean prototype for each class using the K nearest neighbors.
4. The class with the prototype that has the minimum euclidean distance to the sample point is the chosen class for the sample. Label the grid matrix element with this class indicator.
 - a. Set 1: 1 = A, 2 = B
 - b. Set 2: 1 = C, 2 = D, 3 = E
5. Plot the contour of the labelled matrix, which will show the decision boundary of both sets of classes.

3.2 Results

Case 1: MED, GED & MAP

Figure 3.0 illustrates the decision boundaries created by classifiers MED, GED and MAP for case 1. Case 1 consisted of two classes, A and B, with no correlation between dimensions.

As the figure shows, GED and MAP give the exact same decision boundary, so much so that the GED boundary cannot be seen due to the overlap. This overlap occurs for two reasons. Firstly, because MAP uses the number of points as the probabilities and since the number of points are the same for both classes, the prior is the same. Secondly, the covariances matrices for both classes are the same and, thus, their determinant is the same.

The MED boundary only considers the distance to the mean and not the covariances. Thus, the boundary is linear and equidistant from both the means. Comparatively, the GED/MAP boundary considers the covariances and deviates accordingly. If the two classes had been perfect circles, the MED, GED and MAP boundaries would have all overlapped. Since the classes are slightly elliptical - longer twice as long in the x direction as in the y - the boundary for GED/MAP is tilted to account for it.

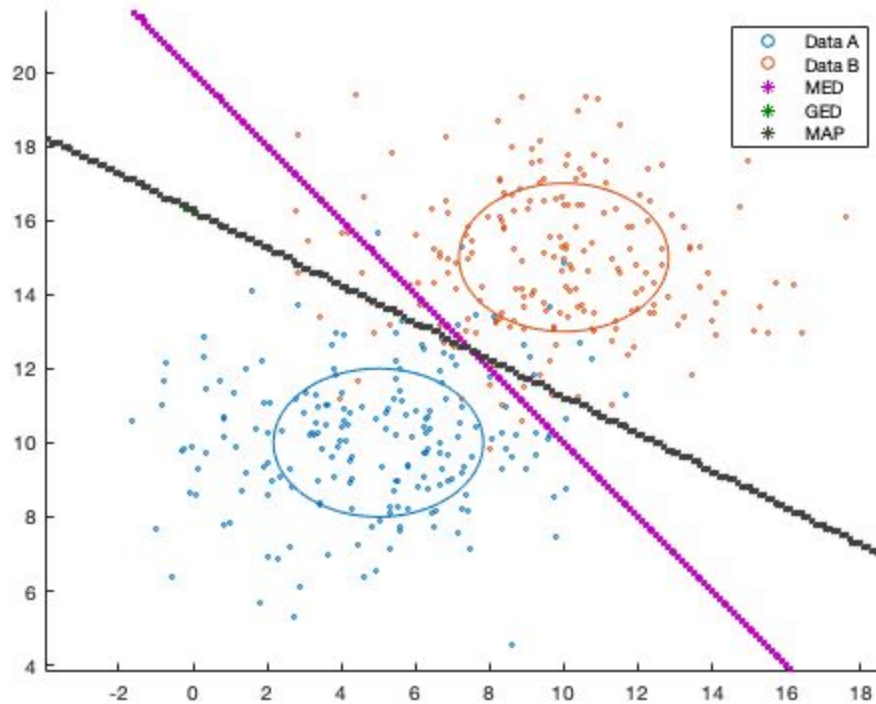


Figure 3.0: MED, GED and MAP decision boundaries for Case 1

Case 2: MED, GED & MAP

The MED, GED and MAP boundaries for the classes in case 2 are shown in Figure 3.1 below. As expected, the MED boundary consists of straight lines which mark the points equidistant from the means of their nearest classes. Since there are three classes, three linear boundaries are observed which all meet at a center point which is equidistant from the means of all three classes. In direct contrast to this are the GED and MAP decision boundaries which are not linear but quadratic. Unlike case 1, the GED and MAP boundaries no longer completely overlap but consist of the same shape with an offset. Unlike the previous case, the covariance matrices are not the same for the three classes which causes a change in the boundaries. Furthermore, the number of points which make up the prior are varying. Since data points prefer the class with a high number of points, the MAP boundary moves away from classes with greater data sets towards the class with the lower set. In this case, this can be seen as the MAP boundary moves towards class C which has the lowest number of points.

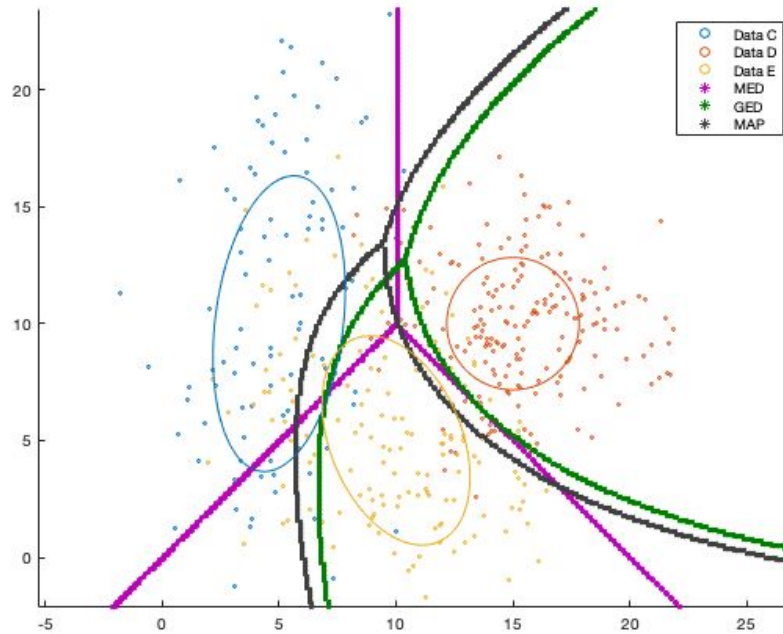


Figure 3.1: MED, GED and MAP decision boundaries for Case 2

Case 1: NN & KNN

In the figures below, the green boundary is the NN classifier decision boundary, and the black boundary is the 5NN classifier decision boundary. By directly comparing these two, it can be seen that the NN boundary is “tighter” than the 5NN boundary.

In case 1, where there are only two classes case (A, B), the NN boundary performs better since is better able to recognize outliers. Simply put, the NN boundary “overfits” the data and tries to perfectly fit a decision boundary. This is noticed as outliers are given their own boundaries in clusters belonging to other classes, as seen in Figure 3.2. Additionally, the 5NN classifier is less accurate, as it is grouping Class B items within the class A boundary, and vice versa.

Case 2: NN & KNN

The performance of NN and KNN classifiers are more exaggerated in the case with three classes. While more computationally complex, it is still maintained that both classifiers, especially compared to the others (such as MAP and GED), are in ways overfitting the data.

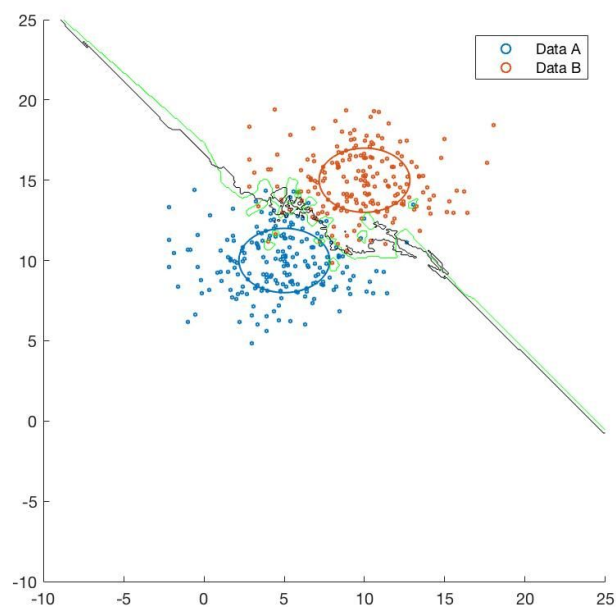


Figure 3.2: NN, 5NN decision boundaries for Case 1

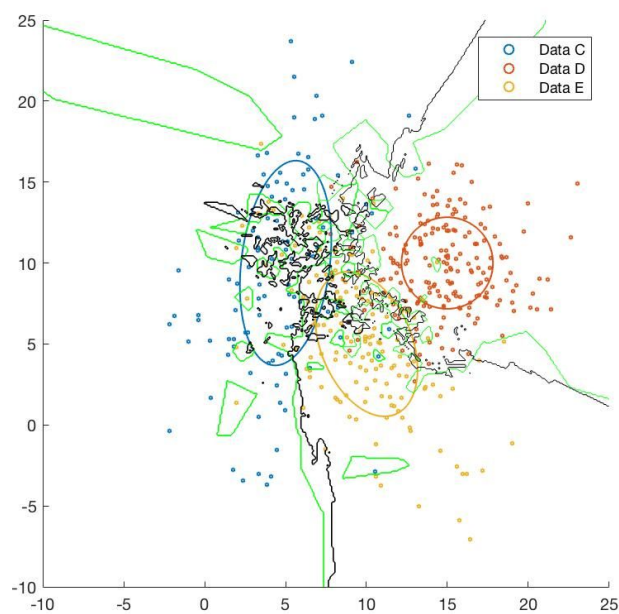


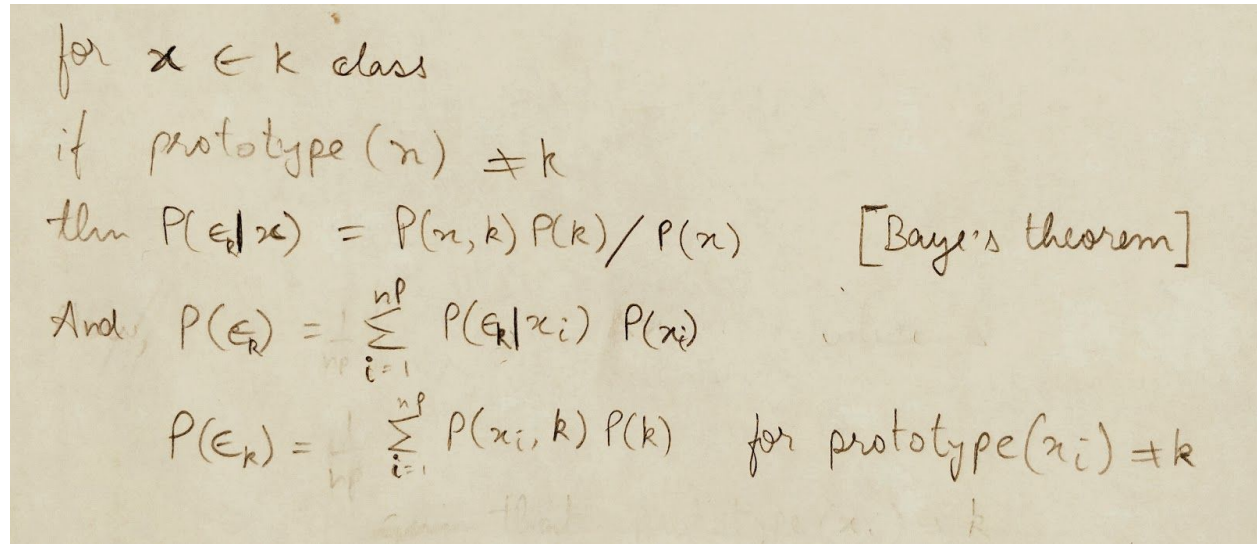
Figure 3.3: NN, 5NN decision boundaries for Case 2

4.0 Error Analysis

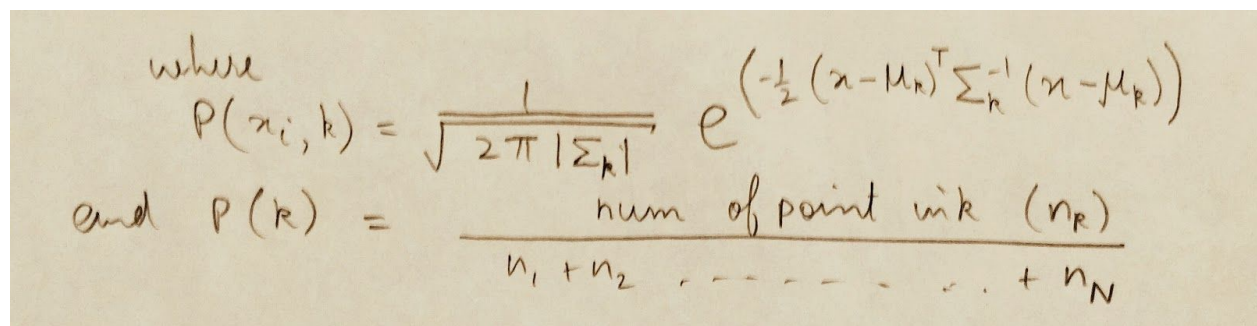
4.1 Implementation

Experimental Error Rate

Here $P(\epsilon_k)$ represents the probability of error in predicting the class k .



for $x \in k$ class
if $\text{prototype}(x) \neq k$
then $P(\epsilon_k | x) = P(x, k) P(k) / P(x)$ [Bayes's theorem]
And, $P(\epsilon_k) = \frac{1}{n_p} \sum_{i=1}^{n_p} P(\epsilon_k | x_i) P(x_i)$
 $P(\epsilon_k) = \frac{1}{n_p} \sum_{i=1}^{n_p} P(x_i, k) P(k)$ for $\text{prototype}(x_i) \neq k$
again that



where
 $P(x_i, k) = \frac{1}{\sqrt{2\pi |\Sigma_k|}} e^{\left(-\frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k)\right)}$
and $P(k) = \frac{\text{num of point in } k (n_k)}{n_1 + n_2 + \dots + n_N}$

Thus the net probability of error for the prototype would be:

$$P(\epsilon) = P(\epsilon_1) + P(\epsilon_2) + \dots + P(\epsilon_N)$$

This is how the following probability results have been computed.

Confusion Matrix

MATLAB already contains a function named `confusionmat` which creates a confusion matrix given the true labels and the predicted labels. The implementation of the confusion matrix consisted of creating a matrix with all the data points and their true labels. Then, these data points were compared to the output of the MED, GED and MAP decision boundaries to find their predicted labels. Finally, the true labels and predicted labels were fed to the `confusionmat` function to get the matrices.

4.2 Results

Experimental Error Rate

The results of the experimental error analysis are shown in Table 1. As expected for case 1, the error for MED is larger than that of GED and MAP which both have the same error. This is because for case 1, the decision boundaries for GED and MAP overlapped thus the classifications were identical.

Table 1: $P(\epsilon)$ for MED, GED and MAP per case

	MED	GED	MAP
Case 1	0.0383	0.0323	0.0323
Case 2	0.0498	0.0513	0.0385

For case 2, GED has a higher error than MED. This is unexpected as GED is expected to be more accurate than MED as it takes into account the variance of the data. The lowest error, as expected, is produced by the MAP classification which caters the decision boundary not only to the mean and covariance of the data but also to the probability of occurrence which, in this particular case, was the number of data points present in the class.

Confusion Matrix

Table 2 shows the resultant confusion matrices for case 1 and 2 for MED, GED and MAP classifiers. Since MED is the least accurate classifier, taking into account only the mean of the class, it has the lowest accuracies in both case 1 and 2.

However, in case 1, as Figure 3.0 illustrated, the decision boundaries for GED and MAP are the same. This implies that the accuracy of classification would also be the same. This can be seen in the table below where GED and MAP both have accuracies of 92.5% for case 1.

Table 2: Confusion Matrices for MED, GED and MAP

	MED	GED	MAP
Case 1	<div>ConfmatABmed =</div> <div><div>18515</div><div>21179</div></div> <div>Accuracy: 91.0%</div>	<div>ConfmatABmap =</div> <div><div>18713</div><div>17183</div></div> <div>Accuracy: 92.5%</div>	<div>ConfmatABged =</div> <div><div>18713</div><div>17183</div></div> <div>Accuracy: 92.5%</div>

Case 2	ConfmatCDEmed =	ConfmatCDEged =	ConfmatCDEmap =
	77 1 22	94 0 6	85 1 14
	6 179 15	6 169 25	3 188 9
	32 18 100	39 16 95	29 19 102
	Accuracy: 79.11%	Accuracy: 79.55%	Accuracy: 83.33%

Case 2 represents a more complicated scenario where there are three classes rather than just two. For this case, the MED has the lowest boundary and GED has only slightly higher. MAP outperforms both the other classifiers with 83.33% accuracy. This is expected as MAP accounts for the number of data points (used as probabilities) in the class as well as the covariance matrix. GED does not account for the number of data points and MED does not account for anything other than the mean.

5.0 Conclusion

Overall, based on the results of the error analysis, it can be concluded that MAP classification gives lower error and better accuracy. This is seen in section 4 where in both cases, MAP outperforms the other classifiers consistently and by a good margin.

This makes logical sense as the more information about the data that a classifier considers in its decision making, the better the classification is likely to be. It is important to realize that although MAP is the better classifier, there may be situations where MED or GED are equally accurate. An example of this is case 1 where due to equal covariance matrices and number of points, MAP and GED had equivalent decision boundaries.

In such cases, it would be prudent to opt for the simpler classifier.