Pattern Recognition

Assignment 1

CS 479/679 Pattern Recognition (Spring 23)
Programming Assignment 1

Shankar Poudel & Aminul Huq
Contribution: Equal Contribution in both coding as well as report writing.

Computer Science and Engineering Department
University of Neavada, Reno

February 27, 2023

1 Part 1: Theory

Bayes classifier can be represented by a set of discriminant functions for each classes/states and as per the maximum discriminant value corresponding to the state, prediction can be carried out. Considering a scenario where we have more than one features, c number of state of nature, discriminant function using Bayes' formula in case of zero-one loss can be written as:

$$g_i(\mathbf{x}) = lnP(\mathbf{x}|w_i) + lnp(w_i) \tag{1}$$

Where w_i is the state of nature from $i = 1, ..., c$ and $\mathbf{x}$ is a D dimensional feature vector. If we assume that $P(\mathbf{x}|w_j)$ is sampled from a multivariate normal distribution i.e. $P(\mathbf{x}|w_j) \sim N(\mu_i, \Sigma_i)$ then we can re-write equation 1 as,

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \mu_i)^t \Sigma_i^{-1}(\mathbf{x} - \mu_i) - \frac{d}{2} ln2\pi - \frac{1}{2} ln|\Sigma_i| + lnp(w_i). \tag{2}$$

Here, μ and Σ represent the mean and covariance matrix respectively.
The complexity of the discriminant here depends on the covariance matrix. As per the different considerations for the covariance matrices there are three special cases which can be considered from equation (2). These are:

1. Case 1: $\Sigma_i = \sigma^2\mathbf{I}$:
   In this scenario the features are uncorrelated and have same covariance matrices. So the data distribution is spherical in shape and are of same size. Here, we will always have linear discriminants. Considering the two classes condition, the decision boundary created will also be linear and perpendicular to the line joining the means. The decision boundary for this situation can be defined as,

   $$\mathbf{w}^t(\mathbf{x} - \mathbf{x_0}) = 0 \tag{3}$$

   where, $\mathbf{w} = \mu_i - \mu_j$ and $\mathbf{x_0} = \frac{1}{2}(\mu_i + \mu_j) - \frac{\sigma^2}{\|\mu_i - \mu_j\|^2} \ln \frac{P(w_i)}{P(w_j)}(\mu_i - \mu_j)$

2. Case 2: $\Sigma_i = \Sigma$:
   The data doesn't have same variance in this case and the distribution is of hyper-ellipsoidal in shape. Here too, we will have linear discriminants. For two class cases, we will have linear decision boundary but the hyper-planes are not perpendicular to the line joining the centers. The decision boundary for this scenario can be defined as,

   $$\mathbf{w}^t(\mathbf{x} - \mathbf{x_0}) = 0 \tag{4}$$

   where, $\mathbf{w} = \Sigma^{-1}(\mu_i - \mu_j)$ and $\mathbf{x_0} = \frac{1}{2}(\mu_i + \mu_j) - \frac{\ln \frac{P(w_i)}{P(w_j)}}{(\mu_i - \mu_j)^t \Sigma^{-1}(\mu_i - \mu_j)}(\mu_i - \mu_j)$

3. Case 3: Σ_i = arbitrary:
   Each category has their own covariance matrix in this case so the data distributions are of different sizes and shapes. In this case, we will have quadratic discriminants and decision boundary between two-classes will be hyperquadratic in nature. It's decision boundary is,

$$\mathbf{x^t(W_1 - W_2)x + (w_1^t - w_2^t)x + (w_{1,0} - w_{2,0}) = 0} \tag{5}$$

where, $\mathbf{W_i} = -\frac{1}{2}\Sigma_i^{-1}$, $w_i = \Sigma^{-1}\mu_i$ and $w_{i,0} = -\frac{1}{2}\mu_i^t\Sigma_i^{-1}\mu_i - \frac{1}{2}\ln|\Sigma_i| + \ln P(w_i)$

In two class cases, one can find the discriminant vales for each of the classes and predict the class with highest discriminant value for given features. It is equivalent to finding the decision boundary function for two-class case and predicting it to be of a class if decision boundary function yields value less than zero and to be of next class if it yields greater than zero.

Euclidean Distance Classifier:
We can obtain the Euclidean distance classifier from Case 1. As this is a special scenario of Case 1. The discrimant from Case 1 is used in this scenario as well however we have to assume that the prior probabilities are same in this case i.e. $p(w_i) = p(w_j)$. This provides the following equation which is called Euclidean distance classifier,

$$g_i(x) = -\|\mathbf{x} - \mu_i\|^2 \tag{6}$$

where, $\|\mathbf{x} - \mu_i\|^2 = (\mathbf{x} - \mu_i)^t(\mathbf{x} - \mu_i)$.

Decision Boundary:
Discriminants divide the feature space into various decision regions R1, R2,...RN, which are separated by decision boundries. Using the various conditions mentioned above, we can determine the discriminants. Then to get the decision boundary, we need to find the locus where the discriminants of neighbouring decision regions equals.

Classification Error and Error Bound:
In real life data, no pattern recognition methodology is perfect and so the there will be errors. We can use misclassification rate as one of the measure for error. It is simply the ratio of number of wrongly classified data by total data.

$$Misclassificationrate = \frac{FP + FN}{TP + TN + FP + FN} \tag{7}$$

Where, TP = True Positive, FP = False Positive, TN = True Negative, FN = False Negative.

Considering the true class conditional densities to be in Gaussian distribution, we can estimate the theoretical bound of the error. With the help of Chernoff bound we can know about the tight error bound. We can use Bhattacharyya error bound which is a simpler and easier to compute error bound. The following equation can be used to calculate the Bhattacharyya error bound:

$$P(error) \leq P^{\beta}(w_1)P^{(1-\beta)}(w_2)\exp^{-k(\beta)} \tag{8}$$

where, $\beta = 0.5$ is to be set for the Bhattacharyya error bound and we can get the value of $k(\beta)$ from the equation below:

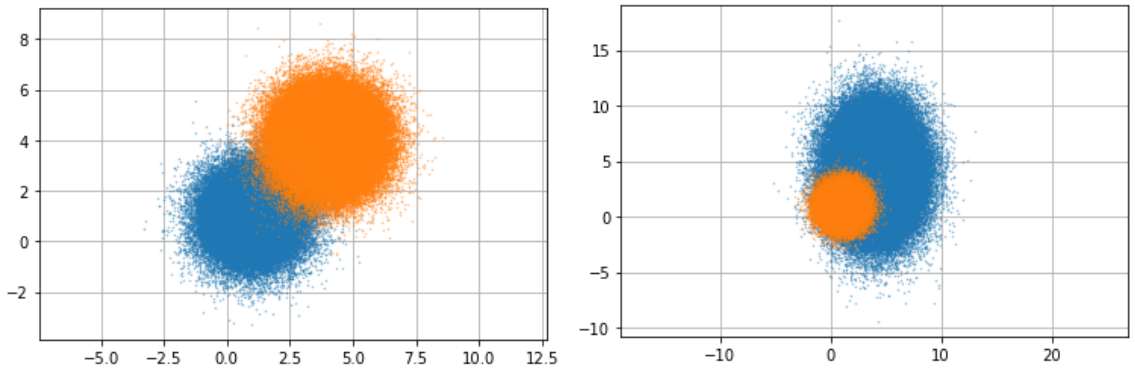
$$k(\beta) = \frac{\beta(1-\beta)}{2}(\mu_1 - \mu_2)^t[(1-\beta)\Sigma_i + \beta\Sigma_2]^{-1}(\mu_1 - \mu_2) + \frac{1}{2}\ln\frac{|[(1-\beta)\Sigma_i + \beta\Sigma_2]|}{|\Sigma_1|^{1-\beta}|\Sigma_2|^{\beta}} \quad (9)$$

here, μ and Σ represents the mean and covariance of two classes.

2 Part 2. Results ans Discussion

2.1 Data Generation

In order to generate data from 2D Gaussian density we took help from the numpy python library [1]. It can take inputs of mean as a 1-D array and covariance matrix as a 2D matrix where the covariance matrix is of $N \times N$ shape. This approach also takes input of how many output samples we want i.e B and it will return a $B \times N$ shaped array of samples drawn from the given distribution. Figure 1 shows the visualization of the two datasets.



(a) (b)

Figure 1: Visualization of (a) Dataset A (b) Dataset B.

2.2 Question 1

2.2.1 a.

The data we have here (data set A) is generated using multivariate Gaussian distribution so we designed discriminant function for the classifier and thus the dichotomizer assuming the multivariate Gaussian density. We used the first case of the creation of discriminant for multivariate Gaussian distributed data as the covariance of both the classes are same and all the elements in the diagonal of the covariance matrix is also equal to each other.
We have 60,000 random data from class0 and 140,000 random data from class1 with total being 200,000 data in set A. We are assuming the data in class A is generated as the results of random experiments. Thus we can set the Priors $P(\omega 1)$ and $P(\omega 2)$ to be 60,000/200,000 and 140,000/200,000 respectively.

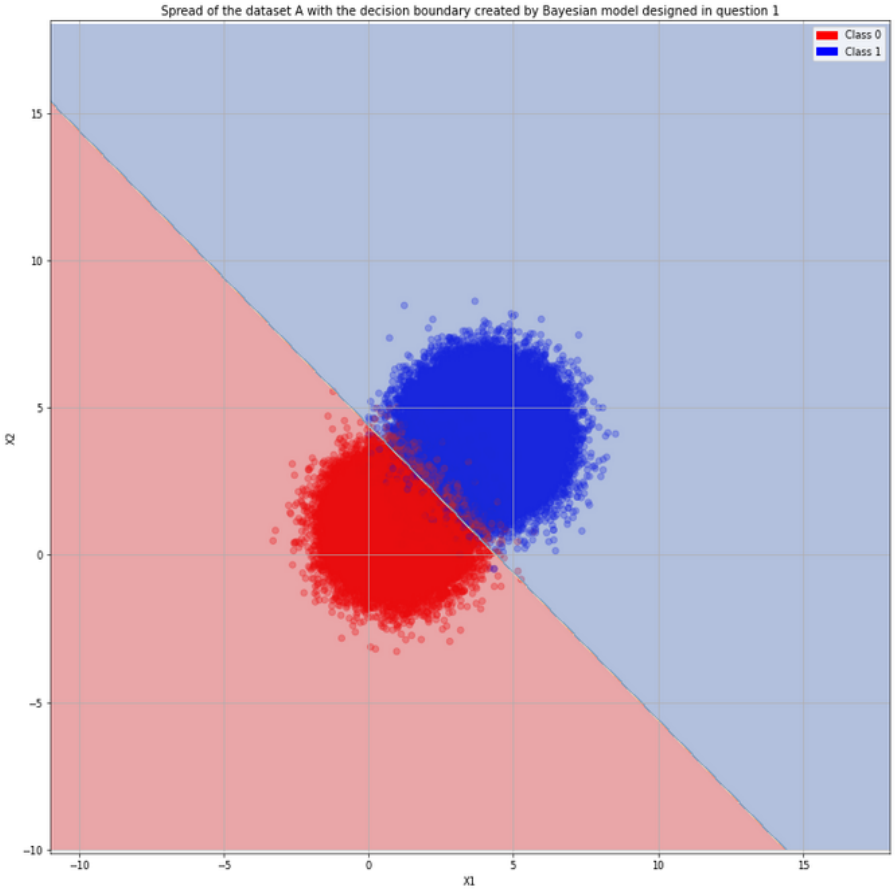


Figure 2: Spread of the dataset A with the decision boundary created by Bayesian model designed in question 1.

2.2.2 b.

The scatter plot of randomly generated samples from two Gaussian distributions in Set A and the decision boundary generated by the Bayesian classifier created in Question 1a is shown in Figure 2.

2.2.3 c.

Using the model created in Question 1a, all the data in set A is classified and following misclassification rates are obtained.

1. Misclassification rate in class 0: 4.4406%
2. Misclassification rate in class 1: 0.5492%
3. Total misclassification rate: 1.7064%

2.2.4 d.

We calculated the Bhattacharya Error Bound (i.e. assuming setting $\beta = 0.5$ in Chernoff Error Bound) to be 4.8299%. This the the maximum error bound we can get considering the true class conditional densities are Gaussian. Here data set A is generated following the condition required so this bound should be reliable. And, looking result from Question 1c., where we got the total misclassification rate to be 1.7064%, we are below the bound.

2.3 Question 2

2.3.1 Repeat of experiment as in Question 1

The data we have here (data set B) is generated using multivariate Gaussian distribution so we designed discriminant function for the classifier and thus the dichotomizer(as we have case of two categories/classes) assuming conditional densities to follow multivariate Gaussian density. We used the third case of the creation of discriminant as the covariance matrix of the two classes are different from each other.We have 60,000 random data from Class0 and 140,000 random data from Class1 with total being 200,000 data in set A. We are assuming the data in class A is generated as the results of random experiments. Thus we can set the Priors $P(\omega 1)$ and $P(\omega 2)$ to be 60,000/200,000 and 140,000/200,000 respectively.
The scatter plot of randomly generated samples from two Gaussian distributions in Set B and the decision boundary generated by the Bayesian classifier created is shown in Figure 3.

Using the model created, all the data in set B is classified and following misclassification rates are obtained.

1. Misclassification rate in class 0: 8.3783%
2. Misclassification rate in class 1: 7.3985%
3. Total misclassification rate: 7.6925%

We calculated the Bhattacharya Error Bound (i.e. assuming setting $\beta = 0.5$ in Chernoff Error Bound) for the data set B to be 20.3539%. This the the maximum error bound we can get considering the true class conditional densities are Gaussian. Looking result from Question 1c., where we get the total misclassification rate to be 7.6925%, we are below the bound.

2.3.2 Result Comparison

Data in set A is generated from two Gaussian distribution for two classes with same covariance matrix but at two mean/center points. So, using the first case of discriminant creation, the decision boundary is linear and perpendicular to the line joining the two centers. Also, as the P($\omega 1$) < $P(\omega 2)$, the decision boundary plane is nearer to center of first class (i.e. Class0).
Whereas, data set B is generated from two Gaussian distribution for two classes with different covariance matrix, and one of the distribution has non-equal diagonals in the covariance

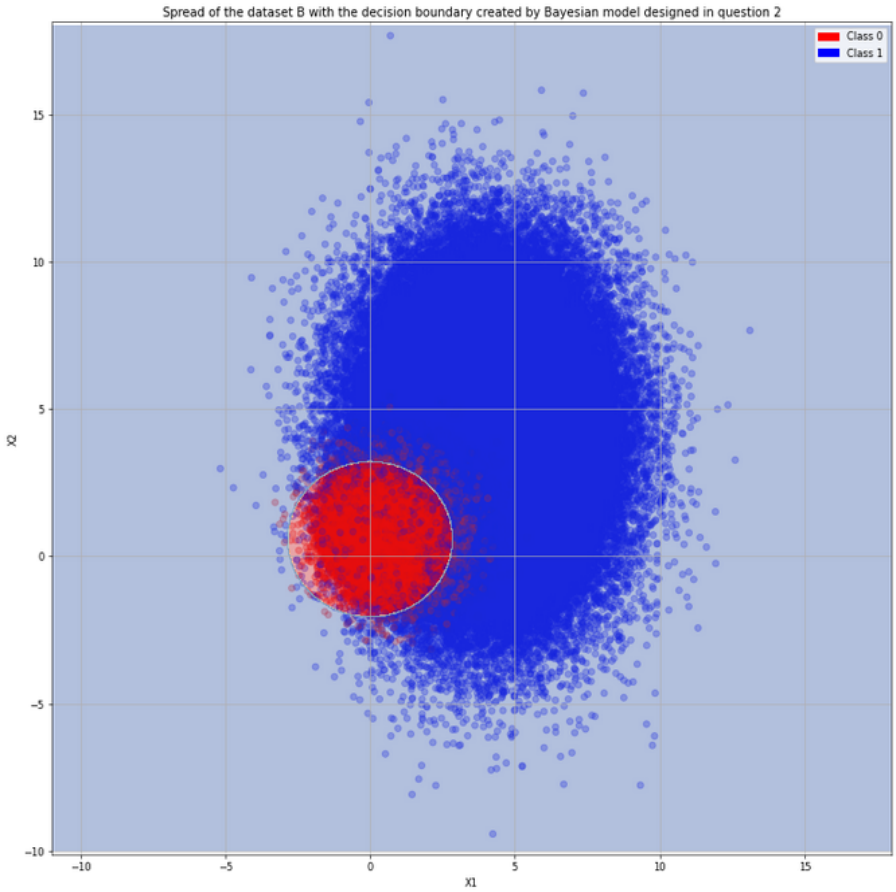


Figure 3: Spread of the dataset B with the decision boundary created by Bayesian model designed in question 2.

matrix. This gave the data spread in one of the class to be elliptical. We used the third case of creation of discriminant for the Bayesian Classifier. This gave a non-linear (almost spherical) decision boundary. Class0 is tight whereas class1 is widely spread which can be interpreted from the covariance matrix as well as seen in figure 3. Thus, using the Bayesian Classifier we get the decision boundary generalizing the aforementioned fact.

2.4 Question 3

As mentioned in the theory part, equation (2) is a discriminant function in which $P(\mathbf{x}|w_i)$ is drawn from a multivariate normal distribution $N(\mu_i, \Sigma_i)$. Euclidean distance classifier is an optimal one in cases where we have same covariance matrix for all the classes, the clusters of data should be spherical and the features are uncorrelated. Also this classifier can perform well in case the prior probabilities are equal i.e. $p(w_i) = p(w_j)$. For data set A, we have different mean values and same covariance matrix which is a diagonal matrix which tells us that the data distribution will be spherical in shape, centering at the mean values and will be of equal size. At the same time we can also say because we have diagonal matrix then the features are uncorrelated as well. However, one thing that this dataset doesn't satisfy is

that the priors won't be equal because we 60,000 data of Class 0 and 140,000 data of Class 1.

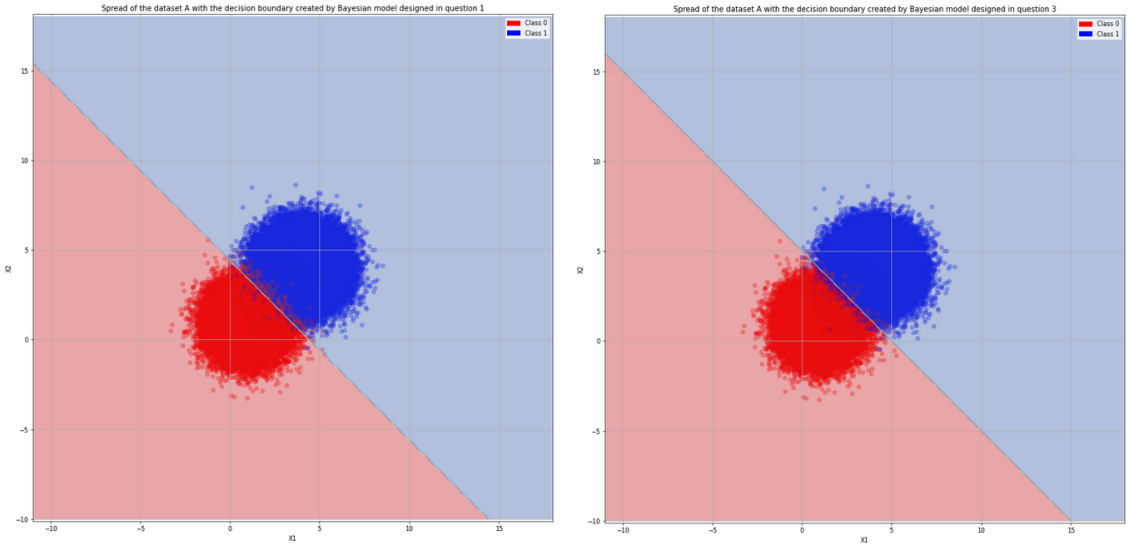
2.4.1 Experimental Results

In this section, we will discuss about the results that were obtained on data A after using Euclidean Distance Classifier. Table 1 shows the misclassification rates in scale of 100 that were obtained after using Euclidean distance classifier and Bayes Classifier for Case I.

Table 1: Comparison of different classifiers in a scale of 100.

Misclassification Rates	Class 0	Class 1	Total	Error Bound
Euclidean distance classifier	1.6866	1.7635	1.7405	4.8299
Bayes Classifier : Case I	4.4066	0.5492	1.7065	

From Table 1 we can see that Euclidean distance classifier achieved 1.6866%, 1.7635% and 1.7405% of misclassification rates for class 0, class 1 and for the whole data respectively. The decision boundary generated by Euclidean distance classifier can be seen in figure 4.



(a) (b)

Figure 4: For the data set B, (a) shows the decision boundary generated by Bayes Classifier and (b) shows decision boundary created by the Euclidean Distance Classifier.

2.4.2 Comparison and Discussion

In comparison to Euclidean distance classifier, Bayes Classifier for Case I approach performed better overall in terms of total misclassfiation rates. Euclidean distance classifier attained 1.7405% of misclassification rate while the Bayes classifier achieved 1.7065%. We believe that Euclidean distance classifier might have done better if the prior probabilities of the

dataset A were equal because this is one of the assumptions of Euclidean distance classifier which is not met by the dataset A. We can also see that the misclassification rates of class 1 is less than class 0 in case of Bayes Classifier because since the prior probabilities of class 1 is higher than class 0 so the decision boundary moves away from class 1 and classifies more data as class 1. That is why we have a higher misclassification error for class 0 than class 1.

2.5 Question 4

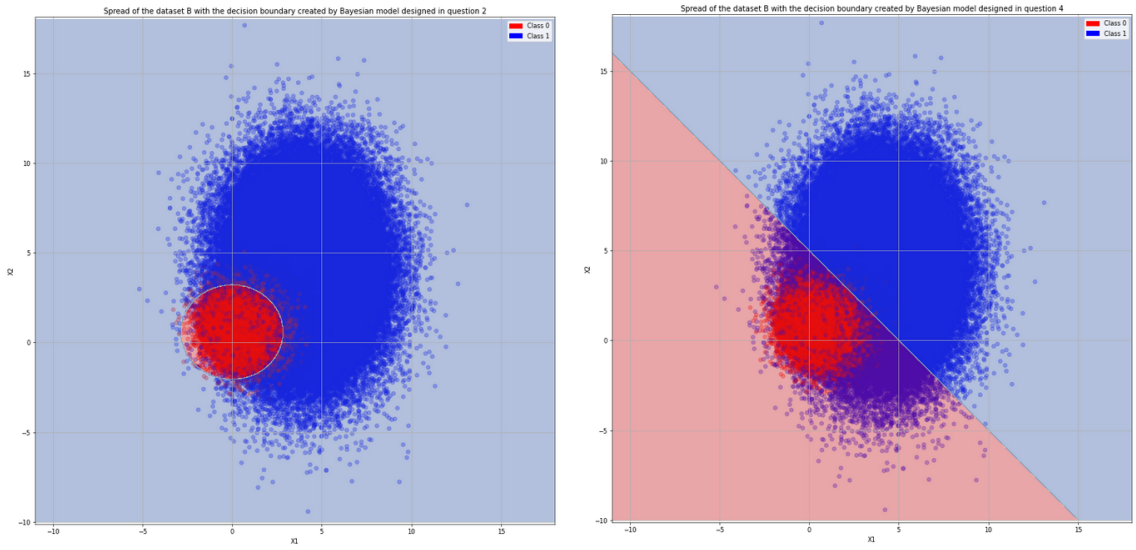
For this section we need to perform the same experiments that we did in question 3 for dataset B and compare with the results from question 2 and 3.

2.5.1 Experimental Results

The performance of Euclidean distance classifier and Bayes Classifier for dataset A and B can be found in Table 2.

Table 2: Comparison of different classifiers on dataset A and B in a scale of 100.

Misclassification Rates	Class 0	Class 1	Total	Error Bound
Euclidean distance classifier - A	1.6866	1.7635	1.7405	4.8299
Euclidean distance classifier - B	1.7816	19.3621	14.0880	20.3539
Bayes Classifier : Case III	8.3783	7.3985	7.6925	



(a) (b)

Figure 5: For the data set B, (a) shows the decision boundary generated by Bayes Classifier and (b) shows decision boundary created by the Euclidean Distance Classifier.

From Table 2 we can see that compared to the Bayes Classifier for Case III, Euclidean distance classifier performed much worse. Euclidean distance classifier achieved 14.0880%

of misclassification error while the Bayes Classifier obtained 7.6925% misclassification error. Visualization of the decision boundaries created by Euclidean distance classifier and Bayes Classifier for Case III is shown in Figure 5.

2.5.2 Comparison and Discussion

One of the reason for the Bayes Classifier to outperform Euclidean distance classifier is that, in dataset B the data distribution for the two classes overlap to some extend in two dimensional plane. Since Euclidean distance classifier draws a straight line for the data and there are more samples of data in Class 1 it is natural that the misclassification rates for Class 1 will be higher than Class 0. However, the Bayes classifier for Case III creates a decision boundary which is non-linear ensures better performance from the part of Bayes Classifier. Another thing that we can see that the performance of both our model is better than the error bound which ensures that our implementation is correct. Compared to the performance of the Euclidean distance classifier for dataset A to dataset B we can see the model performs poorly in dataset B. This is occurring primarily because the data distribution of the datasets are different. Compared to dataset A, dataset B is has different shape for his clusters and there is a significant portion of overlap to the other class.

References

[1] https://numpy.org/doc/stable/reference/random/generated/numpy.random.multivariate_normal.html