Bayes Classifiers

Assignment 1

CS 479

Ethan Brown, Ethan Park

Due: February 25, 2020

Handed in: February 25, 2020

Division of work: Ethan Park made the files and the structure for the code to generate the random samples for each distribution. He also worked out the Battacharrya error bound for each part. For the report, Ethan Park worked on the theory, implementation, and source code sections. Ethan Brown worked out and implemented the Bayes Classifier for each part and plotted the generated data along with the decision boundary. For the report, Ethan Brown worked on the results and discussion section.

# Table of Contents

## Theory................................................................................................ 3

### Problem Part 1A........................................................................... 3

### Problem Part 1B........................................................................... 4

### Problem Part 2A........................................................................... 4

### Problem Part 2B........................................................................... 5

### Problem 3.................................................................................... 5

## Implementation.................................................................................. 6

## Results and discussion....................................................................... 6

### Problem 1.................................................................................... 6

### Problem 2.................................................................................... 9

### Problem 3.................................................................................... 12

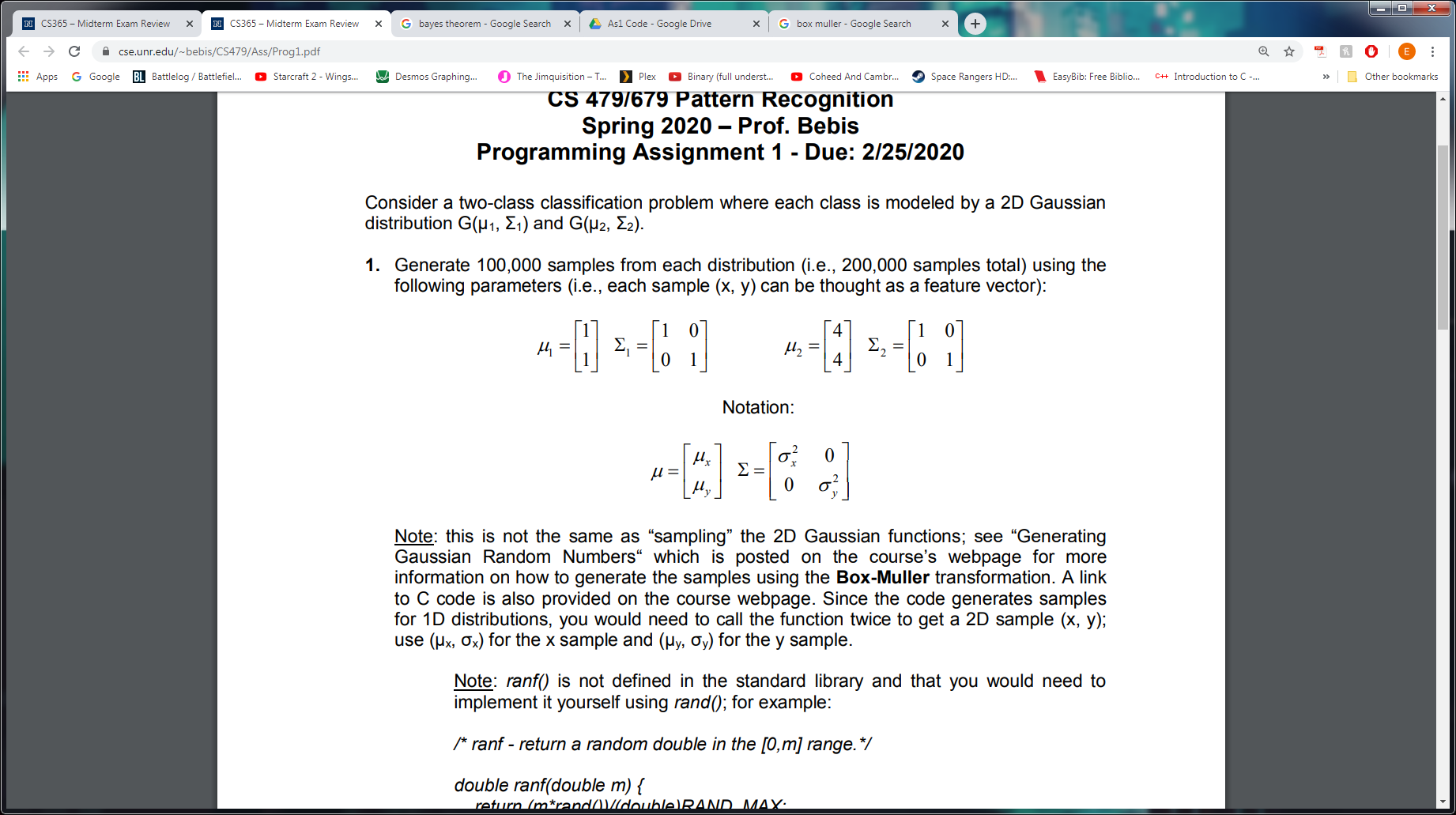
## Source Code.......................................................................................14

## 

**Theory**

Problem Part 1A

Section i requires designing a Bayes classifier for minimum error in which each class is modeled by a 2D Gaussian distribution and where



and the probability of each state of nature occurring is the same i.e. . There are three different discriminants for three different cases. For this problem the discriminant for the first case

is chosen since features are statistically independent with each feature having the same variance. The discriminants for both distributions are calculated for each sample and if the discriminant for the first discriminant is greater than the second, the sample will be classified under the first class, otherwise it is classified under the second.

Section ii requires plotting the Bayes decision boundary with the generated samples. To find the Bayes decision boundary set the discriminants for both distributions equal to each other and solving.

Section iii involves finding the number of misclassified samples for each class as well as the total misclassifications. Finding misclassifications for each class can be achieved by having the program keep track of correct classifications and counting how many times the classifier makes a decision that contradicts these correct classifications. The number of misclassifications for each class can then be added together for the total.

Section iv was not completed as this section is a graduate student requirement.

Section v involves calculating the Bhattacharyya bound. The Bhattacharyya bound is an error bound given by the equation

,

where the error boundary, , is 0.5.

Problem Part 1B

Section i is similar to section i in 1A except the probability of each state of nature occurring is not the same with and . Since the features do not change from 1A, the same discriminant case can be used. The only notable difference is that since the prior probabilities are no longer equivalent, is not zero. The discriminants for both distributions are again calculated and compared to determine classification.

Section ii requires the calculation of the Bayes decision boundary. This calculation is the same as 1A and is found by setting the discriminants for both distributions equal to each other and solving.

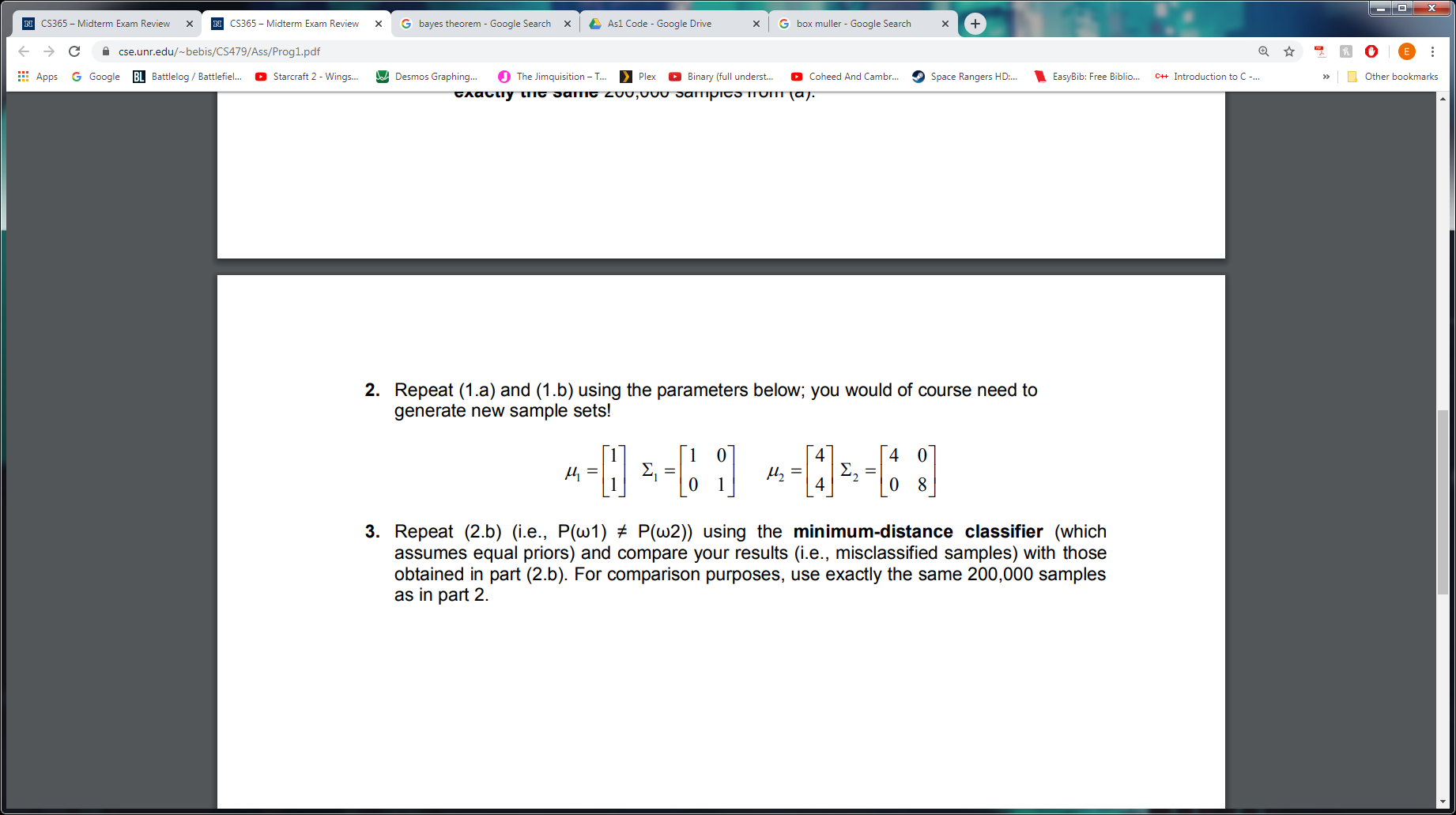
Section iii involves finding the number of misclassified samples and is completed in the same way as section iii of A1.

Section iv was not completed as this section is a graduate student requirement.

Section v involves calculating the Bhattacharyya bound which is done in the same way as section v of A1.

Problem Part 2A

Section i requires designing a Bayes classifier for minimum error in which each class is modeled by a 2D Gaussian distribution and where



and the probability of each state of nature occurring is the same i.e. . For this problem case 3 is used as variances are arbitrary. The discriminants for both distributions are again calculated for each sample, but are now calculated with the case 3 discriminant. If the first discriminant is greater than the second the sample is classified under the first class and it is classified under the second class otherwise.

Section ii requires plotting the Bayes decision boundary with the generated samples which is calculated in the same way as section ii of A1.

Section iii, like 1A and 1B, involves finding the number of misclassified samples and is completed in the same way as section iii of A1.

Section iv was not completed as this section is a graduate student requirement.

Section v again involves calculating the Bhattacharyya bound which is done in the same way as section v of A1.

Problem Part 2B

Section i is similar to section i in 2A except the probability of each state of nature occurring is not the same with and . The same discriminant case (case 3) is used as variances are still arbitrary. Like in section i of 2A, the discriminants for both distributions are calculated and compared to determine classification.

Section ii, like previous sections, requires plotting the Bayes decision boundary with the generated samples which is calculated in the same way as section ii of A1.

Section iii, again involves finding the number of misclassified samples and is completed in the same way as section iii of A1.

Section iv was not completed as this section is a graduate student requirement.

Section v involves calculating the Bhattacharyya bound which is again done in the same way as section v of A1.

Problem Part 3

Section i is similar to section i of 2B with the same distributions, prior probabilities, and generated samples. However, instead of using the optimal discriminant, it is required that the minimum-distance classifier be used. The discriminants are used in the same way as previous sections with the class with the larger discriminant being chosen.

Section ii, requires plotting the Bayes decision boundary with the generated samples and is calculated in the same way as section ii of A1.

Section iii, involves finding the number of misclassified samples and is completed in the same way as section iii of A1.

Section iv was not completed as this section is a graduate student requirement.

Section v involves calculating the Bhattacharyya bound which is again done in the same way as section v of A1.

**Implementation**

For parts that require random samples generated, the Box-Muller Transformation functions provided were used to generate the random samples and were stored.The x coordinates of the first distribution were stored in the float array distribution1x, the y coordinates of the first distribution were stored in distribution1y, the x coordinates of the second distribution were stored in distribution2x, and the y coordinates of the second distribution were stored in distribution2y. The distributions were then written to files so that the data could be plotted in Microsoft Excel. The decisions that each classifier made were stored in boolean arrays and were also logged on output files.

For the Bayesian classifier the first and third discriminant case were implemented as functions as these were the two cases needed for the problems. In both cases, the use of arrays instead of vectors made calculations involving matrices difficult. Because of these difficulties, the equations needed were simplified by hand such that the cases could be calculated without using these calculations. Parts A and B for problem 1 both require case 1 which is implemented using the functions discriminant\_case\_1a and discriminant\_case\_1b respectively. The second problem required case 3 which is implemented using the function discriminant\_case\_3. Finally, the third problem required a minimum distance discriminant which is implemented using the function discriminant\_min\_distance.

A function to find the Bhattacharyya bound was also implemented and is named calculateBhattacharyya. Because of the calculation difficulties when implementing the discriminant cases, the team decided to use vectors to calculate the Bhattacharyya bound. The team decided to use the Eigen library, using the data structures Vector2f and Matrix2f to store the matrices.

**Results and Discussion**

1.a.i

Since the features were statistically independent and each feature had the same variance, we used the discriminant for case 1.

1.a.ii

Having found the decision boundary line and generated random samples from each distribution, we graphed them together. The equation for the boundary line is obtained by setting gi(x) and gj(x) equal to each other then solving. The equation derived from this is y = 5 - x. The graph is shown in figure 1.

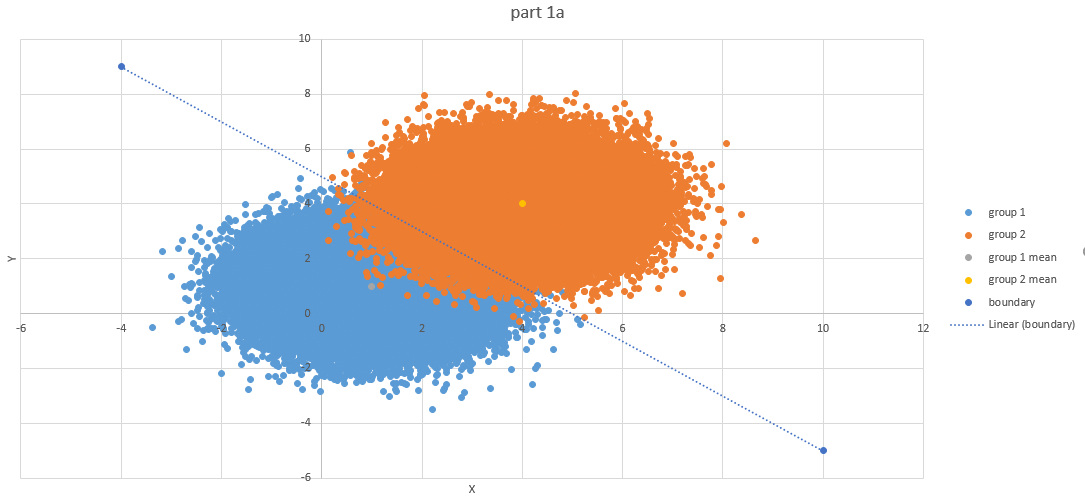


Figure 1 shows the distributions from group 1 and group 2 from part 1 a. Mean for group 1 is (1, 1) and the mean for group 2 is (4, 4). The decision boundary line is given by the equation y = 5-x.

1.a.iii

Our Bayesian classifier misidentified 2866 samples from class 1 and 1299 samples from class 3 for a total of 4165 misclassifications. In figure 1, the misclassifications from group 1 are represented by the blue dots that lie above the boundary line, and the misclassifications from group 2 are represented by the orange dots that lie below the boundary line.

1.a.v

The Battacharyya bound is calculated by setting beta to 0.5. After simplifying the expression and performing the calculations we determined that k(0.5) = 2.250 and P(error) = .1054.

1.b.i

Since the prior probabilities of each class are no longer equal, we included ln(P(omega1)/P(omega2)) in our calculations. The samples remained the same, so the only difference was how they were classified.

1.b.ii

With the new prior probabilities, we calculated the decision boundary line to be the equation y = 2.73104 - x. The boundary line has the same slope as the one from part 1.a.ii, it is just shifted downward. This is shown in figure 2.

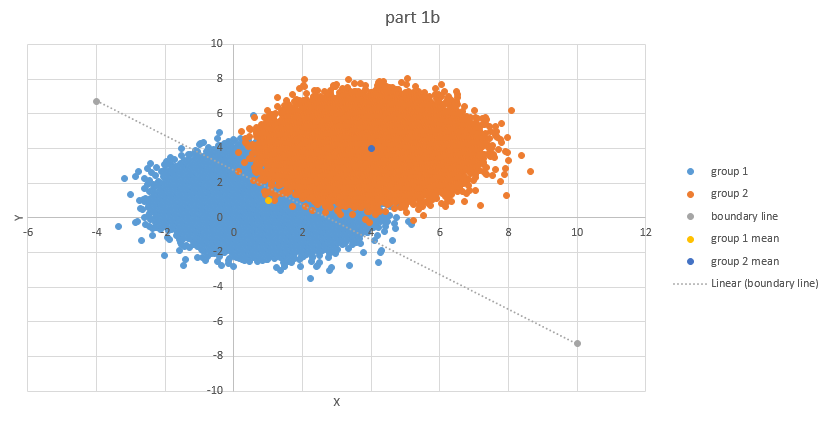


Figure 2 shows the distribution from group 1 and 2 along with the decision boundary line. The same samples were used from part 1.a so the distributions are identical to those from figure 1. The decision boundary line however, is affected by the prior probabilities, so it shifts downward. From the graph it is easy to see that more samples from group 1 were classified as class 2 than the opposite.

1.b.iii

The number of misclassified samples for each group is different than it was in part 1.a. In this case, there were 3632 misclassifications from samples from group 1 and 720 misclassifications from samples from group 2. This led to a total of 4352 misclassifications. It is also noteworthy to point out that there were significantly more misclassifications from group 1 than group 2. This is because the prior probabilities influence the classifications for each sample. This can be seen in figure 2. There are much more blue dots above the boundary line than orange dots below it.

1.b.v

The Battacharrya bound is the same as part 1.a since the only change was the prior probabilities of both classes and the prior probabilities do not affect the Battacharrya bound. Once again k(0.5) = 2.250 and P(error) = .1054

Figure 3 shows what the program outputs for the first question. More specifically, it shows the number of misclassified samples for each class, the total number of misclassified samples, and the Bhattacharyya bound for each question part.

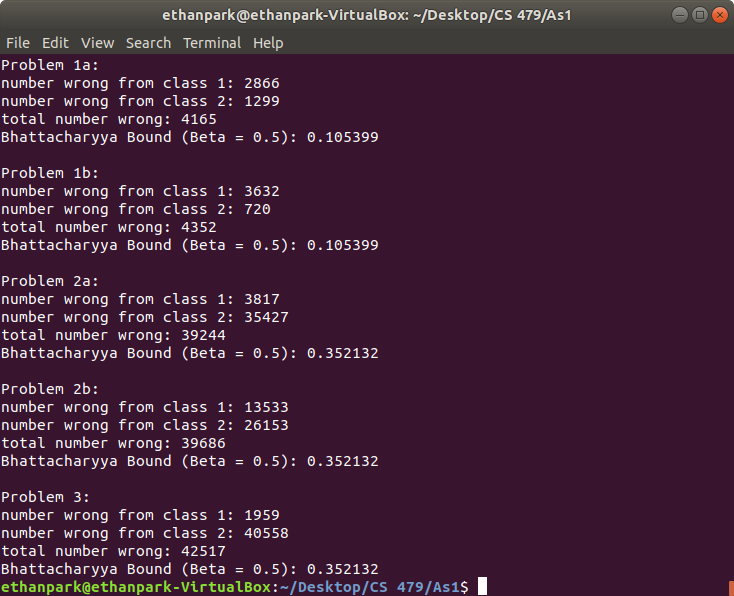


Figure 3 shows the terminal output from the program for question 1.

2.a.i

Since the variance matrices for each feature were arbitrary, we used the discriminant for case 3.

2.a.ii

Since the variance for distribution 2 was much larger than that of distribution 1, distribution 2 covered a much larger area than distribution 1. As a result, distribution 1 is completely inside the area of distribution 2. This can be seen in figure 4 with the decision boundary shown in figure 5.

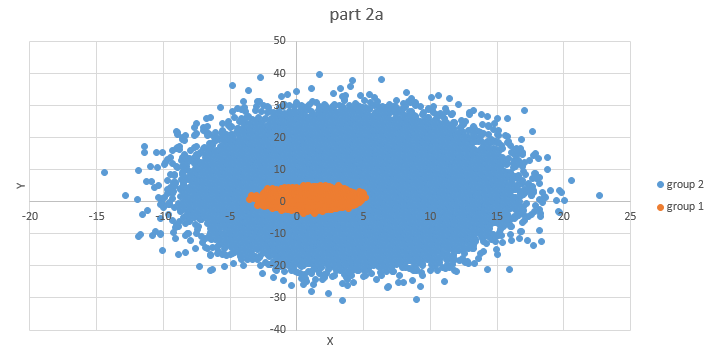


Figure 4 shows the distributions obtained from part 2a. Group 1 has a much smaller variance so the samples are packed more tightly together. Group 2 has a much larger variance so the samples cover a larger area.

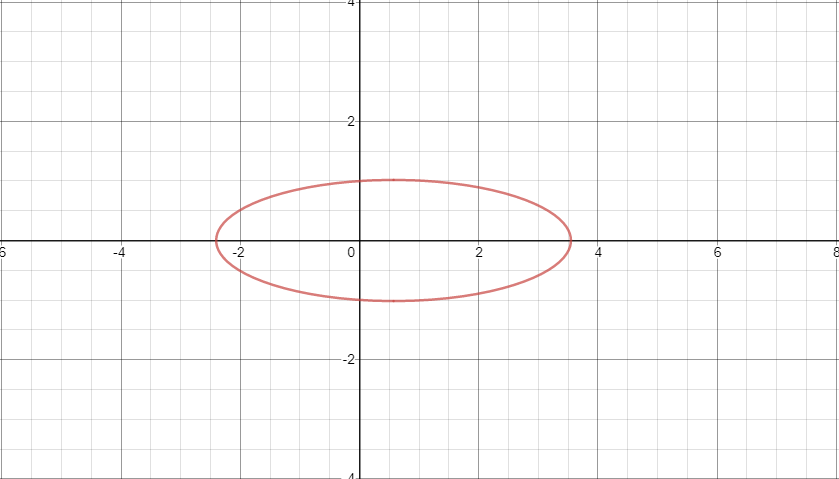


Figure 5 shows the boundary line with the equation -3.75y^{2}-.4375x^{2}+.5x+3.7329=0

2.a.iii

Our Bayesian Classifier for part 2.a misidentified 3817 samples from class 1 and 35427 samples from class 2 for a total of 39244 misclassifications. There were far more misclassifications from group 2 represented by blue dots inside of the boundary ellipse in figure 4 and figure 5 than misclassifications from group 1 represented by orange dots outside the boundary ellipse.

2.a.v

The Battacharyya bound is calculated by setting beta to 0.5. After simplifying the expression and performing the calculations we determined that P(error) = .3521.

2.b.i

Similarly to part 1.b.i, the prior probabilities are no longer equal so we included ln(P(omega1)/P(omega2)) in our calculations. Once again the samples were the same so the only differences came in the form of different classifications.

2.b.ii

Since the samples were the same from part 2.a, the plot of the samples is identical. The only difference once again is the decision boundary line which is a smaller ellipse. As a result,there were more samples misclassified from group 1, and fewer samples misclassified from group 2. This can be seen in figure 6 with the decision boundary shown in figure 7.

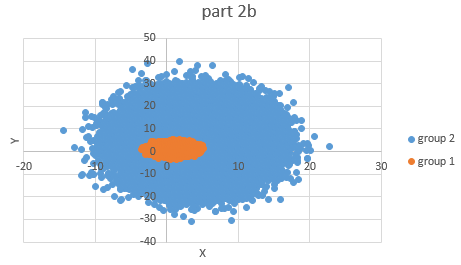


Figure 6 shows the distributions obtained from part 2b which is identical to that of part 2a since the same samples were used. The different variances are once again shown by different areas covered by samples.

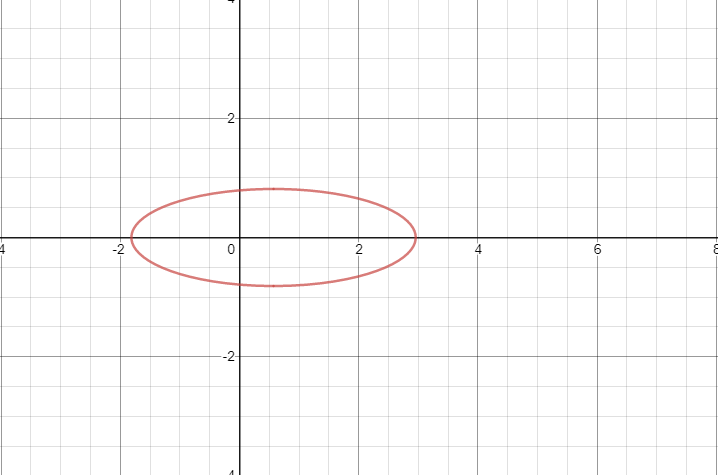


Figure 7 shows the boundary line with the equation -3.75y^{2}-.4375x^{2}+.5x+2.3466=0

2.b.iii

Similarly to part 1, the change in prior probability led to a change in the number of misclassified samples. In this case, the number of misclassified samples was 13533 from group 1, 26153 from group 2, and 39686 in total. The total number of misclassified samples was not much different from part 2.a.iii, but there was a significant difference between the number of misclassifications for each group. This can be seen in figure 7 by the smaller ellipse. This means that more orange dots fall outside the ellipse and fewer blue dots fall inside the ellipse.

2.b.v

The Battacharrya bound is the same as part 2.a since the only change was the prior probabilities of both classes and the prior probabilities do not affect the Battacharrya bound. Once again P(error) = .3521

Figure 8 shows what the program outputs for the second question. It shows the number of misclassified samples for each class, the total number of misclassified samples, and the Bhattacharyya bound for each question part.

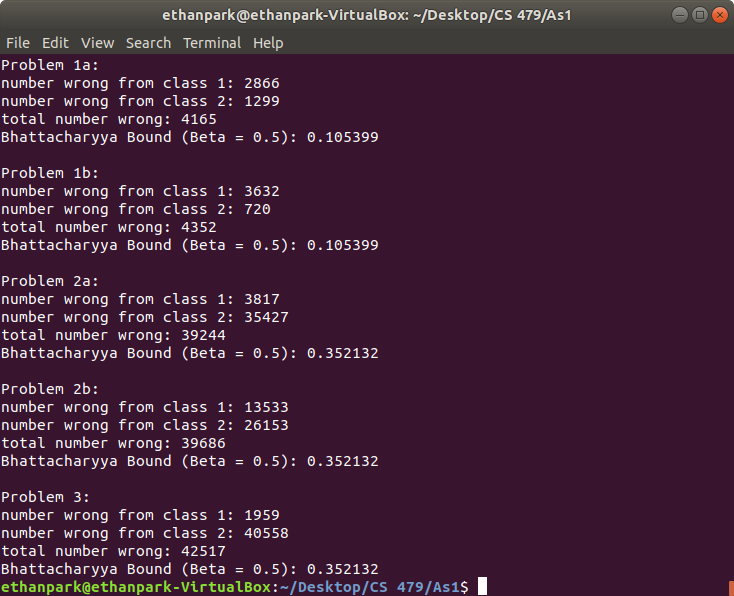


Figure 8 shows the terminal output from the program for question 2.

3.i

For this part, we used the minimum distance classifier from case 1 to determine the discriminants.

3.ii

The samples used were the same samples as part 2 so the graphs are identical. The decision boundary is the same as part 1, however, since we used the minimum distance classifier from case one that assumes equal priors rather than the equation from case 3 that does not make that assumption. The decision boundary does not do a very good job of separating the classes which leads to a multitude of misclassifications. This graph can be seen in Figure 9.

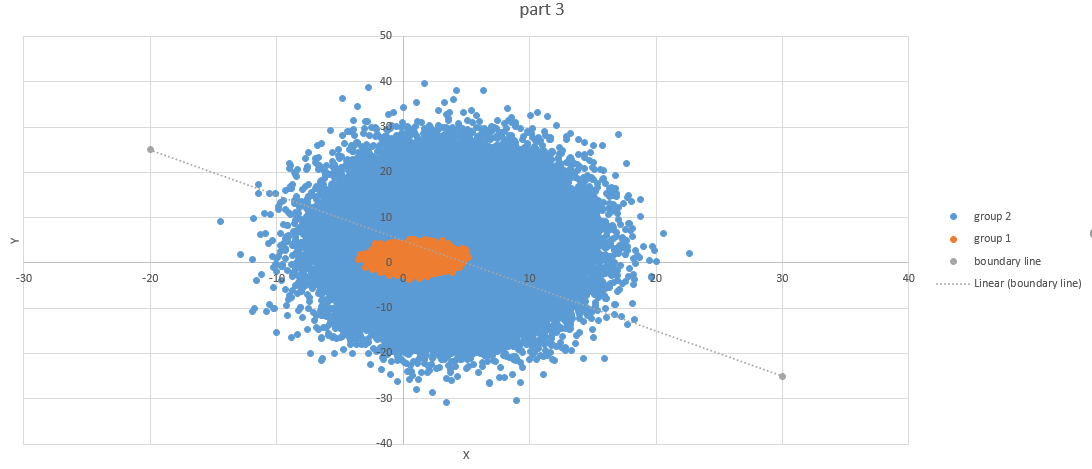


Figure 9 shows the same distributions from parts 2.a and 2.b shown in Figure 4 and Figure 6. The decision boundary is the same as the decision boundary from part 1 since the same classifier was used. It is clear that the decision boundary line does not separate the 2 classes very well leading to classification errors.

3.iii

The minimum distance classifier misclassified 1959 samples from class 1 and 40558 samples from class 2 for a total of 42517 misclassifications. The minimum distance classifier with the distributions from part 3 gave us the largest number of total misclassifications. This is because the minimum distance classifier assumes that the priors are equal which is not true.

3.v

The Battacharrya bound is the same as part 2.a since the only changes were the prior probabilities of both classes and the determinant case, neither of which affect the Battacharrya bound. Once again P(error) = .3521

Figure 10 shows what the program outputs for the third question. It shows the number of misclassified samples for each class, the total number of misclassified samples, and the Bhattacharyya bound..

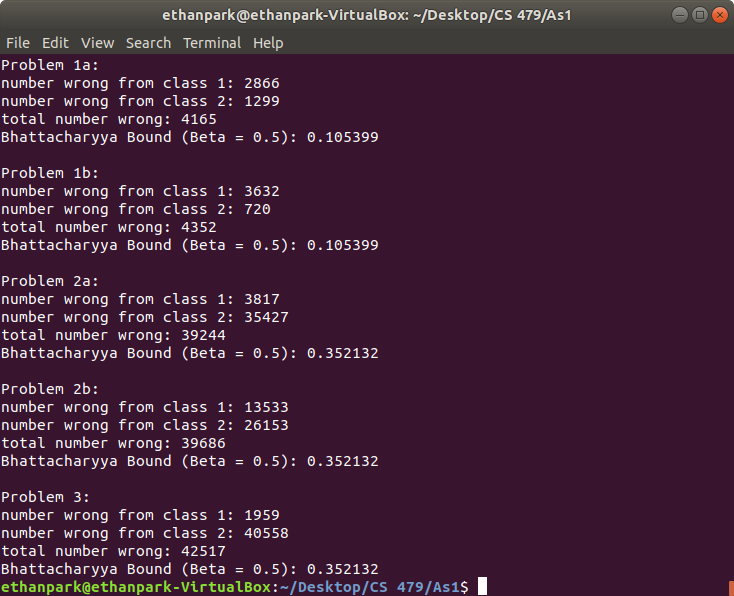


Fig: 10 shows the terminal output from the program for question 3.

**Source Code**

main.cpp

#include <iostream>

#include "boxmuller.cpp"

#include <fstream>

#include <cmath>

#include <math.h>

#include <vector>

#include <Eigen/Dense>

using namespace Eigen;

using namespace std;

/\*\*\*Discriminant cases were simplified in a way that eliminates matrix calculations\*\*\*/

/\*\*Case 1 implemented for Problem 1a\*\*/

float discriminant\_case\_1a(float data\_x, float data\_y, float mean\_x, float mean\_y)

{

return -((sqrt((data\_x\*data\_x)+(data\_y\*data\_y)) - sqrt((mean\_x\*mean\_x)+(mean\_y\*mean\_y)))\*(sqrt((data\_x\*data\_x)+(data\_y\*data\_y)) - sqrt((mean\_x\*mean\_x)+(mean\_y\*mean\_y))));

}

/\*\*Case 1 implemented for Problem 1b\*\*/

//sigma is sigma squared. prob is lnP(omegai)

float discriminant\_case\_1b(float x1, float x2, float mu1, float mu2, float sigma, float prob)

{

return (((mu1\*x1)/sigma)+((mu2\*x2)/sigma)-((mu1\*mu1)/(2\*sigma))-((mu2\*mu2)/(2\*sigma))+prob);

}

/\*\*Case 3 implemented for Problem 2\*\*/

//sigma is sigma squared. prob is lnP(omegai) - 1/2ln(sigma1\*sigma2)

float discriminant\_case\_3(float x1, float x2, float mu1, float mu2, float sigma1, float sigma2, float prob)

{

return (-((x1\*x1)/(2\*sigma1)) - ((x2\*x2)/(2\*sigma2)) + ((x1\*mu1)/sigma1) + ((x2\*mu2)/sigma2) - ((mu1\*mu1)/(2\*sigma1)) - ((mu2\*mu2)/(2\*sigma2)) + prob);

}

/\*\*Minimum Distance Case implemented for Problem 3\*\*/

float discriminant\_min\_distance(float data\_x, float data\_y, float mean\_x, float mean\_y, float prob)

{

return -((sqrt((data\_x\*data\_x)+(data\_y\*data\_y)) - sqrt((mean\_x\*mean\_x)+(mean\_y\*mean\_y)))\*(sqrt((data\_x\*data\_x)+(data\_y\*data\_y)) - sqrt((mean\_x\*mean\_x)+(mean\_y\*mean\_y))) + prob);

}

/\*\*Function that calculates the Bhattacharyya bound\*\*/

float calculateBhattacharyya(float beta, Vector2f mu\_1, Vector2f mu\_2, Matrix2f sigma\_1, Matrix2f sigma\_2)

{

float k\_beta = (beta\*(1-beta))/2.0;

k\_beta \*= (mu\_1 - mu\_2).transpose() \* ((1-beta)\*sigma\_1 + (beta)\*sigma\_2).inverse() \* (mu\_1-mu\_2);

k\_beta += 0.5 \* log(((1-beta)\*sigma\_1 + (beta)\*sigma\_2).determinant() / (pow(sigma\_1.determinant(), 1-beta)

\* pow(sigma\_2.determinant(), beta)));

float bhattacharyya\_error = exp(-1.0\*k\_beta);

return bhattacharyya\_error;

}

int main()

{

// arrays for storing first distribution

float distribution1x[100000];

float distribution1y[100000];

// arrays for storing second distribution

float distribution2x[100000];

float distribution2y[100000];

/\*Problem 1\*/

// generate the samples

for(int i=0;i<100000;i++)

{

distribution1x[i] = box\_muller(1, 1);

distribution1y[i] = box\_muller(1, 1);

distribution2x[i] = box\_muller(4, 1);

distribution2y[i] = box\_muller(4, 1);

}

// log the samples on output files

ofstream outputFile;

outputFile.open("p1\_data1x.txt");

for(int i=0;i<100000;i++)

{

outputFile << distribution1x[i] << endl;

}

outputFile.close();

outputFile.open("p1\_data2x.txt");

for(int i=0;i<100000;i++)

{

outputFile << distribution2x[i] << endl;

}

outputFile.close();

outputFile.open("p1\_data1y.txt");

for(int i=0;i<100000;i++)

{

outputFile << distribution1y[i] << endl;

}

outputFile.close();

outputFile.open("p1\_data2y.txt");

for(int i=0;i<100000;i++)

{

outputFile << distribution2y[i] << endl;

}

outputFile.close();

cout << "Problem 1a: " << endl;

// array to hold decision made for the classifer

bool p1\_a\_decision[200000];

// classify the samples using case 1

for (int i = 0; i < 200000; i++)

{

if (i < 100000)

{

float gi = discriminant\_case\_1a(distribution1x[i], distribution1y[i], 1, 1);

float gj = discriminant\_case\_1a(distribution1x[i], distribution1y[i], 4, 4);

if (gi - gj > 0)

{

p1\_a\_decision[i] = 1;

}

else

{

p1\_a\_decision[i] = 0;

}

}

else

{

float gi = discriminant\_case\_1a(distribution2x[i-100000], distribution2y[i-100000], 1, 1);

float gj = discriminant\_case\_1a(distribution2x[i-100000], distribution2y[i-100000], 4, 4);

if (gi - gj > 0)

{

p1\_a\_decision[i] = 1;

}

else

{

p1\_a\_decision[i] = 0;

}

}

}

// calculate the number of misclassifications made and log decisions to an output file

outputFile.open("p1\_a\_decisions.txt");

int wrong1 = 0;

int wrong2 = 0;

for (int i = 0; i < 200000; i++)

{

outputFile << p1\_a\_decision[i] << endl;

if (i < 100000)

{

if (p1\_a\_decision[i] == 0)

{

wrong1++;

}

}

else

{

if (p1\_a\_decision[i] == 1)

{

wrong2++;

}

}

}

outputFile.close();

cout << "number wrong from class 1: " << wrong1 << endl;

cout << "number wrong from class 2: " << wrong2 << endl;

cout << "total number wrong: " << wrong1 + wrong2 << endl;

// calculate the Bhattacharyya bound

float beta = 0.5;

Vector2f mu\_1;

mu\_1 << 1, 1;

Matrix2f sigma\_1;

sigma\_1 << 1, 0,

0, 1;

Vector2f mu\_2;

mu\_2 << 4, 4;

Matrix2f sigma\_2;

sigma\_2 << 1, 0,

0, 1;

float bhattacharyya = calculateBhattacharyya(beta, mu\_1, mu\_2, sigma\_1, sigma\_2);

cout << "Bhattacharyya Bound (Beta = " << beta << "): " << bhattacharyya << endl;

cout << endl << "Problem 1b: " << endl;

// array to hold decisions made by the classifer

bool p1\_b\_decision[200000];

// classify the samples using case 1

for (int i = 0; i < 200000; i++)

{

if (i < 100000)

{

float gi = discriminant\_case\_1b(distribution1x[i], distribution1y[i], 1, 1, 1, -1.60943);

float gj = discriminant\_case\_1b(distribution1x[i], distribution1y[i], 4, 4, 1, -.22314);

if (gi - gj > 0)

{

p1\_b\_decision[i] = 1;

}

else

{

p1\_b\_decision[i] = 0;

}

}

else

{

float gi = discriminant\_case\_1b(distribution2x[i-100000], distribution2y[i-100000], 1, 1, 1, -1.60943);

float gj = discriminant\_case\_1b(distribution2x[i-100000], distribution2y[i-100000], 4, 4, 1, -.22314);

if (gi - gj > 0)

{

p1\_b\_decision[i] = 1;

}

else

{

p1\_b\_decision[i] = 0;

}

}

}

// calculate the number of misclassifications made and log decisions to an output file

outputFile.open("p1\_b\_decisions.txt");

wrong1 = 0;

wrong2 = 0;

for (int i = 0; i < 200000; i++)

{

outputFile << p1\_b\_decision[i] << endl;

if (i < 100000)

{

if (p1\_b\_decision[i] == 0)

{

wrong1++;

}

}

else

{

if (p1\_b\_decision[i] == 1)

{

wrong2++;

}

}

}

outputFile.close();

cout << "number wrong from class 1: " << wrong1 << endl;

cout << "number wrong from class 2: " << wrong2 << endl;

cout << "total number wrong: " << wrong1 + wrong2 << endl;

// calculate the Bhattacharyya bound

bhattacharyya = calculateBhattacharyya(beta, mu\_1, mu\_2, sigma\_1, sigma\_2);

cout << "Bhattacharyya Bound (Beta = " << beta << "): " << bhattacharyya << endl;

/\*Problem 2\*/

// generate the samples

for(int i=0;i<100000;i++)

{

distribution1x[i] = box\_muller(1, 1);

distribution1y[i] = box\_muller(1, 1);

distribution2x[i] = box\_muller(4, 4);

distribution2y[i] = box\_muller(4, 8);

}

// log the samples on output files

outputFile.open("p2\_data1x.txt");

for(int i=0;i<100000;i++)

{

outputFile << distribution1x[i] << endl;

}

outputFile.close();

outputFile.open("p2\_data2x.txt");

for(int i=0;i<100000;i++)

{

outputFile << distribution2x[i] << endl;

}

outputFile.close();

outputFile.open("p2\_data1y.txt");

for(int i=0;i<100000;i++)

{

outputFile << distribution1y[i] << endl;

}

outputFile.close();

outputFile.open("p2\_data2y.txt");

for(int i=0;i<100000;i++)

{

outputFile << distribution2y[i] << endl;

}

outputFile.close();

cout << endl << "Problem 2a: " << endl;

// array to hold decisions made by the classifer

bool p2\_a\_decision[200000];

// classify the samples using case 3

for (int i = 0; i < 200000; i++)

{

if (i < 100000)

{

float gi = discriminant\_case\_3(distribution1x[i], distribution1y[i], 1, 1, 1, 1, -.69315);

float gj = discriminant\_case\_3(distribution1x[i], distribution1y[i], 4, 4, 4, 8, -2.42602);

if (gi - gj > 0)

{

p2\_a\_decision[i] = 1;

}

else

{

p2\_a\_decision[i] = 0;

}

}

else

{

float gi = discriminant\_case\_3(distribution2x[100000-i], distribution2y[100000-i], 1, 1, 1, 1, -.69315);

float gj = discriminant\_case\_3(distribution2x[100000-i], distribution2y[100000-i], 4, 4, 4, 8, -2.42602);

if (gi - gj > 0)

{

p2\_a\_decision[i] = 1;

}

else

{

p2\_a\_decision[i] = 0;

}

}

}

// calculate the number of misclassifications made and log decisions to an output file

outputFile.open("p2\_a\_decisions.txt");

wrong1 = 0;

wrong2 = 0;

for (int i = 0; i < 200000; i++)

{

outputFile << p2\_a\_decision[i] << endl;

if (i < 100000)

{

if (p2\_a\_decision[i] == 0)

{

wrong1++;

}

}

else

{

if (p2\_a\_decision[i] == 1)

{

wrong2++;

}

}

}

outputFile.close();

cout << "number wrong from class 1: " << wrong1 << endl;

cout << "number wrong from class 2: " << wrong2 << endl;

cout << "total number wrong: " << wrong1 + wrong2 << endl;

// calculate the Bhattacharyya bound

beta = 0.5;

mu\_1 << 1, 1;

sigma\_1 << 1, 0,

0, 1;

mu\_2 << 4, 4;

sigma\_2 << 4, 0,

0, 8;

bhattacharyya = calculateBhattacharyya(beta, mu\_1, mu\_2, sigma\_1, sigma\_2);

cout << "Bhattacharyya Bound (Beta = " << beta << "): " << bhattacharyya << endl;

cout << endl << "Problem 2b: " << endl;

// array to hold decisions made by the classifer

bool p2\_b\_decision[200000];

// classify the samples using case 3

for (int i = 0; i < 200000; i++)

{

if (i < 100000)

{

float gi = discriminant\_case\_3(distribution1x[i], distribution1y[i], 1, 1, 1, 1, -1.60944);

float gj = discriminant\_case\_3(distribution1x[i], distribution1y[i], 4, 4, 4, 8, -1.95601);

if (gi - gj > 0)

{

p2\_b\_decision[i] = 1;

}

else

{

p2\_b\_decision[i] = 0;

}

}

else

{

float gi = discriminant\_case\_3(distribution2x[100000-i], distribution2y[100000-i], 1, 1, 1, 1, -1.60944);

float gj = discriminant\_case\_3(distribution2x[100000-i], distribution2y[100000-i], 4, 4, 4, 8, -1.95601);

if (gi - gj > 0)

{

p2\_b\_decision[i] = 1;

}

else

{

p2\_b\_decision[i] = 0;

}

}

}

// calculate the number of misclassifications made and log decisions to an output file

outputFile.open("p2\_b\_decisions.txt");

wrong1 = 0;

wrong2 = 0;

for (int i = 0; i < 200000; i++)

{

outputFile << p2\_b\_decision[i] << endl;

if (i < 100000)

{

if (p2\_b\_decision[i] == 0)

{

wrong1++;

}

}

else

{

if (p2\_b\_decision[i] == 1)

{

wrong2++;

}

}

}

outputFile.close();

cout << "number wrong from class 1: " << wrong1 << endl;

cout << "number wrong from class 2: " << wrong2 << endl;

cout << "total number wrong: " << wrong1 + wrong2 << endl;

// calculate the Bhattacharyya bound

bhattacharyya = calculateBhattacharyya(beta, mu\_1, mu\_2, sigma\_1, sigma\_2);

cout << "Bhattacharyya Bound (Beta = " << beta << "): " << bhattacharyya << endl;

/\*Problem 3\*/

cout << endl << "Problem 3: " << endl;

// the samples generated for problem 2 are also used for problem 3

// array to hold decisions made by the classifer

bool p3\_decision[200000];

// classify the samples using minimum distance case

for (int i = 0; i < 200000; i++)

{

if (i < 100000)

{

float gi = discriminant\_min\_distance(distribution1x[i], distribution1y[i], 1, 1, -1.60943);

float gj = discriminant\_min\_distance(distribution1x[i], distribution1y[i], 4, 4, -.22314);

if (gi - gj > 0)

{

p3\_decision[i] = 1;

}

else

{

p3\_decision[i] = 0;

}

}

else

{

float gi = discriminant\_min\_distance(distribution2x[100000-i], distribution2y[100000-i], 1, 1, -1.60943);

float gj = discriminant\_min\_distance(distribution2x[100000-i], distribution2y[100000-i], 4, 4, -.22314);

if (gi - gj > 0)

{

p3\_decision[i] = 1;

}

else

{

p3\_decision[i] = 0;

}

}

}

// calculate the number of misclassifications made and log decisions to an output file

outputFile.open("p3\_decision.txt");

wrong1 = 0;

wrong2 = 0;

for (int i = 0; i < 200000; i++)

{

outputFile << p3\_decision[i] << endl;

if (i < 100000)

{

if (p3\_decision[i] == 0)

{

wrong1++;

}

}

else

{

if (p3\_decision[i] == 1)

{

wrong2++;

}

}

}

outputFile.close();

cout << "number wrong from class 1: " << wrong1 << endl;

cout << "number wrong from class 2: " << wrong2 << endl;

cout << "total number wrong: " << wrong1 + wrong2 << endl;

// calculate the Bhattacharyya bound

bhattacharyya = calculateBhattacharyya(beta, mu\_1, mu\_2, sigma\_1, sigma\_2);

cout << "Bhattacharyya Bound (Beta = " << beta << "): " << bhattacharyya << endl;

}

boxmuller.c

/\* boxmuller.c Implements the Polar form of the Box-Muller

Transformation

(c) Copyright 1994, Everett F. Carter Jr.

Permission is granted by the author to use

this software for any application provided this

copyright notice is preserved.

\*/

#include <math.h>

extern float ranf(); /\* ranf() is uniform in 0..1 \*/

float box\_muller(float m, float s) /\* normal random variate generator \*/

{ /\* mean m, standard deviation s \*/

float x1, x2, w, y1;

static float y2;

static int use\_last = 0;

if (use\_last) /\* use value from previous call \*/

{

y1 = y2;

use\_last = 0;

}

else

{

do {

x1 = 2.0 \* ranf() - 1.0;

x2 = 2.0 \* ranf() - 1.0;

w = x1 \* x1 + x2 \* x2;

} while ( w >= 1.0 );

w = sqrt( (-2.0 \* log( w ) ) / w );

y1 = x1 \* w;

y2 = x2 \* w;

use\_last = 1;

}

return( m + y1 \* s );

}

extern float ranf()

{

return (1\*rand())/(double)RAND\_MAX;

}