CS479 PROGRAMMING ASSIGNMENT 1

Ashlee Ladouceur and Adam Lychuk

Due Date: 2/26/2019
Date Turned In: 2/26/2019

Division of work:
Ashlee Ladouceur designed and developed the PA1.cpp file and created all graphs
Adam Lychuk designed and developed the Classifier.cpp and Classifier.h files
Ashlee Ladouceur wrote the Implementation Details and the Program Listings
Adam Lychuk wrote the Technical Discussion and Results and Discussion

Table of Contents

Technical Discussion

Bayesian decision theory is a simple and elegant way to approach statistical pattern recognition problems. The theory focuses on differentiating between classes of objects using a set of known probabilities. Decisions are made based on a set of prior probabilities represented by ω, where ω is the "state of nature". For example, given a set of objects ω_n, where at least two subsets exist $\omega \supset \omega_1$ and $\omega \supset \omega_2$, and $\omega_1 \not\subset \omega_2$, $\omega_2 \not\subset \omega_1$, there exists a certain likelihood that the object we pick is n_1 or n_2. These prior probabilities are then augmented by some value x dependent on ω, or $p(x|\omega)$, forming a class-conditional probability density function where x is defined as a specific feature observed of ω_1 or ω_2. Bayes formula is defined in Figure 1.

$$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)} \text{ where } p(x) = \sum_{j=1}^{c} p(x/\omega_j)P(\omega_j)$$

Figure 1: Bayes formula

Bayes formula describes the likelihood that the prior probabilities correctly classify the object, or the posterior probability. Given these likelihoods we can minimize our error by choosing the correct posterior probability, shown in Figure 2.

$$P(error|x) = \begin{cases} P(\omega_1|x) \ \ if\ we\ decide\ \omega_2 \\ P(\omega_2|x)\ if\ we\ decide\ \omega_1 \end{cases} \text{ or } P(error|x) = \min[\, P(\omega_1|x), P(\omega_2|x).$$

Figure 2: Probability of error formula.

The most likely class, or state of nature ω, can be calculated for every value of x, the observed feature, using this methodology. This is the foundation of classification using Bayesian theory, however the methodology described thus far is only applicable when using a singular feature and two categories. We also do not have the ability to reject irrational data outside of a certain boundary and have no expected risk of making an error.

Bayesian theory can be expanded to allow for more features and categories. Risk of making an incorrect decision can be defined, and a system for rejecting irrational data can be formulated. The risk of a particular action, or the probability that an incorrect decision is made given a feature x. Using this risk function, we can make our decisions based on the cost of a decision, and more importantly weight certain errors as costlier than others. This allows us reject data in which the cost of a decision will be too great. Formulaically, conditional risk is defined in Figure 3.

$$R(\alpha_i|\omega_j) = \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j)P(\omega_j|x)$$

Figure 3: Conditional risk formula for action α given x.

$\lambda(\alpha_i/\omega_j)$ describes the cost of taking action α_i, given state of nature ω_j, and combined with the posterior probability density function $P(\omega_j/x)$ gives the conditional risk probability density function $R(\alpha_i/x)$. The conditional risk formula can only calculate the risk for a finite state of

values and can be expanded through integration for calculation of every decision based on every possible observed feature x, shown in Figure 4.

$$R = \int R(\alpha(x)|x)p(x)dx$$

Figure 4: Overall risk formula.

The function for overall risk tells us that by calculating every possible conditional risk and making the decision that minimizes we can find the minimum overall risk traditionally denoted Bayes Risk $R^* = \min R$. These risk functions calculations can be costly. By making certain assumptions we can minimize the calculation required to make a decision.

Two category classification is a special case of Bayesian decision theory; however, it appears often enough in practice for its specific properties to be useful. When only two categories exist, the conditional risk becomes the equations shown in Figure 5.

$$R(\alpha_1|x) = \lambda_{11}P(\omega_1|x) + \lambda_{12}P(\omega_2|x)$$
$$R(\alpha_2|x) = \lambda_{21}P(\omega_1|x) + \lambda_{22}P(\omega_2|x)$$

Figure 5: Simplified two category classification conditional risk equations.

Using these equations, the decision rule can be expressed in a couple of unique ways. Fundamentally we can decide based on the values of the above risk equations. Deciding ω_1 if $R(\alpha_1|x)$ is less than $R(\alpha_2|x)$, and otherwise decide ω_2. This can be rewritten just in terms of the posterior probabilities shown in Figure 6.

$$Decide\ \omega_1\ if\ (\lambda_{21}-\lambda_{11})P(\omega_1|x) > (\lambda_{12}-\lambda_{22})P(\omega_2|x); otherwise\ decide\ \omega_2$$

Figure 6: Decision rule focused on posterior probabilities.

This decision rule focuses on the most likely state of nature scaled by the difference of loss between ω_1 and ω_2. This decision rule assumes that the difference between each respective loss is almost always positive as the penalty for an error is almost always greater than the penalty for a correct decision. This assumption allows the creation of the final unique two category classification decision rule shown in Figure 7.

$$Decide\ \omega_1\ if\ \frac{p(\omega_1|x)}{p(\omega_2|x)} > \frac{(\lambda_{12}-\lambda_{22})}{(\lambda_{21}-\lambda_{11})}\frac{P(\omega_1|x)}{P(\omega_2|x)}; otherwise\ decide\ \omega_2$$

Figure 7: Decision rule focused on the likelihood of a state of nature being over a certain threshold.

This version of the decision rule focuses on the prior probabilities given a feature x. Referred to as the "likelihood ratio" $\frac{p(\omega_1|x)}{p(\omega_2|x)}$ it denotes the relation between prior probabilities. The rule makes a decision based on the likelihood of being correct over a certain threshold, defined by the cost associated with a decision, denoted by loss ratio $\frac{(\lambda_{12}-\lambda_{22})}{(\lambda_{21}-\lambda_{11})}$ multiplied by the ratio of posterior probabilities $\frac{P(\omega_1|x)}{P(\omega_2|x)}$.

Simplifying associated risk of incorrect decisions would allow further simplification of the above decision rules, as well as general Bayesian decision rules. The Zero-One Loss function simplifies risk by assigning only two possible values, 0 and 1, to the cost function $\lambda(\alpha_i/\omega_j)$. $R(\alpha_i|\omega_j)$ becomes $1 - P(\omega_i|x)$, the average probability of error, because the conditional risk no longer assigns loss to a correct decision, only an incorrect one. Due to this simplification of the conditional risk the decision rule for classification is shown in Figure 8.

$$Decide\ \omega_1\ if\ P(\omega_1|x) > P(\omega_2|x); otherwise\ decide\ \omega_2$$

Figure 8: Simplified decision rule assuming a Zero-One loss function and two category classification.

Now the decision rule for classification is correlated only to the posterior probabilities reducing required calculation to decide by a considerable amount. By focusing on only two categories, using the Zero-One loss function, or both Bayesian classification can be very fast. These can be naturally made by using discriminant functions which allow for set decision regions and boundaries based on the Bayesian decision rules.

The Bayesian classifier can be represented as a set of discriminant functions $g_i(x)$ where i = 1,…,c. A feature vector is assigned to a state of nature $\omega_1$if $g_i(x) > g_j(x)$ where $j \neq i$. The risk or cost function is written $g_i(x) = -R(\alpha_i|x)$. If using the Zero-One loss function $g_i(x) = P(\omega_i|x)$. This expands by the definition of a posterior probability to Bayes formula, or $g_i(x) = \frac{p(x|\omega_i)P(\omega_i)}{p(x)}$ which simplifies to the formula shown in Figure 9.

$$g_i(x) = \ln p(x|\omega_i) + \ln P(\omega_i)$$

Figure 9: Bayes classifier discriminant function.

This is the general version of the Bayes classifier discriminant function. Decision regions are created by decision boundaries which are defined by setting two discriminates equal to each other ie. $g_1(x) = g_2(x)$. If we assume two categories, creating a “dichotomizer”, we only need one discriminant function for classification shown in Figure 10. This simplification is possible because of the formulation in Figure 7. It is common to have two categories so often only one discriminant function in necessary for classification; therefore, only one boundary and two regions.

$$g(x) = \frac{p(x|\omega_1)}{p(x|\omega_2)} + \ln P\frac{P(\omega_1)}{P(\omega_2)}$$

Figure 10: Function for Bayes known as the dichotomizer.

Often in nature, distributions of random variables follow a Gaussian Density. Bayes decision theory can be expanded using the discriminant of the Multivariate Gaussian Density, as a representation of the probability density function of the state of nature ω. The discriminant of a Multivariate Gaussian Density is defined as $N(\mu, \Sigma) = \frac{1}{2\pi^{d/2}|\Sigma|^{1/2}} \exp[-\frac{1}{2}(x-\mu)^t\Sigma^{-1}(x-\mu)]$ and the Bayes classifier using a Multivariate Gaussian Density is defined generally in Figure 11.

$$g_i(x) = -\frac{1}{2}(x-\mu)^t\Sigma^{-1}(x-\mu) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i)$$

Figure 11: Bayes classifier using a Multivariate Gaussian Density.

This is obviously a very calculation intensive discriminant, but it can be simplified by making certain assumptions in three different cases. The difference between these cases as well as the unique advantages and disadvantages, are discussed in the Results and Discussion section.

Implementation Details

Summary

The Bayes classifier, the minimum-distance classifier, the Chernoff bound, and the Bhattacharyya bound are implemented via functions in the Classifier.h file. The PA1.cpp file contains the main function that computes the classifiers and bounds after randomly generating samples from the 2D Gaussian distribution parameters using the Box-Muller C file provided that was converted to C++ for use in this assignment. In addition, the PA1.cpp file contains a function to export data to a csv for data plotting used in this report.

Classifier and Bound Methods

The Bayes classifier functionality is split into three different functions: case one, case two, and case three. Each of these cases specific equations are detailed in the technical discussion. For each case, the function calculates the discriminate of each sigma and mu combination using the formula for that case. If P(ω1) and P(ω2) are not equal, the log of the P value is added to that discriminate value, respectfully. Then, a 1 is returned to show that the discriminate for 1 is bigger than the discriminate for 2. A 2 is returned if the discriminate for 2 is bigger than the discriminate for 1.

```
int BAYESCASEX(Vector2f mu1, Vector2f mu2, Matrix2f sigma1, Matrix2f sigma2, float p1, float p2)
{
	float discriminateOne = FORMULA USING SIGMA1 AND MU1
	float discriminateTwo = FORMULA USING SIGMA2 AND MU2

	if(p1 != p2)
	{
		discriminateOne += log(p1)
		discriminateTwo += log(p2)
	}

	if(discriminateOne > discriminateTwo)
		return 1
	else
		return 2
}
```

The minimum distance classifier simply calculates the discriminate of each sigma and mu combination using the formula detailed in the technical discussion. Then, a 1 is returned to show that the discriminate for 1 is bigger than the discriminate for 2. A 2 is returned if the discriminate for 2 is bigger than the discriminate for 1.

```
int MINIMUMDISTANCE(Vector2f mu1, Vector2f mu2)
{
	float discriminateOne = FORMULA USING MU1
	float discriminateTwo = FORMULA USING MU2

	if(discriminateOne > discriminateTwo)
		return 1
	else
```

```
                return 2
}
```

The Chernoff bound uses an error formula detailed in the technical discussion that is calculated using the mu and sigma values for both samples. The error boundary is then minimized to get the value.

```
float CHERNOFF(Vector2f muOne, Vector2f muTwo, Matrix2f sigmaOne, Matrix2f sigmaTwo)
{
        float index = 0.0;
        float chernoffVal = ERRORFUNCTION(index, muOne, muTwo, sigmaOne, sigmaTwo)

        for(I = 0.0; i <= 1; i += 0.00001)
        {
                float NEWchernoffVal = ERRORFUNCTION(index, muOne, muTwo, sigmaOne, sigmaTwo)
                if(NEWchernoffVal < chernoffVal)
                {
                        chernoffIndex = i
                        chernoffVal = NEWchernoffVal
                }
        }
        return chernofVal;
}
```

The Bhattacharyya bound uses the error function detailed in the technical discussion that is calculated using the mu and sigma values for both samples. It always uses 0.5 as the error boundary.

```
float BHATTACHARYYA(Vector2f muOne, Vector2f muTwo, Matrix2f sigmaOne, Matrix2f sigmaTwo)
{
        return ERRORFUNCTION(0.5, muOne, muTwo, sigmaOne, sigmaTwo)
}
```

Main Implementation

The main function first generates random samples by calling the Box-Muller function twice in order to create a 2D Gaussian distribution. This is done by first creating two matrixes storing sigma values and two vectors storing mu values. Those are then passed to the function to generate samples where the Box-Muller function is called twice for both sigma and mu combinations 100,000 times via a loop. The generate samples function returns a vector of x and y values for the two set of samples, totally 200,000 samples together.

The main function then goes onto complete question 1(a). A loop is ran to use the Bayes classifier case one function to check the return using $P(\omega 1) = P(\omega 2) = 0.5$. If it is found that the discriminate of sample one is bigger than the discriminate of sample two, a counter for sample two's misclassified is iterated and that specific sample in the loop is added to a misclassified vector. If it is found that the discriminate of sample one is smaller than the discriminate of sample two, a counter for sample one's misclassified is iteration and that specific sample in the loop is added to a misclassified vector. Those misclassified in 1(a) are then exported to a csv.

For question 1(b), the same samples are ran through the Bayes classifier case one function again using $P(\omega 1) = 0.2$ and $P(\omega 2) = 0.8$. Those misclassified are also stored into a misclassified vector that are then exported into a csv. Finally, the Chernoff and Bhattacharyya bound for the question 1 samples are calculated via function calling and printed to terminal for the report.

Question 2 is completed by first resetting and clearing all variables and vectors used for question 1. New samples are generated using the new mu and sigma values provided in the assignment. The samples are then passed into a loop running the Bayes classifier case three function to check the return using $P(\omega 1) = P(\omega 2) = 0.5$. If it is found that the discriminate of sample one is bigger than the discriminate of sample two, a counter for sample two's misclassified is iterated and that specific sample in the loop is added to a misclassified vector. If it is found that the discriminate of sample one is smaller than the discriminate of sample two, a counter for sample one's misclassified is iteration and that specific sample in the loop is added to a misclassified vector. Those misclassified in 2(a) are then exported to a csv.

For question 2(b), the same samples are ran through the Bayes classifier case three function again using $P(\omega 1) = 0.2$ and $P(\omega 2) = 0.8$. Those misclassified are also stored into a misclassified vector that are then exported into a csv. Finally, the Chernoff and Bhattacharyya bound for the question 1 samples are calculated via function calling and printed to terminal for the report.

Finally, for question 3 the same samples used in question 2 remain the same, but misclassified vector data is cleared. The samples are passed into the minimum distance classifier function to check the return. If it is found that the discriminate of sample one is bigger than the discriminate of sample two, a counter for sample two's misclassified is iterated and that specific sample in the loop is added to a misclassified vector. If it is found that the discriminate of sample one is smaller than the discriminate of sample two, a counter for sample one's misclassified is iteration and that specific sample in the loop is added to a misclassified vector. Those misclassified in 3 are then exported to a csv.

Results and Discussion

After running the code, the team was able to use the csv files exported of both sample data and misclassified data and create graphs used within this section. The team did this by simply using the excel graphing functions available.

Question 1

We completed Question 1 using generated Gaussian distributions based on the values shown in Figure 12.

$$u_1 = \begin{bmatrix}1\\1\end{bmatrix} \Sigma_1 = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \qquad u_2 = \begin{bmatrix}4\\4\end{bmatrix} \Sigma_2 = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$$

Figure 12: Values used to generate Gaussian distribution.

Question 1(a)

Our posterior probabilities P(ω_i) for Question 1(a) were set $P(\omega_1) = P(\omega_2) = 0.5$.

(i) Design a Bayes classifier for minimum error

The Bayes classifier for minimum error for a two-class problem modeled by a 2D Gaussian Distribution can be obtained using the algorithm described in case one of the three case multivariate Gaussian density discriminant functions for Bayes classification whose equations are included in Fig.10. Assuming every feature is statistically independent from one another, and that each feature has the same variance, our representative discriminant function for each feature vector is the Euclidean distance $g_i(x) = -||x - \mu_i||^2$. Our decision boundary is calculated by setting $g_i(x) = g_j(x)$ which results in $x_0 = \frac{1}{2}(\mu_i + \mu_j)$ after algebraic simplification using our previous assumptions. For the Gaussian distributions generated in Figure 12, the decision boundary is calculated $x = \frac{1}{2}\left(\begin{bmatrix}1\\1\end{bmatrix} + \begin{bmatrix}4\\4\end{bmatrix}\right)$. The decision boundary is therefore $x_1 + x_2 = 5$.

(ii) Plot Bayes decision boundary together with samples.

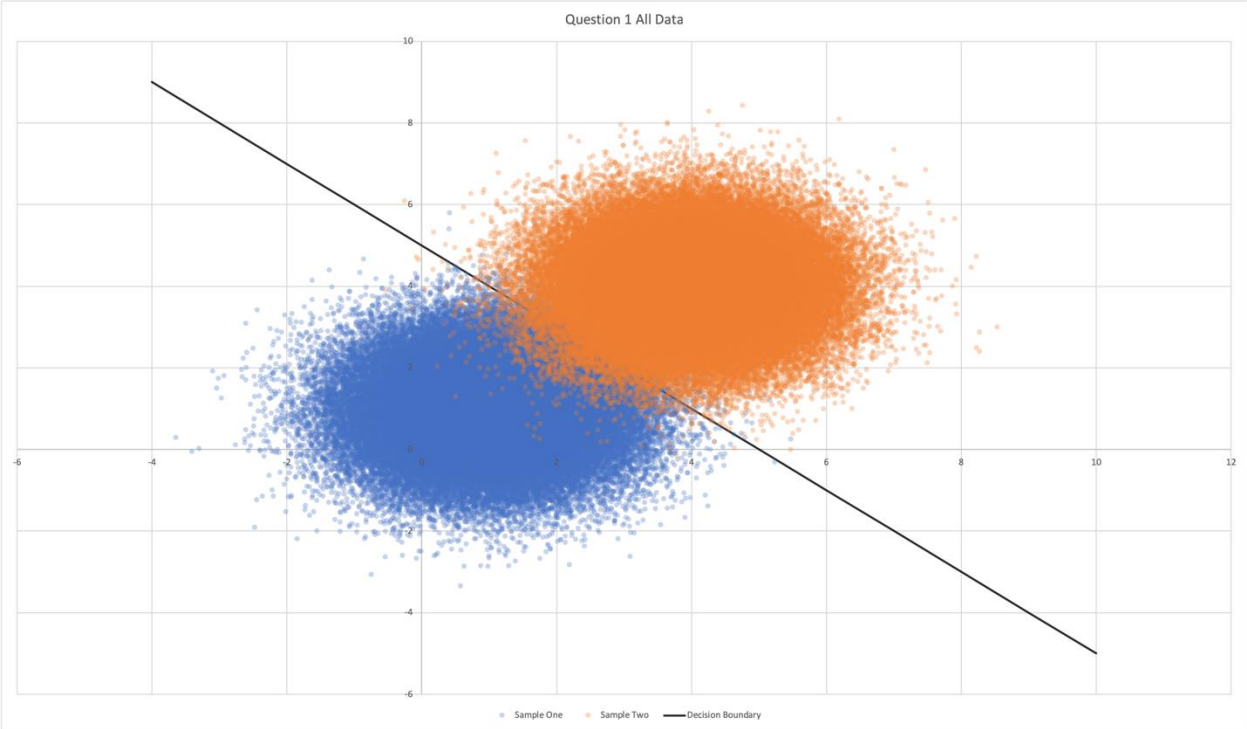


Figure 13: The two plotted samples with decision boundary overlaid for 1(a).

As shown in Figure 13 the decision boundary is roughly in between the two samples which is the expected result given our posterior probabilities. As shown in Figure 14, the misclassified samples are almost equal which is expected with almost equal posterior probabilities.

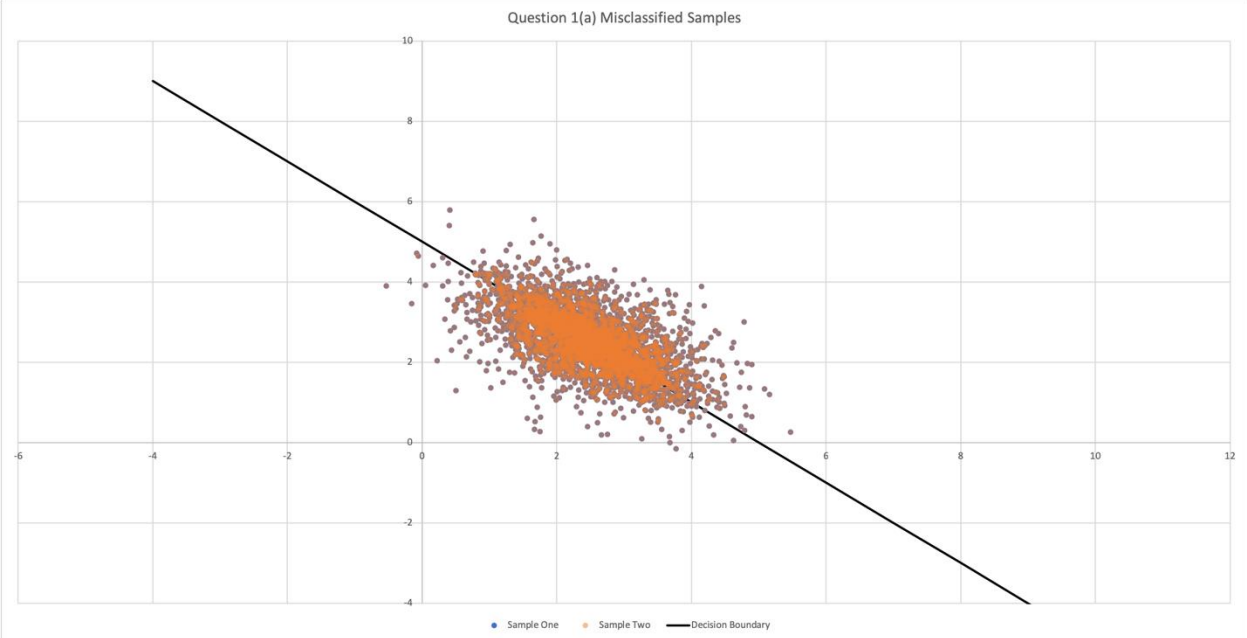


Figure 14: Misclassified samples plotted with the decision boundary overlaid for part 1(a).

(iii) Report (i) the number of misclassified samples for each class separately and (ii) the total number of misclassified samples.

```
QUESTION 1(A):
(i) Design a Bayes classifier for minimum error - DONE
(ii) Plot the Bayes decision boundary together with the generated samples to better visualize and interpret the classification results - REPORT
(iii) Report the number of misclassified samples for each class separately and the total number of misclassified samples...
        Samples from the first 2D Gaussian misclassified: 1658
        Samples from second 2D Gaussian misclassified: 1726
        Total misclassified: 3384
(iv) Plot the Chernoff bound as a function B and find the optimum B for the minimum - REPORT
(v) Calculate the Bhattacharyya bound. Is it close to the experimental error - REPORT
```

Figure 15: Terminal output describing the number of misclassified samples for each class along with the total number of misclassified samples question 1(a).

The number of misclassified samples from each distribution was roughly equal at 1658 and 1726 respectively. The total number of misclassified samples was 3384, roughly 1.692% of samples.

(iv) Plot the Chernoff bound as a function of β and find the optimum β for the minimum.

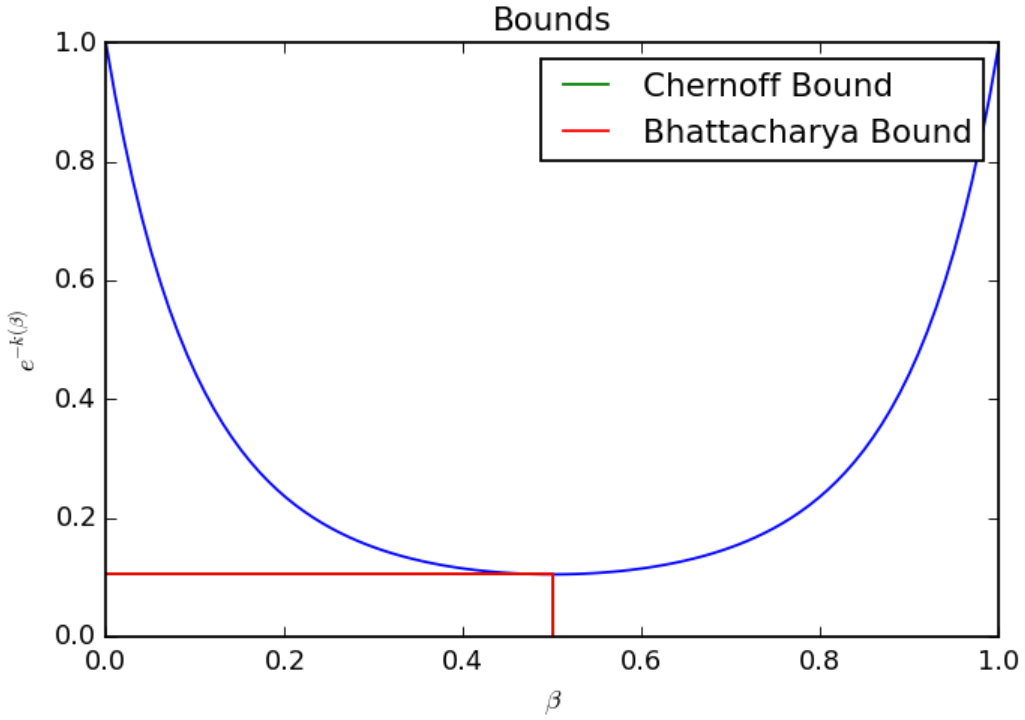


Figure 16: Plotted Chernoff and Bhattacharya Bound.

v. Calculate the Bhattacharyya bound. Is it close to the experimental error?

```
Question 1 error calculations:
With beta = 0.499851, Chernoff Bound = 0.105399
With beta = 0.5, Bhattacharyya Bound = 0.105399
```

Figure 17: Betas used and calculated Chernoff and Bhattacharyya Bounds.

Our Chernoff and Bhattacharyya bound were calculated as completely identical at 0.105399. The experimental error was only 0.01692, far below the value of the Bhattacharyya bound. However, if we were to increase our dimensionality or use different priors, it is quite possible we could reach or exceed our Bhattacharyya bound.

Question 1(b)

Our posterior probabilities P(ω_i) for Question 1(b) were set $P(\omega_1) = 0.2$ and $P(\omega_2) = 0.8$.

(i) Design a Bayes classifier for minimum error.

We use the same Bayes classifier described in question 1 with different posterior values P(ω_i). Our decision boundary is now not as easily simplified as before as we can't assume $P(\omega_1) = P(\omega_2)$ and are left with $x_0 = \frac{1}{2}(\mu_1 + \mu_2) - \frac{\sigma^2}{|u_1 - u_2|^2} \ln \frac{P(\omega_1)}{P(\omega_2)} (\mu_1 - \mu_2)$ which simplifies after plugging in our given values to our decision boundary $x_1 + x_2 = 2.73104$.

(ii) Plot Bayes decision boundary together with samples

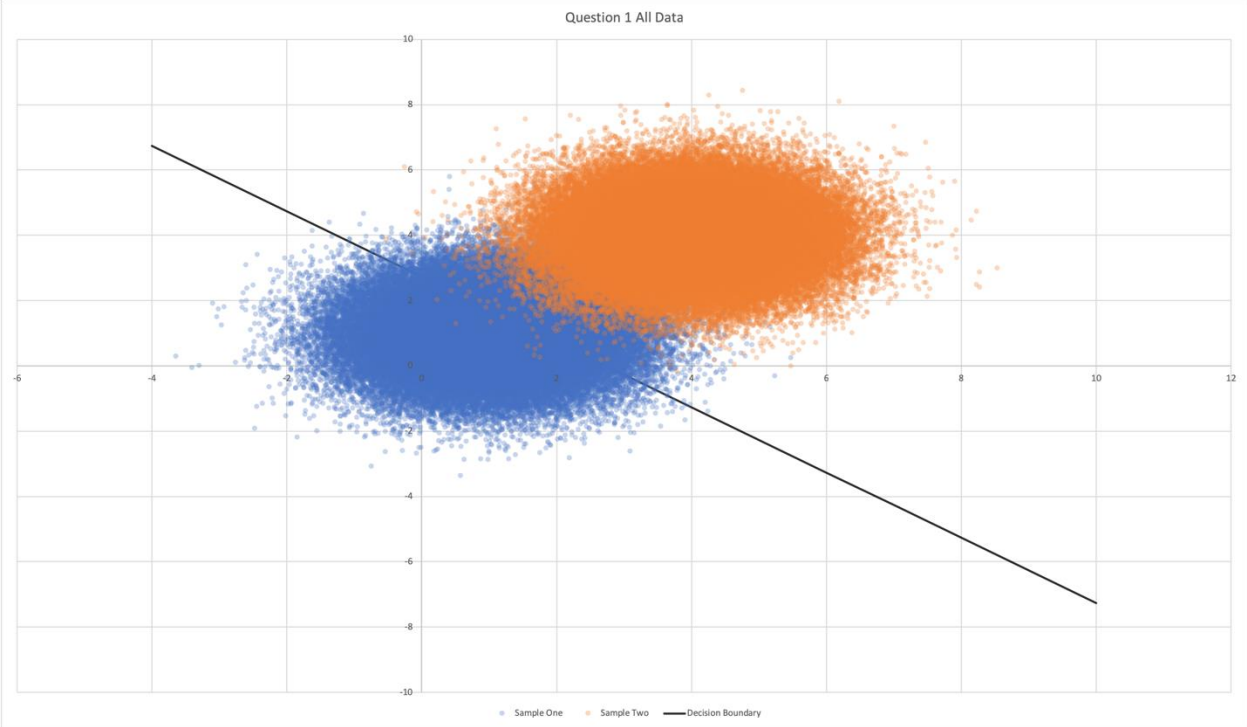


Figure 18: The two plotted samples with decision boundary overlaid for part a question 1(b).

As shown in Figure 18, the decision boundary has shifted to favor results for the more likely posterior probability $P(\omega_2)$. As shown in Figure 19 the number of misclassified samples has increased due to the change in posterior probabilities which directly influenced the new decision boundary.

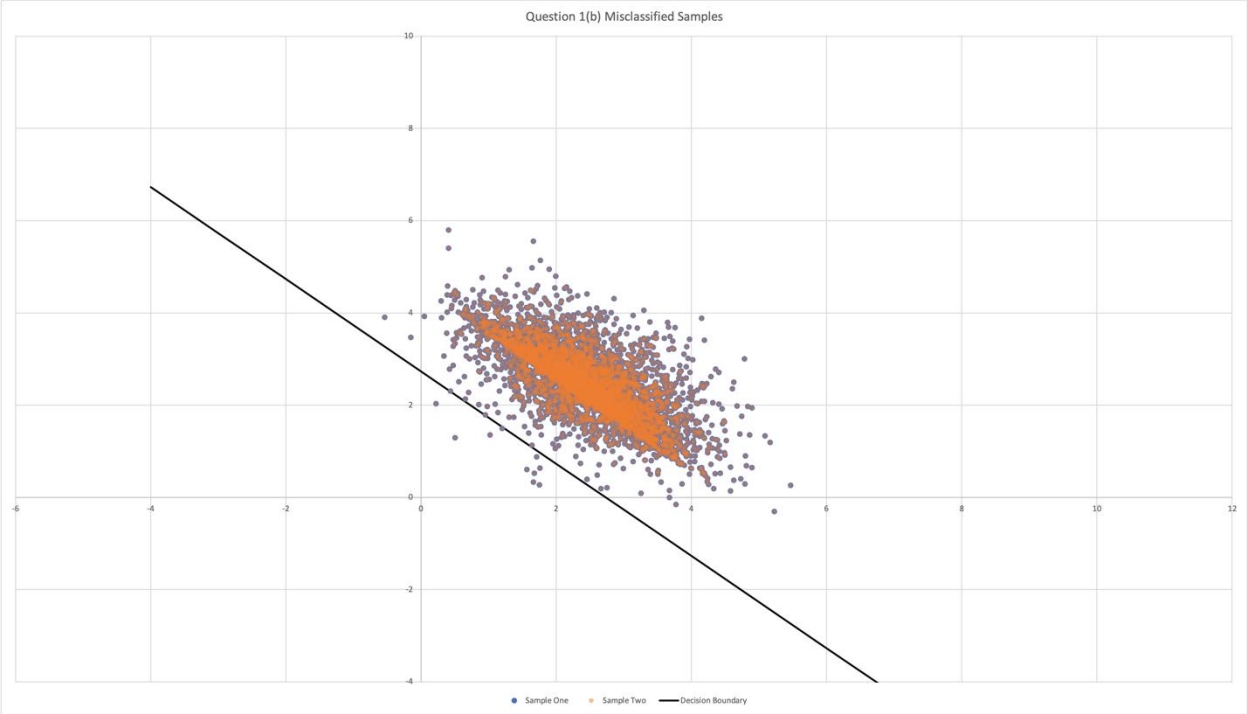


Figure 19: Misclassified samples plotted with the decision boundary overlaid for 1(b).

(iii) Report (i) the number of misclassified samples for each class separately and (ii) the total number of misclassified samples.

```
QUESTION 1(B):
(i) Design a Bayes classifier for minimum error - DONE
(ii) Plot the Bayes decision boundary together with the generated samples to better visualize and interpret the classification results - REPORT
(iii) Report the number of misclassified samples for each class separately and the total number of misclassified samples...
        Samples from the first 2D Gaussian misclassified: 3607
        Samples from second 2D Gaussian misclassified: 738
        Total misclassified: 4345
(iv) Plot the Chernoff bound as a function B and find the optimum B for the minimum - REPORT
(v) Calculate the Bhattacharyya bound. Is it close to the experimental error - REPORT
```

Figure 20: Terminal output describing the number of misclassified samples for each class along with the total number of misclassified samples for question 1(b).

The number of misclassified samples from each distribution was roughly equal at 3607 and 738 respectively. The total number of misclassified samples was 4345, roughly 2.173% of samples.

(iv) Plot the Chernoff bound as a function of β and find the optimum β for the minimum.
See question 1, part a(iv).

(v) Calculate the Bhattacharyya bound. Is it close to the experimental error?
See question 1, part a(v). The experimental error was only 0.021725, far below the value of the Bhattacharyya bound. However, if we were to increase our dimensionality or use different priors, it is quite possible we could reach or exceed our Bhattacharyya bound.

Question 2

We completed Question 2 using generated Gaussian distributions based on the values shown in Figure 21.

$$u_1 = \begin{bmatrix}1\\1\end{bmatrix} \Sigma_1 = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \qquad u_2 = \begin{bmatrix}4\\4\end{bmatrix} \Sigma_2 = \begin{bmatrix}4 & 0\\0 & 8\end{bmatrix}$$

Figure 21: Values used to generate Gaussian distribution.

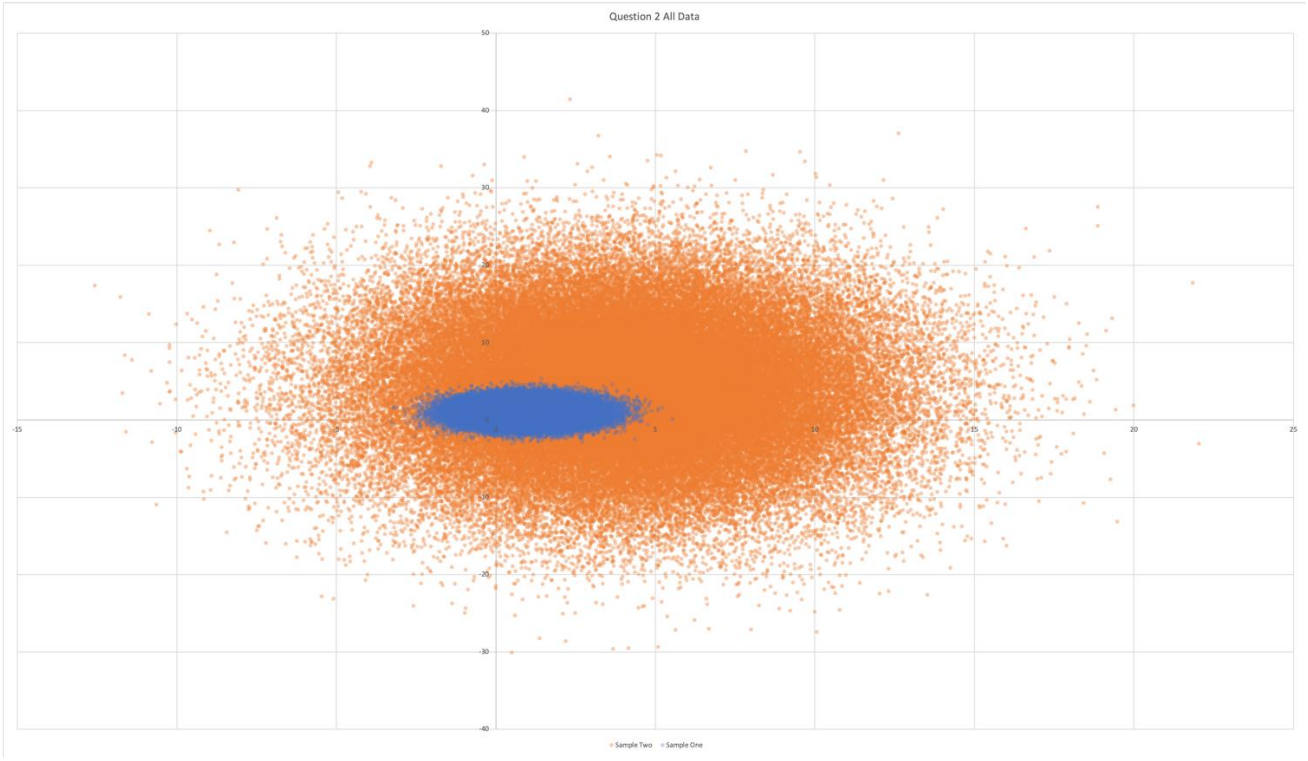
Question 2(a)

Our posterior probabilities P(ω_i) for Question 2(a) were set $P(\omega_1) = P(\omega_2) = 0.5$.

(i) Design a Bayes classifier for minimum error

The Bayes classifier for minimum error is now described by case 3 as the covariance matrices are now arbitrary, and inequal. Our representative discriminant function for each feature vector, disregarding constants becomes $g_i(x) = x^t W_i x + w_i^t x + w_{i0}$ where $W_i = -\frac{1}{2}\Sigma_i^{-1}, w_i = \Sigma_i^{-1}\mu_i$, and $w_{i0} = -\frac{1}{2}\mu_i^t\Sigma_i^{-1}\mu_i - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i)$. Our decision boundary is calculated by setting $g_i(x) = g_j(x)$ which results in hyper quadratic boundary. After algebraic simplification assuming $P(\omega_1) = P(\omega_2)$ and using the Gaussian distributions generated in Figure 21, the decision boundary is calculated to be $x = -.75x_1^2 + 1.125x_2^2 + y + 5.6123$.

(ii) Plot Bayes decision boundary together with samples



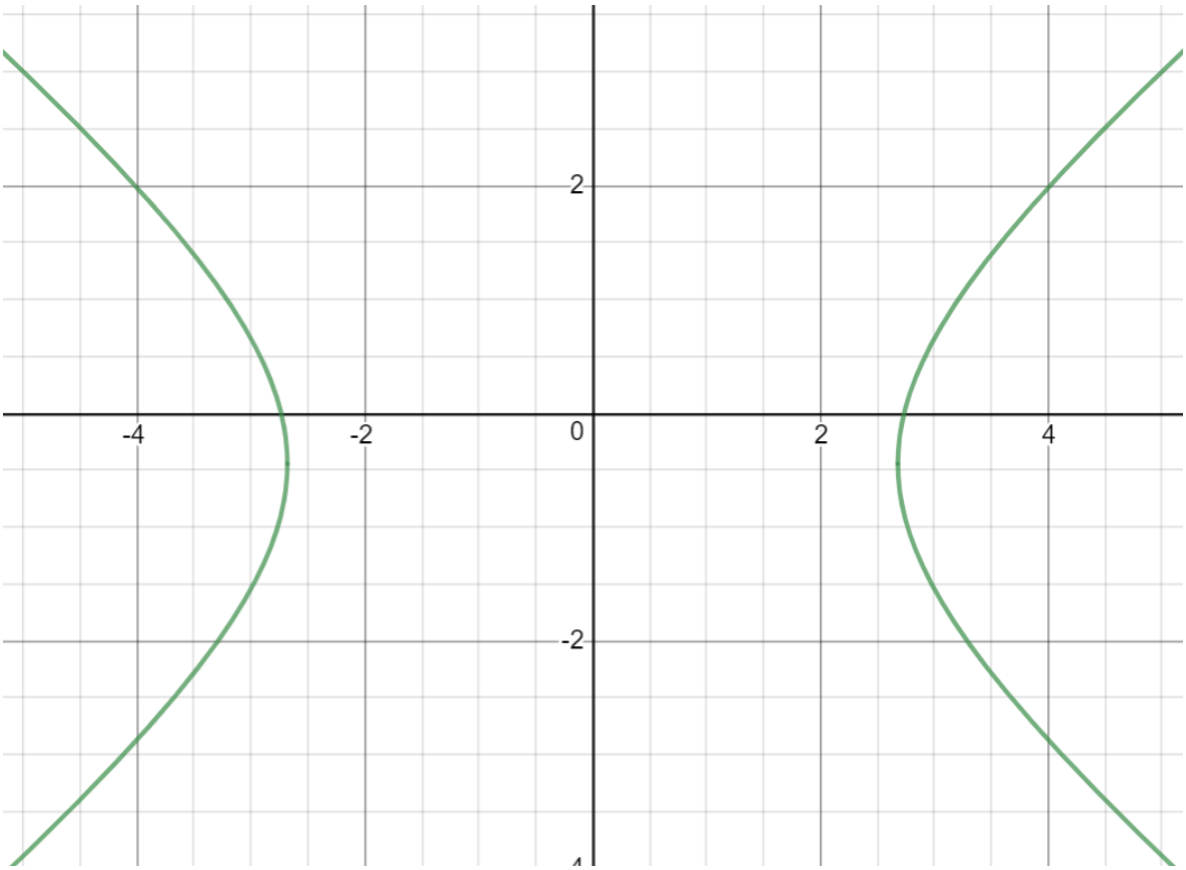


Figure 22: Decision boundary next to plotted generated sample from Figure 21 for 2(a).

Due to the nature of the quadratic, we get a hyperbolic decision boundary. It is shifted to cover a large quantity of the sample data; however, it misses much of the second distribution. This can be seen more clearly in Figure 23. More samples are misclassified from the second distribution.

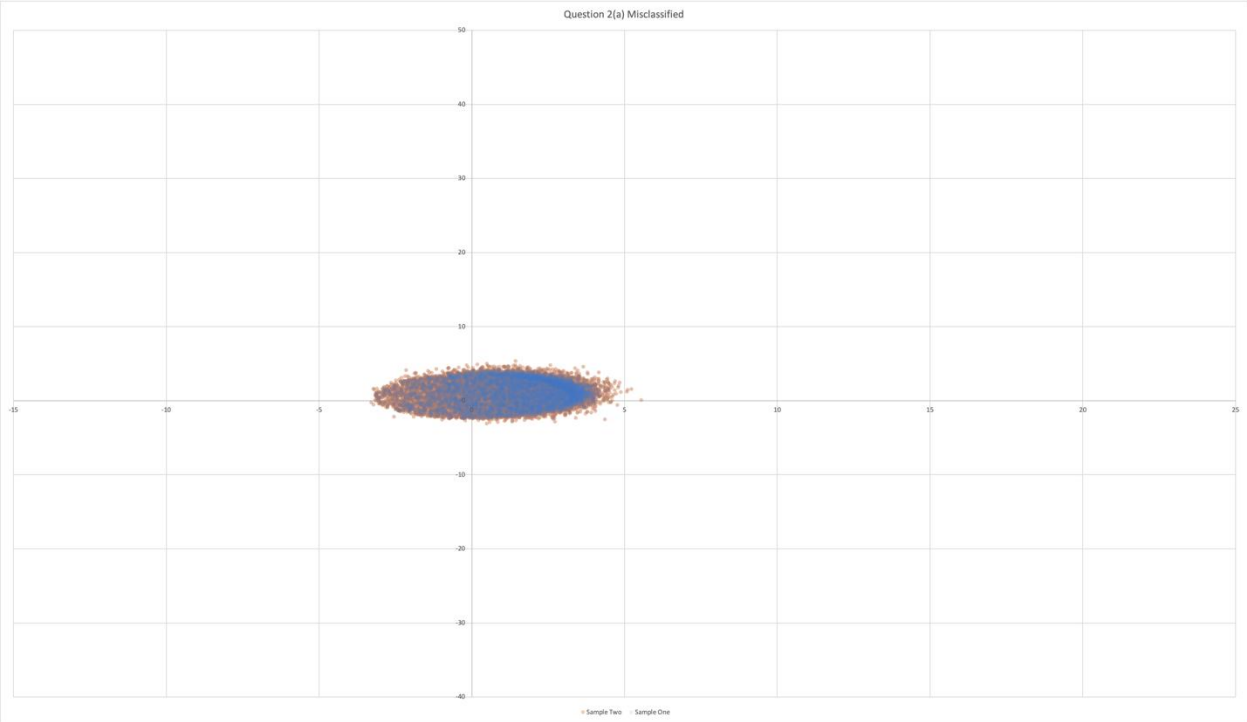


Figure 23: The misclassified samples using the classifier outlined in 2(a).

(iii) Report (i) the number of misclassified samples for each class separately and (ii) the total number of misclassified samples.

```
QUESTION 2(A):
(i) Design a Bayes classifier for minimum error - DONE
(ii) Plot the Bayes decision boundary together with the generated samples to better visualize and interpret the classification results - REPORT
(iii) Report the number of misclassified samples for each class separately and the total number of misclassified samples...
       Samples from the first 2D Gaussian misclassified: 3806
       Samples from second 2D Gaussian misclassified: 8122
       Total misclassified: 11928
(iv) Plot the Chernoff bound as a function B and find the optimum B for the minimum - REPORT
(v) Calculate the Bhattacharyya bound. Is it close to the experimental error - REPORT
```

Figure 24: Terminal output describing the number of misclassified samples for each class along with the total number of misclassified samples for question 2(a).

The number of misclassified samples from distribution 2 at 8122 nearly doubled the number of misclassified samples from distribution 1 at 3806. The total number of misclassified samples was 11928, roughly 5.964% of samples.

(iv) Plot the Chernoff bound as a function of β and find the optimum β for the minimum.

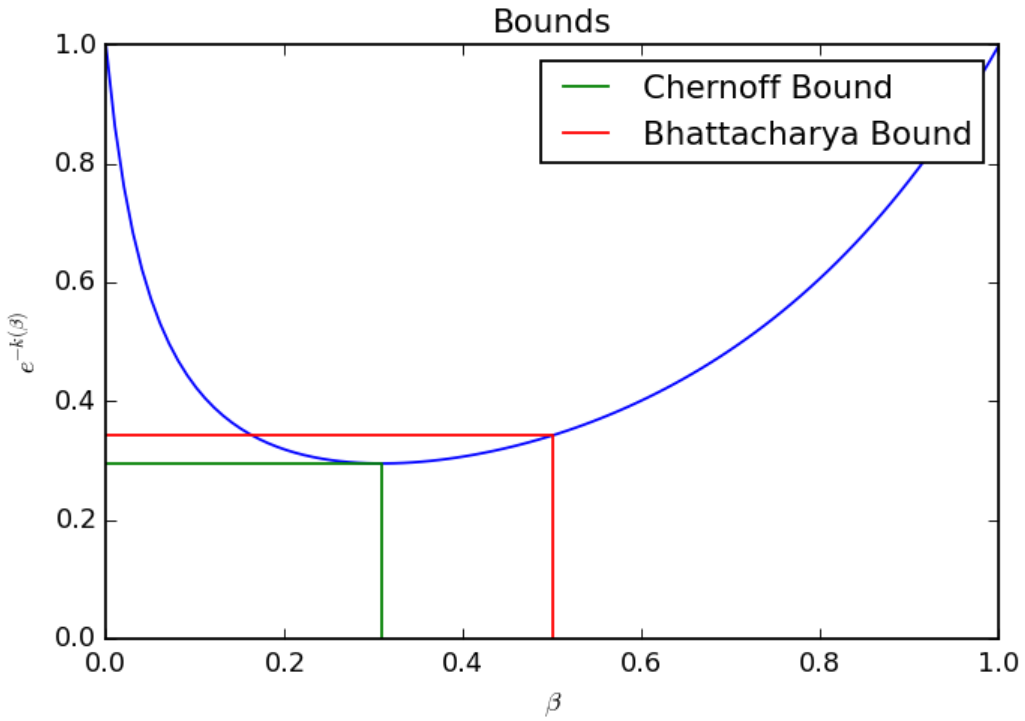


Figure 25: Plotted Chernoff and Bhattacharya Bound.

(v) Calculate the Bhattacharyya bound. Is it close to the experimental error?

```
Question 2 error calculations:
With beta = 0.321879, Chernoff Bound = 0.309878
With beta = 0.5, Bhattacharyya Bound = 0.352132
```

Figure 26: Betas used as well as calculated Chernoff and Bhattacharyya Bounds.

Our Chernoff and Bhattacharyya bound were calculated as similar at 0.309878 and 0.352132, respectively. The experimental error was only 0.05964, far below the value of the Bhattacharyya bound.

Question 2(b)

Our posterior probabilities P(ω_i) for Question 2(b) were set $P(\omega_1) = 0.2$ and $P(\omega_2) = 0.8$.

(i) Design a Bayes classifier for minimum error

Since the posterior probabilities are the only thing to change for part b the decision boundary remains the same with the addition of $\ln\frac{P(\omega_1)}{P(\omega_2)}$. After algebraic simplification using the the Gaussian distributions generated in Figure 20, the decision boundary is calculated to be $x = -.75x_1^2 + 1.125x_2^2 + y + 4.22601$.

(ii) Plot Bayes decision boundary together with samples

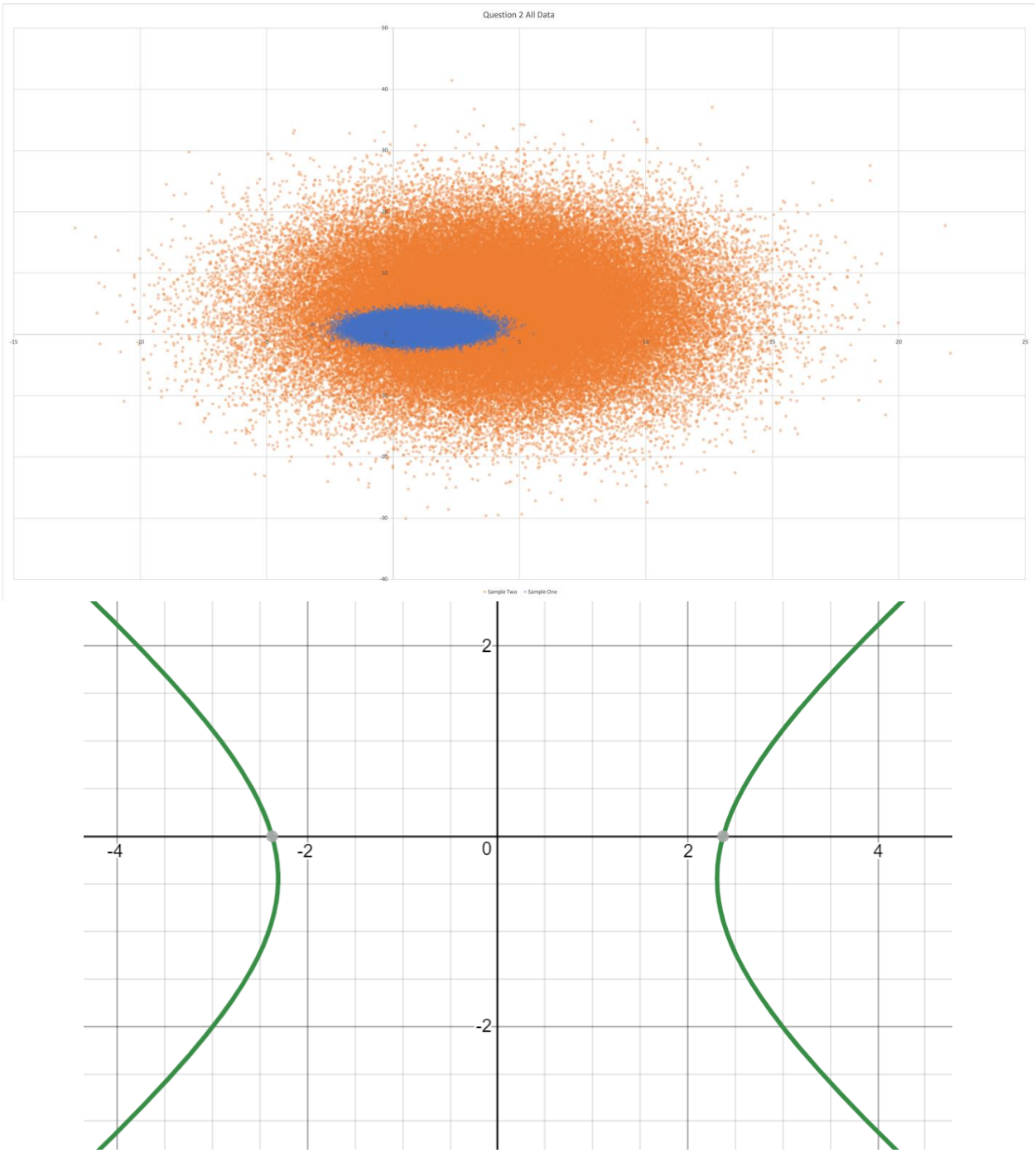


Figure 27: Decision boundary next to plotted generated sample from Figure 21 for question 2(b).

Our hyperbolic decision boundary moves based upon the new posterior probabilities. It is shifted to cover a larger quantity of the sample data; however, it misses much of the first distribution this time which can be seen more clearly in Figure 28.

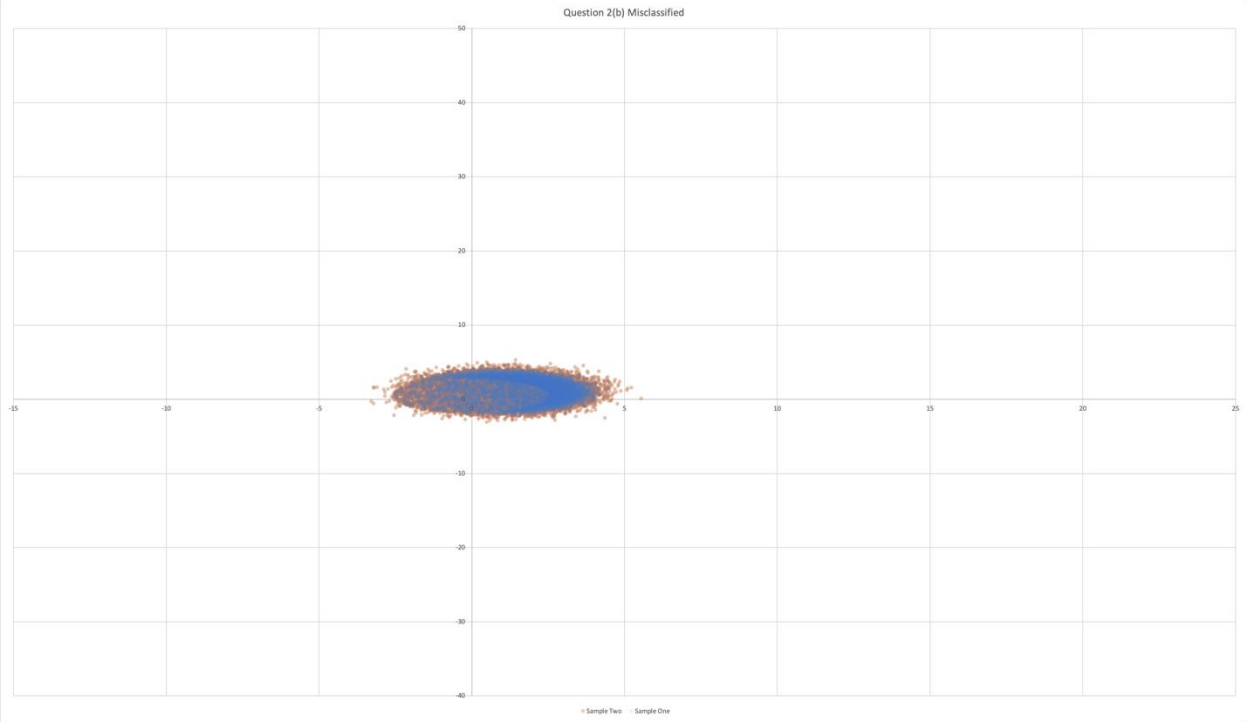


Figure 28: The misclassified samples using the classifier outlined in 2(b).

(iii) Report (i) the number of misclassified samples for each class separately and (ii) the total number of misclassified samples.

```
QUESTION 2(B):
(i) Design a Bayes classifier for minimum error - DONE
(ii) Plot the Bayes decision boundary together with the generated samples to better visualize and interpret the classification results - REPORT
(iii) Report the number of misclassified samples for each class separately and the total number of misclassified samples...
        Samples from the first 2D Gaussian misclassified: 13700
        Samples from second 2D Gaussian misclassified: 5254
        Total misclassified: 18954
(iv) Plot the Chernoff bound as a function B and find the optimum B for the minimum - REPORT
(v) Calculate the Bhattacharyya bound. Is it close to the experimental error - REPORT
```

Figure 29: Terminal output describing the number of misclassified samples for each class along with the total number of misclassified samples for question 2(b).

The number of misclassified samples from distribution 2 at 5,254 had this time less than half of the misclassified samples from distribution 1 at 13,700. The total number of misclassified samples was 18954, roughly 9.477% of samples. As more dimensionality is added, our results are much more error prone.

(iv) Plot the Chernoff bound as a function of β and find the optimum β for the minimum.
See question 2, part a(iv).

(v) Calculate the Bhattacharyya bound. Is it close to the experimental error?
See question 2, part a(v). The experimental error was only 0.09477, below the value of the Bhattacharyya bound. However, if we were to increase our dimensionality or use different priors, it is quite possible we could reach or exceed our Bhattacharyya bound.

Question 3

(ii) Plot Bayes decision boundary together with samples

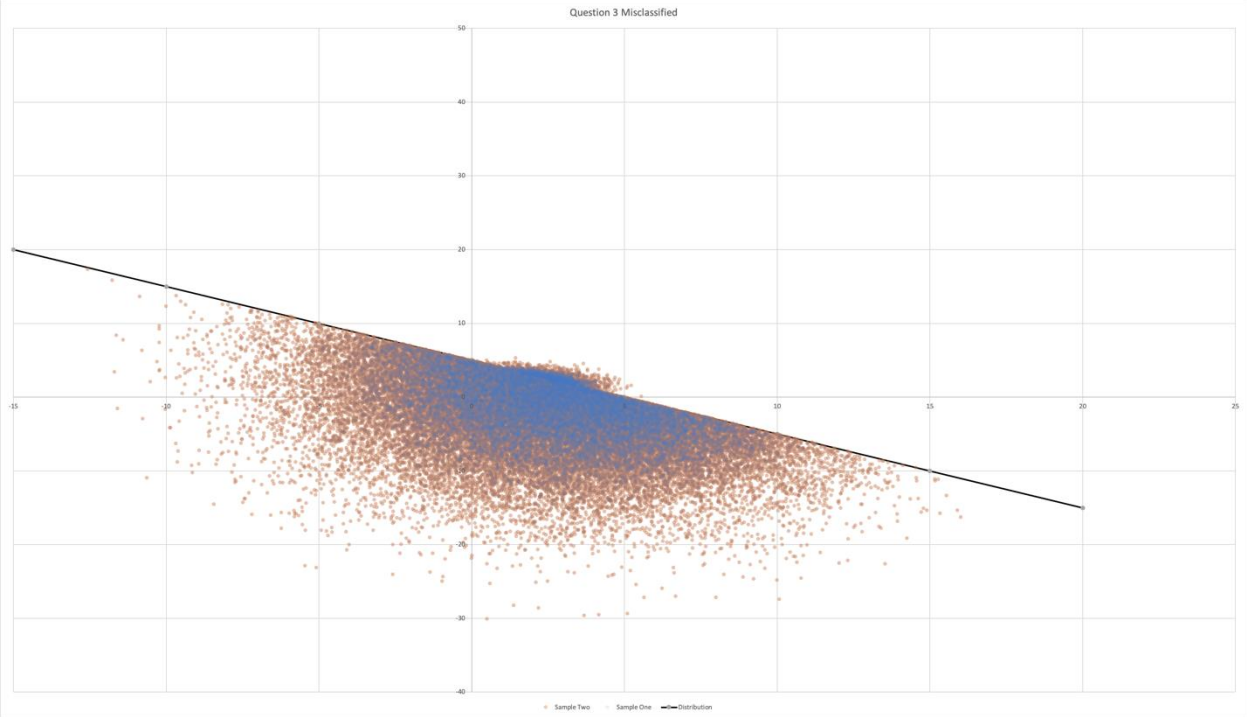


Figure 30: Minimum Euclidean distance classifier applied to the second distribution.

Using the same generated Gaussian distributions from question with parameters described in Figure 21, we applied the minimum Euclidean distance classifier defined in question one part a i to the distribution. As our assumption of equivalent variance is no longer true the classifier does not perform well. The calculated decision boundary $x = \frac{1}{2}\left(\begin{bmatrix}1\\1\end{bmatrix} + \begin{bmatrix}4\\4\end{bmatrix}\right)$ where $x_1 + x_2 = 5$ does not cover many of the values in the more complicated second distribution.

(iii) Report (i) the number of misclassified samples for each class separately and (ii) the total number of misclassified samples.

```
QUESTION 3:
(i) Design a Bayes classifier for minimum error - DONE
(ii) Plot the Bayes decision boundary together with the generated samples to better visualize and interpret the classification results - REPORT
(iii) Report the number of misclassified samples for each class separately and the total number of misclassified samples...
        Samples from the first 2D Gaussian misclassified: 1622
        Samples from second 2D Gaussian misclassified: 37198
        Total misclassified: 38820
```

Figure 31: Terminal output describing the number of misclassified samples for each class along with the total number of misclassified samples for question 3.

As seen in Figure 31, the number of misclassified samples is approximately 19%, exceeding our expected error rate by almost 10%. With this high error rate, we can why it is important to apply the correct classification method to every unique problem.

(iv) Plot the Chernoff bound as a function of β and find the optimum β for the minimum.
See question 2, part a(iv).

(v) Calculate the Bhattacharyya bound. Is it close to the experimental error?
See question 2, part a(v). The experimental error was 0.18599, below the value of the Bhattacharyya bound but close. If we were to increase our dimensionality or use different priors, it is possible we could reach or exceed our Bhattacharyya bound.

Program Listings

Classifier.cpp

```
#include "Classifier.h"

// Generate samples
vector<Vector2f> Classifier::generateSamples(Vector2f mu, Matrix2f sigma)
{
	vector<Vector2f> samples;

	for(int i = 0; i < 100000; i++)
	{
		samples.push_back(Vector2f(box_muller(mu(0,0), sigma(0,0)), box_muller(mu(1,0), sigma(1,1))));
	}

	return samples;
}

// Baye's case one
int Classifier::caseOne(Vector2f x, Vector2f muOne, Vector2f muTwo, float varianceOne, float varianceTwo, float priorOne, float
priorTwo)
{
	float discrimOne = (((1.0/varianceOne) * muOne).transpose() * x) - (1.0/(2*varianceOne)) * squared(muOne);
	float discrimTwo = (((1.0/varianceTwo) * muTwo).transpose() * x) - (1.0/(2*varianceTwo)) * squared(muTwo);

	if(priorOne != priorTwo)
	{
		discrimOne += log(priorOne);
		discrimTwo += log(priorTwo);
	}

	if(discrimOne > discrimTwo)
	{
		return 1;
	}
	else
	{
		return 2;
	}
}

// Baye's case two
int Classifier::caseTwo(Vector2f x, Vector2f muOne, Vector2f muTwo, Matrix2f sigmaOne, Matrix2f sigmaTwo, float priorOne,
float priorTwo)
{
	float discrimOne = ((sigmaOne.inverse() * muOne).transpose() * x)(0) - (0.5 * muOne.transpose() *
sigmaOne.inverse() * muOne);
	float discrimTwo = ((sigmaTwo.inverse() * muTwo).transpose() * x)(0) - (0.5 * muTwo.transpose() *
sigmaTwo.inverse() * muTwo);

	if(priorOne != priorTwo)
	{
		discrimOne += log(priorOne);
		discrimTwo += log(priorTwo);
	}

	if(discrimOne > discrimTwo)
```

```
		return 1;
	else
		return 2;
}

// Baye's case three
int Classifier::caseThree(Vector2f x, Vector2f muOne, Vector2f muTwo, Matrix2f sigmaOne, Matrix2f sigmaTwo, float priorOne,
float priorTwo)
{
	float discrimOne = (x.transpose() * (-0.5 * sigmaOne.inverse()) * x) + ((sigmaOne.inverse() * muOne).transpose() *
x)(0) + (-0.5 * muOne.transpose() * sigmaOne.inverse() * muOne) + (-0.5 * log(sigmaOne.determinant()));
	float discrimTwo = (x.transpose() * (-0.5 * sigmaTwo.inverse()) * x) + ((sigmaTwo.inverse() * muTwo).transpose() *
x)(0) + (-0.5 * muTwo.transpose() * sigmaTwo.inverse() * muTwo) + (-0.5 * log(sigmaTwo.determinant()));

	if(priorOne != priorTwo)
	{
		discrimOne += log(priorOne);
		discrimTwo += log(priorTwo);
	}

	if(discrimOne > discrimTwo)
		return 1;
	else
		return 2;
}

// Minimum distance classifier
int Classifier::minimumDistance(Vector2f x, Vector2f muOne, Vector2f muTwo)
{
	float discrimOne = -1.0 * squared(x-muOne);
	float discrimTwo = -1.0 * squared(x-muTwo);

	if(discrimOne > discrimTwo)
	{
		return 1;
	}
	else
	{
		return 2;
	}
}

// Chernoff
pair<float, float> Classifier::chernoffBound(Vector2f muOne, Vector2f muTwo, Matrix2f sigmaOne, Matrix2f sigmaTwo)
{
	float chernoffIndex = 0.0;
	float chernoffValue = error(chernoffIndex, muOne, muTwo, sigmaOne, sigmaTwo);
	for(float i = 0.0; i <= 1; i += 0.00001)
	{
		float curChernoffValue = error(i, muOne, muTwo, sigmaOne, sigmaTwo);
		if(curChernoffValue < chernoffValue)
		{
			chernoffIndex = i;
			chernoffValue = curChernoffValue;
		}
	}

	return pair<float, float>(chernoffIndex, chernoffValue);
```

```
}

// Bhattacharyya
float Classifier::bhattacharyyaBound(Vector2f muOne, Vector2f muTwo, Matrix2f sigmaOne, Matrix2f sigmaTwo)
{
	return error(0.5, muOne, muTwo, sigmaOne, sigmaTwo);
}

// Norm squared
float Classifier::squared(Vector2f x)
{
	return x.transpose() * x;
}

// Error
float Classifier::error(float beta, Vector2f muOne, Vector2f muTwo, Matrix2f sigmaOne, Matrix2f sigmaTwo)
{

	float kb = (beta*(1-beta))/2.0;
	kb *= (muOne - muTwo).transpose() * ((1-beta)*sigmaOne + (beta)*sigmaTwo).inverse() * (muOne-muTwo);
	kb += 0.5 * log( ((1-beta)*sigmaOne + (beta)*sigmaTwo).determinant() / (pow(sigmaOne.determinant(), 1-beta) *
pow(sigmaTwo.determinant(), beta)));

	return exp(-1.0 * kb);
}
```

Classifier.h

```
#ifndef CLASSIFIER_H
#define CLASSIFIER_H

// Libraries used
#include <iostream>
#include <Eigen/Dense>
#include <vector>
#include <math.h>
using namespace Eigen;
using namespace std;

extern float box_muller(float, float);

class Classifier
{
public:
	// Generate 100,000 samples from a 2D Gaussian distribution
	vector<Vector2f> generateSamples(Vector2f mu, Matrix2f sigma);

	// ** CLASSIFIERS **

	// Baye's cases
	int caseOne(Vector2f, Vector2f, Vector2f, float, float, float, float);
	int caseTwo(Vector2f, Vector2f, Vector2f, Matrix2f, Matrix2f, float, float);
	int caseThree(Vector2f, Vector2f, Vector2f, Matrix2f, Matrix2f, float, float);

	// Minimum distance
	int minimumDistance(Vector2f, Vector2f, Vector2f);

	// ** BOUNDS **
```

```
	// Chernoff bound
	pair<float, float> chernoffBound(Vector2f, Vector2f, Matrix2f, Matrix2f);

	// Bhattacharyya bound
	float bhattacharyyaBound(Vector2f, Vector2f, Matrix2f, Matrix2f);

private:
	// Calculates squared of 2x1 vector
	float squared(Vector2f);

	//Calculates error
	float error(float, Vector2f, Vector2f, Matrix2f, Matrix2f);
};
#endif
```

PA1.cpp

```
// Libraries used
#include <iostream>
#include <Eigen/Dense>
#include <vector>
#include <iostream>
#include <fstream>
using namespace Eigen;
using namespace std;

// Files used
#include "Classifier.h"

// Function used to write to results folder
void samplesFile(const char* fileName, vector<Vector2f> one, vector<Vector2f> two)
{
	// Open file
	ofstream output;
	output.open(fileName);

	output << "x1,y1,x2,y2\n";

	// Output and separate the vectors by spaces
	for(unsigned int i = 0; i < one.size(); i++)
	{
		output << one[i](0) << ",";
		output << one[i](1) << ",";
		output << two[i](0) << ",";
		output << two[i](1) << "\n";
	}

	// Close file
	output.close();
}

int main()
{
	srand(time(NULL));

	// LET'S BEGIN
	Classifier classifier;

	// Vectors used
```

```
	vector<Vector2f> one;
	vector<Vector2f> two;
	vector<Vector2f> misclassified;

	// Sigma/Mu
	Matrix2f sigmaOne;
	Vector2f muOne;
	Matrix2f sigmaTwo;
	Vector2f muTwo;

	// For Chernoff
	pair<float, float> chernoffBound;

	// ** QUESTION 1 SAMPLE GENERATION **

	// Set up parameters
	muOne << 1, 1;
	sigmaOne << 1, 0, 0, 1;
	muTwo << 4, 4;
	sigmaTwo << 1, 0, 0, 1;

	// Variables
	float priorOne = 0.5;
	float priorTwo = 0.5;
	int misclassOne = 0;
	int misclassTwo = 0;

	// Let's make 'em
	one = classifier.generateSamples(muOne, sigmaOne);
	two = classifier.generateSamples(muTwo, sigmaTwo);

	// ** QUESTION 1(A) **
	for(int i = 0; i < 100000; i++)
	{
		if(classifier.caseOne(one[i], muOne, muTwo, sigmaOne(0,0), sigmaTwo(0,0), priorOne, priorTwo) == 2)
		{
			misclassOne++;
			misclassified.push_back(one[i]);
		}
		if(classifier.caseOne(two[i], muOne, muTwo, sigmaOne(0,0), sigmaTwo(0,0), priorOne, priorTwo) == 1)
		{
			misclassTwo++;
			misclassified.push_back(two[i]);
		}
	}

	// Write to file
	samplesFile("./ForReport/1A-Misclassified.csv", misclassified, misclassified);

	// Answer questions
	cout << "QUESTION 1(A):" << endl;
	cout << "(i) Design a Bayes classifier for minimum error - DONE" << endl;
	cout << "(ii) Plot the Bayes decision boundary together with the generated samples to better visualize and interpret
the classification results - REPORT" << endl;
	cout << "(iii) Report the number of misclassified samples for each class separately and the total number of
misclassified samples..." << endl;
	cout << "\tSamples from the first 2D Gaussian misclassified: " << misclassOne << endl;
	cout << "\tSamples from second 2D Gaussian misclassified: " << misclassTwo << endl;
```

```
	cout << "\tTotal misclassified: " << misclassOne + misclassTwo << endl;
	cout << "(iv) Plot the Chernoff bound as a function B and find the optimum B for the minimum - REPORT" << endl;
	cout << "(v) Calculate the Bhattacharyya bound. Is it close to the experimental error - REPORT" << endl;

	// ** QUESTION 1(B) **

	// Reconfigure according to question specs
	misclassOne = 0;
	misclassTwo = 0;
	misclassified.clear();
	priorOne = 0.2;
	priorTwo = 0.8;

	// Calculate
	for(int i = 0; i < 100000; i++)
	{
		if(classifier.caseOne(one[i], muOne, muTwo, sigmaOne(0,0), sigmaTwo(0,0), priorOne, priorTwo) == 2)
		{
			misclassOne++;
			misclassified.push_back(one[i]);
		}
		if(classifier.caseOne(two[i], muOne, muTwo, sigmaOne(0,0), sigmaTwo(0,0), priorOne, priorTwo) == 1)
		{
			misclassTwo++;
			misclassified.push_back(two[i]);
		}
	}

	chernoffBound = classifier.chernoffBound(muOne, muTwo, sigmaOne, sigmaTwo);

	// Write to file
	samplesFile("./ForReport/Question1.csv", one, two);
	samplesFile("./ForReport/1B-Misclassified.csv", misclassified, misclassified);

	// Answer questions
	cout << "\nQUESTION 1(B):" << endl;
	cout << "(i) Design a Bayes classifier for minimum error - DONE" << endl;
	cout << "(ii) Plot the Bayes decision boundary together with the generated samples to better visualize and interpret
the classification results - REPORT" << endl;
	cout << "(iii) Report the number of misclassified samples for each class separately and the total number of
misclassified samples..." << endl;
	cout << "\tSamples from the first 2D Gaussian misclassified: " << misclassOne << endl;
	cout << "\tSamples from second 2D Gaussian misclassified: " << misclassTwo << endl;
	cout << "\tTotal misclassified: " << misclassOne + misclassTwo << endl;
	cout << "(iv) Plot the Chernoff bound as a function B and find the optimum B for the minimum - REPORT" << endl;
	cout << "(v) Calculate the Bhattacharyya bound. Is it close to the experimental error - REPORT" << endl;

	cout << "\nQuestion 1 error calculations:" << endl;
	cout << "With beta = " << chernoffBound.first << ", Chernoff Bound = " << chernoffBound.second << endl;
	cout << "With beta = 0.5, Bhattacharyya Bound = " << classifier.bhattacharyyaBound(muOne, muTwo, sigmaOne,
sigmaTwo) << endl;

	// ** QUESTION 2 SAMPLE GENERATION **
	// Set up parameters
	muOne << 1, 1;
	sigmaOne << 1, 0, 0, 1;
	muTwo << 4, 4;
	sigmaTwo << 4, 0, 0, 8;
```

```
	// Variables
	priorOne = 0.5;
	priorTwo = 0.5;
	misclassOne = 0;
	misclassTwo = 0;

	// Let's make 'em
	one = classifier.generateSamples(muOne, sigmaOne);
	two = classifier.generateSamples(muTwo, sigmaTwo);

	// Clear misclassified
	misclassified.clear();

	// ** QUESTION 2(A) **
	for(int i = 0; i < 100000; i++)
	{
		if(classifier.caseThree(one[i], muOne, muTwo, sigmaOne, sigmaTwo, priorOne, priorTwo) == 2)
		{
			misclassOne++;
			misclassified.push_back(one[i]);
		}
		if(classifier.caseThree(two[i], muOne, muTwo, sigmaOne, sigmaTwo, priorOne, priorTwo) == 1)
		{
			misclassTwo++;
			misclassified.push_back(two[i]);
		}
	}

	// Write to file
	samplesFile("./ForReport/2A-Misclassified.csv", misclassified, misclassified);

	// Answer questions
	cout << "\nQUESTION 2(A):" << endl;
	cout << "(i) Design a Bayes classifier for minimum error - DONE" << endl;
	cout << "(ii) Plot the Bayes decision boundary together with the generated samples to better visualize and interpret
the classification results - REPORT" << endl;
	cout << "(iii) Report the number of misclassified samples for each class separately and the total number of
misclassified samples..." << endl;
	cout << "\tSamples from the first 2D Gaussian misclassified: " << misclassOne << endl;
	cout << "\tSamples from second 2D Gaussian misclassified: " << misclassTwo << endl;
	cout << "\tTotal misclassified: " << misclassOne + misclassTwo << endl;
	cout << "(iv) Plot the Chernoff bound as a function B and find the optimum B for the minimum - REPORT" << endl;
	cout << "(v) Calculate the Bhattacharyya bound. Is it close to the experimental error - REPORT" << endl;

	// ** QUESTION 2(B) **

	// Reconfigure according to question specs
	misclassOne = 0;
	misclassTwo = 0;
	misclassified.clear();
	priorOne = 0.2;
	priorTwo = 0.8;

	// Calculate
	for(int i = 0; i < 100000; i++)
	{
		if(classifier.caseThree(one[i], muOne, muTwo, sigmaOne, sigmaTwo, priorOne, priorTwo) == 2)
```

```
		{
			misclassOne++;
			misclassified.push_back(one[i]);
		}
		if(classifier.caseThree(two[i], muOne, muTwo, sigmaOne, sigmaTwo, priorOne, priorTwo) == 1)
		{
			misclassTwo++;
			misclassified.push_back(two[i]);
		}
	}

	chernoffBound = classifier.chernoffBound(muOne, muTwo, sigmaOne, sigmaTwo);

	// Write to file
	samplesFile("./ForReport/Question2.csv", one, two);
	samplesFile("./ForReport/2B-Misclassified.csv", misclassified, misclassified);

	// Answer questions
	cout << "\nQUESTION 2(B):" << endl;
	cout << "(i) Design a Bayes classifier for minimum error - DONE" << endl;
	cout << "(ii) Plot the Bayes decision boundary together with the generated samples to better visualize and interpret the classification results - REPORT" << endl;
	cout << "(iii) Report the number of misclassified samples for each class separately and the total number of misclassified samples..." << endl;
	cout << "\tSamples from the first 2D Gaussian misclassified: " << misclassOne << endl;
	cout << "\tSamples from second 2D Gaussian misclassified: " << misclassTwo << endl;
	cout << "\tTotal misclassified: " << misclassOne + misclassTwo << endl;
	cout << "(iv) Plot the Chernoff bound as a function B and find the optimum B for the minimum - REPORT" << endl;
	cout << "(v) Calculate the Bhattacharyya bound. Is it close to the experimental error - REPORT" << endl;

	cout << "\nQuestion 2 error calculations:" << endl;
	cout << "With beta = " << chernoffBound.first << ", Chernoff Bound = " << chernoffBound.second << endl;
	cout << "With beta = 0.5, Bhattacharyya Bound = " << classifier.bhattacharyyaBound(muOne, muTwo, sigmaOne, sigmaTwo) << endl;

	// ** QUESTION 3 **

	// Reconfigure according to question specs
	misclassOne = 0;
	misclassTwo = 0;
	misclassified.clear();

	// Calculate
	for(int i = 0; i < 100000; i++)
	{
		if(classifier.minimumDistance(one[i], muOne, muTwo) == 2)
		{
			misclassOne++;
			misclassified.push_back(one[i]);
		}
		if(classifier.minimumDistance(two[i], muOne, muTwo) == 1)
		{
			misclassTwo++;
			misclassified.push_back(two[i]);
		}
	}

	// Write to file
```

```
	samplesFile("./ForReport/3-Misclassified.csv", misclassified, misclassified);

	// Answer questions
	cout << "\nQUESTION 3:" << endl;
	cout << "(i) Design a Bayes classifier for minimum error - DONE" << endl;
	cout << "(ii) Plot the Bayes decision boundary together with the generated samples to better visualize and interpret the classification results - REPORT" << endl;
	cout << "(iii) Report the number of misclassified samples for each class separately and the total number of misclassified samples..." << endl;
	cout << "\tSamples from the first 2D Gaussian misclassified: " << misclassOne << endl;
	cout << "\tSamples from second 2D Gaussian misclassified: " << misclassTwo << endl;
	cout << "\tTotal misclassified: " << misclassOne + misclassTwo << endl;
	cout << "(iv) Plot the Chernoff bound as a function B and find the optimum B for the minimum - REPORT" << endl;
	cout << "(v) Calculate the Bhattacharyya bound. Is it close to the experimental error - REPORT" << endl;

	// Finish main
	return 0;
}
```

box-muller.cpp

```
#ifndef BOXMULLER
#define BOXMULLER

/* 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>
#include <cstdlib>
#include <time.h>

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

float ranf()
{
	return (float)rand() / (float)RAND_MAX;
}

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 );
}
#endif
```