CS 679: Pattern Recognition

University of Nevada, Reno - Spring 2024
Assignment 1
Jaleesa Houle
Due: March 11th, 2024

Statement: "I declare that all material in this assignment is my own work except where there is clear acknowledgment or reference to the work of others. I understand that both my report and code may be subjected to plagiarism detection software, and fully accept all consequences if found responsible for plagiarism, as explained in the syllabus, and described in UNR's Academic Standards Policy: UAM 6,502."

1. Theory

Part 1

General background and theory

The core of any Bayes classifier is Bayes formula, which states:

$$P(\omega_j|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_j)P(w_j)}{p(\mathbf{x})},\ p(\mathbf{x}) = \sum_{j=1}^{c} p(\mathbf{x}|\omega_j)P(w_j) \tag{1}$$

Here, $P(\omega_j|\mathbf{x})$ is the conditional probability that data belongs to a certain class (ω_j) given that it has features $\mathbf{x}$(also called the posterior probability). Similarly, $p(\mathbf{x}|\omega_j)$ is the conditional probability that the data has those ($\mathbf{x}$) features given the data belongs to ω_j (i.e. likelihood probability). $P(w_j)$ is the prior probability of how likely data is to belong to a class ω_j, and p($\mathbf{x}$) is the probability density of $\mathbf{x}$ (i.e., evidence).

In designing a Bayes classifier for multivariate data, this formula will become the basis of our discriminant functions, which are a set of equations $g_i(\mathbf{x})$ for $i = 1, ..., n$ through which we are able to assign $\mathbf{x}$ features to a class ω_i by choosing ω_i if $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$. We will first let

$$g_i(\mathbf{x}) = p(\mathbf{x}|\omega_i)P(w_i). \tag{2}$$

Then, taking the natural log of this equation gives

$$g_i(\mathbf{x}) = ln(p(\mathbf{x}|\omega_i)) + ln(P(w_i)). \tag{3}$$

This manipulation makes $g_i(\mathbf{x})$ a monotonically increasing function, which helps to simplify the classification process. For this project, we are asked to classify two categories of data. Thus, a dichotomizer can be used to classify the data, and can be defined as:

$$g(\mathbf{x}) = g_1(\mathbf{x}) - g_2(\mathbf{x}). \tag{4}$$

In this case, we will choose ω_1 if $g(\mathbf{x}) > 0$ and ω_2 if $g(\mathbf{x}) < 0$. The corresponding decision boundary for the two categories can be found by setting $g_1(\mathbf{x}) = g_2(\mathbf{x})$.

Now, if we assume $p(\mathbf{x}|\omega_j) \sim N(\mu_i, \Sigma_i)$, we can substitute the probability density function (pdf) of a normal distribution into Equation 3. For a multivariate normal distribution,

$$N(\mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} exp[-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)],\ \mathbf{x} \in R^d. \tag{5}$$

Substituting this expression into Equation 3 gives the discriminant function:

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\mu_i)^T\Sigma_i^{-1}(\mathbf{x}-\mu_i) - \frac{d}{2}ln(2\pi) - \frac{1}{2}ln(|\Sigma_i|) + ln(P(\omega_i)). \quad (6)$$

Equation 6 is the general discriminant function used for multivariate Gaussian data, and is what will be used to classify the data generated in this assignment. As discussed in class, there are 3 different scenarios in which Equation 6 can be further discussed: case I ($\Sigma_i = \sigma^2 I$ for each w_i), case II ($\Sigma_i = \Sigma$ for each ω_i), and case III ($\Sigma_i =$ arbitrary for each ω_i).

Classifier design based on case I parameters

In the most simple case, (i.e., case I), we assume the features $\mathbf{x}$ for each ω_i are uncorrelated with the same variance. If this is true, the discriminant can be greatly simplified (by ignoring the constants $\frac{d}{2}ln(2\pi)$ and $\frac{1}{2}ln(\Sigma_i)$) as

$$g_i(\mathbf{x}) = \frac{-||\mathbf{x}-\mu||^2}{2\sigma^2} + ln(P(\omega_i)). \quad (7)$$

By expanding and further simplifying this equation, we can see that the discriminant function for case I is linear, with

$$g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + w_{i0} \quad (8)$$

where:

$$\mathbf{w}_i = \frac{1}{\sigma^2}\mu_i, \mathrm{w}_{i0} = \frac{-1}{2\sigma^2}\mu_i^T\mu_i + ln(P(\omega_i)). \quad (9)$$

For the data in sample set A, $\Sigma_1 = \Sigma_2 = \sigma^2$. Therefore, the conditions for case I are satisfied and we can use this simplified discriminant form to determine the decision boundary and classify the data from sample A.

By calculating the values of w_i and w_{i0} for each given $[\mu_i, \Sigma_i]$ in sample set A, we can construct two discriminant functions ($g_1(x)$ and $g_2(x)$) for each dataset. Then, we can set those two functions equal to each other in order to determine the decision boundary and classify the samples.

Estimating prior probabilities

For both sample set A and B, we are given the instruction to produce n samples from ω_1 and m samples from ω_2. We can use these numbers to determine the priors as $P(w_1) = \frac{n}{n+m}$ and $P(\omega_2) = \frac{m}{n+m}$. However, if we did not have an idea of how many samples came from each class, we could instead choose to estimate priors by them equal to each other (which may or may not be a valid assumption).

Determining misclassification rates

This assignment asks us to report the misclassification rate of each class and the total misclassification rate. For determining misclassification rates, we can compare the true class that each sample came from with the expected class for each sample (determined by assigning each sample ω_1 if $g(\mathbf{x}) > 0$, else ω_2). The misclassification rate for each class can then be determined as n_i/N_i, where n_i is the number of samples misclassified and N_i is the total number of samples produced from class ω_i. The total misclassification rate can be determined as $(n_1 + n_2)/(N_1 + N_2)$, i.e., the total number of samples misclassified divided by the total number of samples. These results should be our true empirical error rates for the given samples.

Assessing probability of error (Bhattacharyya bound)

The theoretical probability error is often determined using the Bhattacharyya error bound, which is defined as

$$P(error) \leq P^{\beta}(\omega_1)P^{1-\beta}(\omega_2)e^{-k(\beta)} \tag{10}$$

where

$$k(\beta) = \frac{\beta(1-\beta)}{2}(\mu_1-\mu_2)^T[(1-\beta)\Sigma_1+\beta\Sigma_2]^{-1}(\mu_1-\mu_2)+\frac{1}{2}ln\left(\frac{|(1-\beta)\Sigma_1+\beta\Sigma_2|}{|\Sigma_1|^{1-\beta}|\Sigma_2|^{\beta}}\right). \tag{11}$$

For the Bhattacharyya error bound, β is set to 0.5, as opposed to the Chernoff error bound, which finds the β which minimizes Equation 10. The probability of error found using the Bhattacharyya bound will never be the most accurate error bound due to the looser approximation of β, therefore the true error rate is expected to be less than or equal to the probability of error estimated through the Bhattacharyya error bound.

Part 2

Classifier design based on case III parameters

In the case of sample set B, the covariance matrices are not equal. Since $\Sigma_1 \neq \Sigma_2$ and all features in ω_2 do not have the same variance, this data can not be classified using the discriminant found in case I. Instead, the optimum classifier for this data will be case III, which yields a quadratic discriminant function. By rewriting Equation 6 and ignoring the constant $\frac{d}{2}ln(2\pi)$, the discriminant for case III can be written as

$$g_i(\mathbf{x}) = \mathbf{x}^T\mathbf{W}_i\mathbf{x} + \mathbf{w}_i^T\mathbf{x} + w_{i0} \tag{12}$$

where:

$$\mathbf{W}_i = -\frac{1}{2}\Sigma_i^{-1}, \mathbf{w}_i = \Sigma_i^{-1}\mu_i, \text{ and } \mathrm{w}_{i0} = -\frac{1}{2}\mu_i^T\Sigma_i^{-1}\mu_i - \frac{1}{2}ln(|\Sigma_i|) + ln(P(w_i)). \tag{13}$$

It can be noted that for case III, we no longer make the assumption that the features are uncorrelated (i.e., the covariance matrices do not need to be diagonal). This is also true for case II, though we will not discuss case II any further since neither the data from sample set A or B fall into this category. For dataset B, the priors, misclassification rates, and probability of error will be computed the same way as is described in Part 1 above.

Parts 3 & 4

Euclidean distance classifier theory and assumptions

In the case that $\Sigma_i = \sigma^2 I$ for all ω_i and $P(\omega_i) = P(\omega_j)$ for $i \neq j$, the discriminant function found above (Equation 7) can be further simplified to what is known as the Euclidean distance classifier. In this case, $g_i(\mathbf{x})$ can be described as:

$$\begin{aligned} g_i(\mathbf{x}) &= -||\mathbf{x} - \mu_i||^2 \\ &= -(\mathbf{x} - \mu_i)^T(\mathbf{x} - \mu_i). \end{aligned} \tag{14}$$

For this to be an optimum classifier (i.e., error is minimized), all priors must be equal, and the covariance matrices for all ω_i must satisfy $\Sigma_i = \sigma^2 I$. As is true for cases I, II, and III, the assumption that all features are normally distributed must also hold true for the Euclidean distance classifier to be optimum.

2. Results and Discussion

Part 1

a. Classification parameters and resulting discriminant functions

The given parameters for sample set A are shown in Table 1, alongside the priors, which were calculated as $60,000/200,000$ for ω_1 and $140,000/200,000$ for ω_2.

Sample Parameters	Values
μ_1^T	[1, 1]
Σ_1	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
μ_2^T	[4, 4]
Σ_2	$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$
P(ω_1, ω_2)	[0.3, 0.7]

Table 1: Given parameters and calculated priors for data set A.

$g_i(\mathbf{x})$ Parameters	Values
w_1	$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$
w_2	$\begin{bmatrix} 4 \\ 4 \end{bmatrix}$
w_{10}	-2.204
w_{20}	-16.357

Table 2: Values for w_i and w_{i0} for dataset A, found by using Equation 9 above.

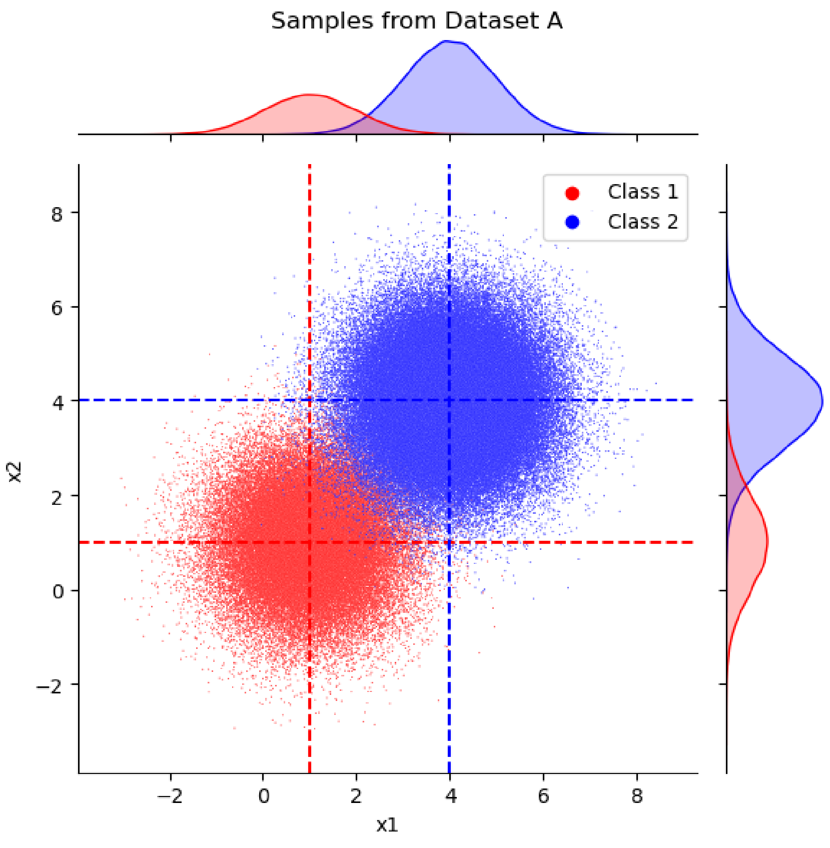


Figure 1: Sample data from distribution parameters [μ_1,Σ_1] and [μ_2, Σ_2]. Mean lines are included to show the center point for each randomly generated sample.

It can be seen in Figure 1 that the data from sample A is spherical and centered at μ_1 and μ_2 for the respective classes. Additionally, the marginal pdfs shown for x_1 and x_2 follow Gaussian distributions with minimal overlap.

The given covariance matrices for sample set A are of the form $\Sigma_1 = \Sigma_2 = \sigma^2 I$, and therefore the case I discriminant was optimum for this data. Table 2 lists the corresponding w_i and w_{i0} values found using Equation 9. By plugging in the values from Table 2 into Equation 8, the discriminant functions for dataset A were found to be:

$$\begin{aligned} g_1(\mathbf{x}) &= x_1 + x_2 - 2.204 \\ g_2(\mathbf{x}) &= 4x_1 + 4x_2 - 16.357 \end{aligned} \tag{15}$$

As expected, $g_1(\mathbf{x})$ and $g_2(\mathbf{x})$ are linear equations.

b. Visualizing the decision boundary

By setting $g_1(\mathbf{x}) = g_2(\mathbf{x})$ and solving for x_2, we are able to plot the decision boundary along side the classified data.

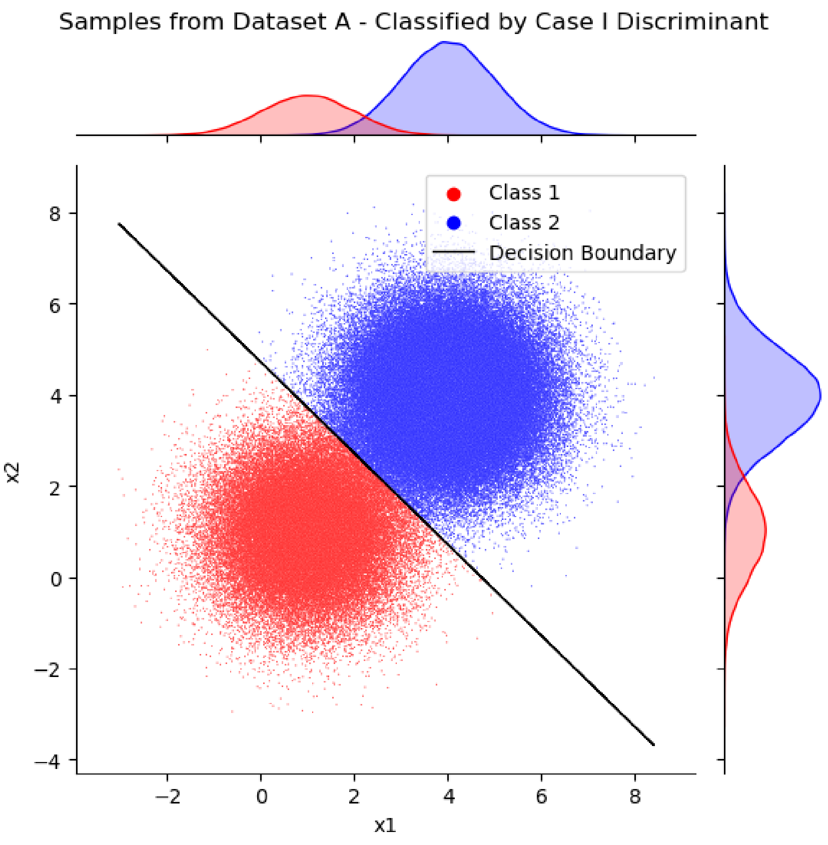


Figure 2: Sample set A, classified using a Case I linear discriminant with decision boundary $x_2 = 4.718 - x_1$.

Figure 2 demonstrates that any sample points below the decision boundary were assigned class 1 and any points above the decision boundary were assigned to class 2. It is clear from a visual inspection that some of the data was misclassified due to areas of overlap between the two distributions.

c. Misclassification rates

The misclassification rates for class 1 and class 2 were 2.67% and 1.01%, respectively. The overall rate of misclassification across sample set A was 1.51%. These rates are very low and demonstrate a high success rate in classification.

d. Theoretical probability error (e.g., Bhattacharyya bound)

Using the Bhattacharyya error bound, the probability of error was found to be $P(error) \leq 4.83\%$. This probability of error is higher than the true misclassification rate of 1.51%. The fact that the theoretical probability error is higher than the true error rate is not surprising, as the Bhattacharyya error bound does not use a β which minimizes Equation 10. Still, this probability of error is quite low, and indicates that at least 95% of any future samples generated with the sample parameters listed in Table 1 should be correctly classified.

Part 2

The given parameters for sample set B are shown in Table 3. It is clear upon examination of Σ_1 and Σ_2 that the covariance matrices are not equal, nor are all of the covariances in Σ_2 the same, therefore, case III was optimum for this data. We can see from Figure 3 that shapes of ω_2 is now ellipsoidal, and there is a significant amount of overlap between ω_1 and ω_2.

Sample Parameters	**Values**
μ_1^T	[1, 1]
Σ_1	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
μ_2^T	[4, 4]
Σ_2	$\begin{bmatrix} 4 & 0 \\ 0 & 8 \end{bmatrix}$
P(ω_1, ω_2)	[0.3, 0.7]

Table 3: Given parameters and calculated priors for data set B.

$g_i(\mathbf{x})$ **Parameters**	**Values**
W_1	$\begin{bmatrix} -0.5 & 0 \\ 0 & -0.5 \end{bmatrix}$
W_2	$\begin{bmatrix} -0.125 & 0 \\ 0 & -0.0625 \end{bmatrix}$
w_1^T	[1, 1]
w_2^T	[1, 0.5]
w_{10}	-2.204
w_{20}	-5.089

Table 4: Values calculated for discriminant equations $g_1(\mathbf{x})$ and $g_1(\mathbf{x})$

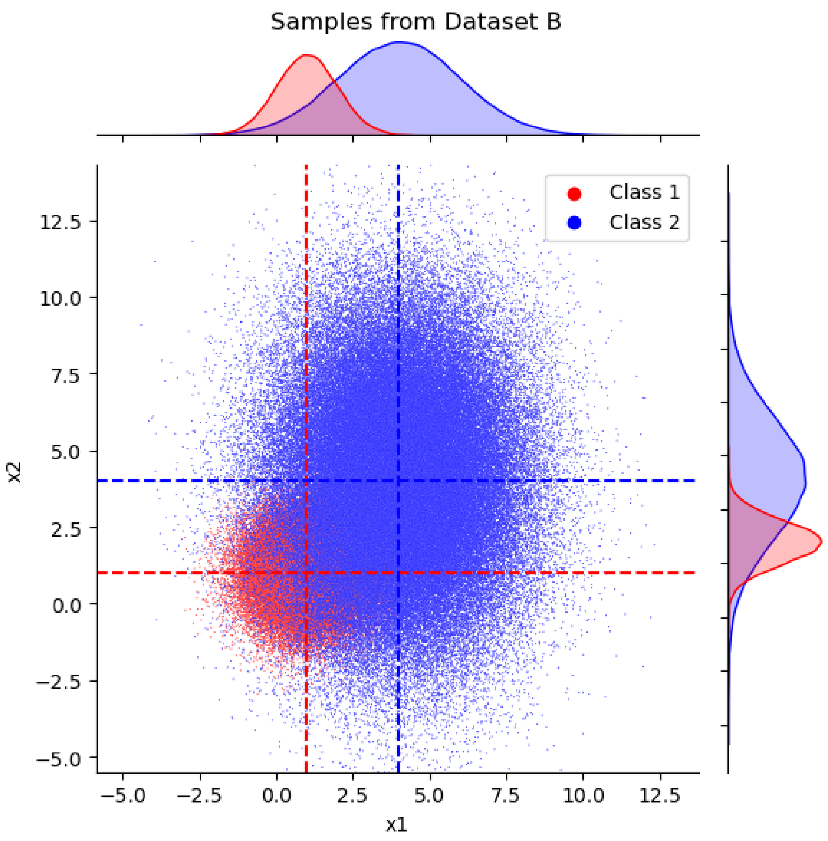


Figure 3: Sample data from distribution parameters $[\mu_1,\Sigma_1]$ and $[\mu_2, \Sigma_2]$. Mean lines are included to show the center point for each randomly generated sample.

Table 4 lists the corresponding W_i, w_i, and w_{i0} values found using Equation 9. By plugging in the values from Table 4 into Equation 8, the discriminant functions for dataset B were found to be:

$$\begin{aligned} g_1(\mathbf{x}) &= -0.5x_1^2 + x_1 - 0.5x_2^2 + x_2 - 2.204 \\ g_2(\mathbf{x}) &= -0.125x_1^2 + x_1 - 0.0625x_2^2 + 0.5x_2 - 5.0895. \end{aligned} \tag{16}$$

Once again setting $g_1(\mathbf{x}) = g_2(\mathbf{x})$ and solving for x_2 resulted in the decision boundary equation $x_2 = \{0.571 \pm 2.631\sqrt{1 - 0.124x_1^2}\}$, $-2.842 < x_1 < 2.842$. This decision boundary is no longer linear, but is instead ellipsoidal, as is evident in Figure 4.

The misclassification rate for classes 1 and 2 in data set B were 8.14% and 7.29%, respectively. The total misclassification rate across sample set B was 7.55%, and the Bhattacharyya bound was found to be $P(error) \leq 16.14\%$. Again, it makes sense for the Bhattacharyya bound to be higher than the true error rate, as β was not optimized.

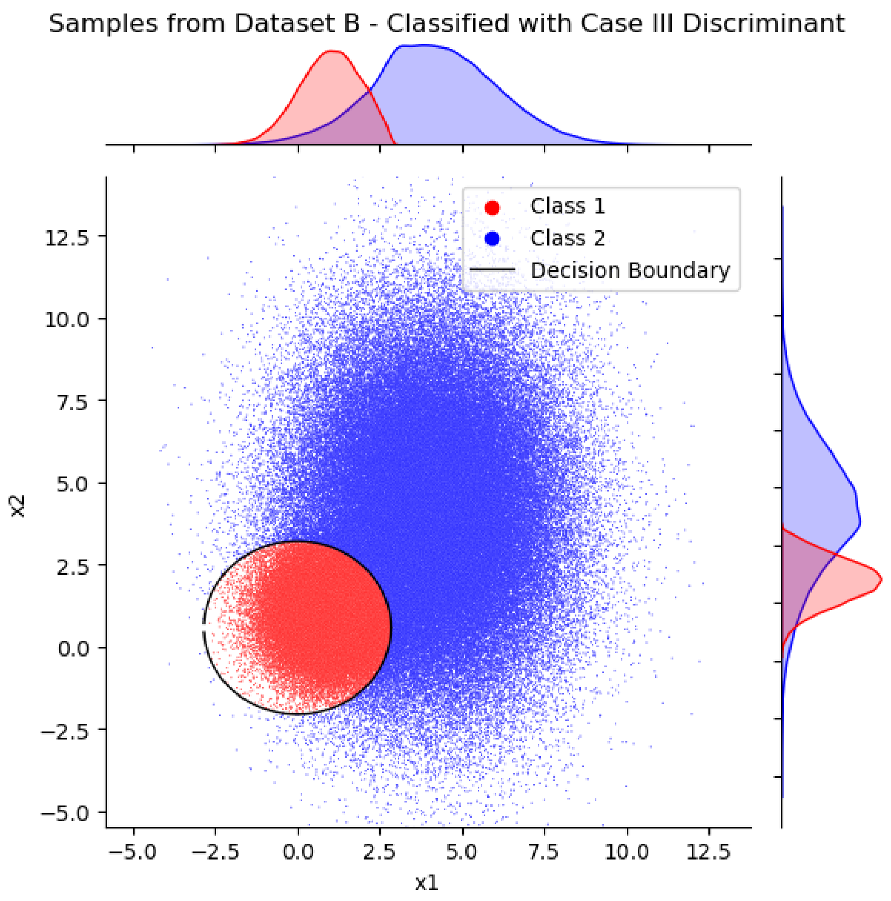


Figure 4: Sample set B, classified using a Case III quadratic discriminant with decision boundary $x_2 = \{0.571 \pm 2.631\sqrt{1 - 0.124x_1^2}\}$, $-2.842 < x_1 < 2.842$.

In comparing the results from sample sets A and B, we can see the difference in classification as a result of the shape and parameters for the given data. While sample A was linearly separable, the data in sample B was instead separated by an ellipsoidal boundary. The misclassification rate was higher for sample set B, which makes sense when comparing the sample data in Figures 1 and 3, as there are more overlapping samples in dataset B than there are in dataset A. This experiment demonstrates that the quality of features can affect accuracy of a classification. Even so, the misclassification rate for dataset B was still fairly low considering the amount of data that was overlapped, which demonstrates the ability for a Bayes classifier to minimize error.

Part 3

Although the data in sample A satisfies most of the assumptions needed for the Euclidean distance classifier, the priors of each class are not equal. This means that the Euclidean distance classifier will not minimize error, and instead will yield a higher misclassification rate than reported in Part 1. Using Equation 14, the resulting functions for $g_1(\mathbf{x})$ and $g_2(\mathbf{x})$ were found to be

$$\begin{aligned} g_1(\mathbf{x}) &= -(x_1-1)^2 - (x_2-1)^2 \\ g_2(\mathbf{x}) &= -(x_1-4)^2 - (x_2-4)^2. \end{aligned} \tag{17}$$

Simplifying, and solving for x_2 gives the linear decision boundary $x_2 = 5 - x_1$. Comparing this to the decision boundary found above, $x_2 = 4.718 - x_1$, we can see that the discriminant equation did not change significantly, but the euclidean distance line shifted toward the more likely class (ω_2) rather than away from it. Figure 5 demonstrates that more samples were determined to be in class 1 than in Figure 2 above.

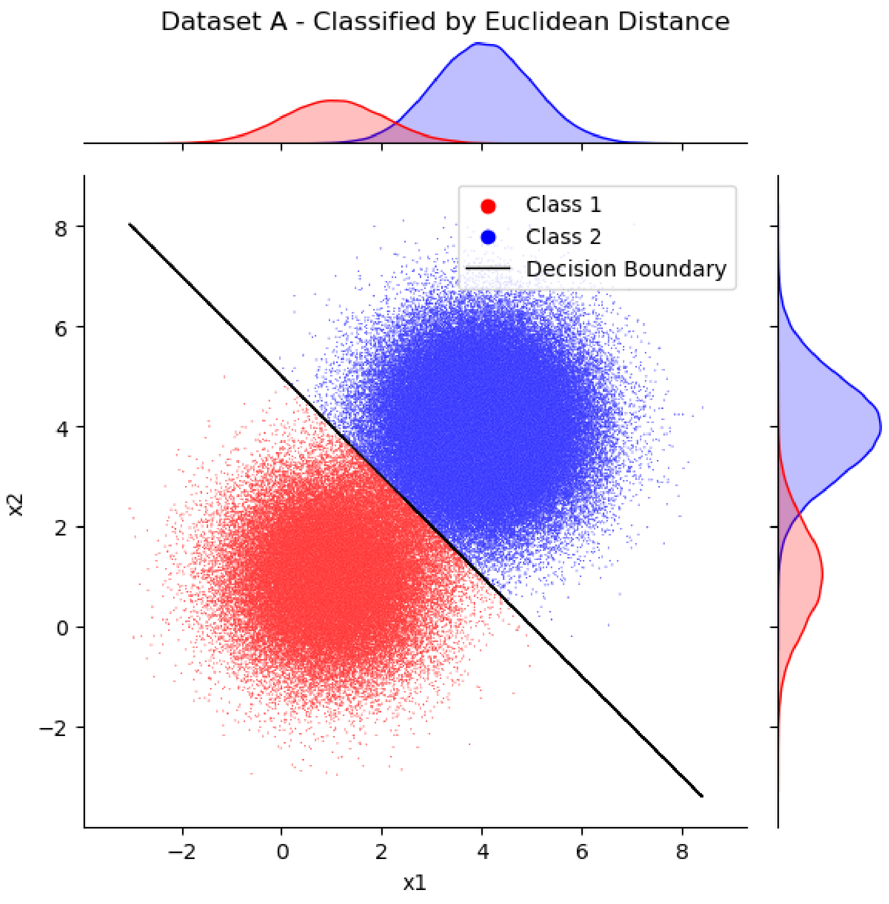


Figure 5: Sample set A, classified using the Euclidean distance classifier (i.e., assuming equal priors).

The misclassification rates for ω_1 and ω_2 using the euclidean distance classifier were 16.32% and 17.21%, respectively, and the total misclassification rate was 16.94%. These error rates are much higher than the approximately 1-2% error rates found above, indicating that using equal priors was not an appropriate assumption for this data set.

Part 4

The data in sample set B does not meet the assumption of equal priors, nor does it meet the assumption of $\Sigma_i = \sigma^2 I$ for all $i \neq j$. Thus, the Euclidean distance classifier will perform even worse for dataset B than it did for dataset A. Figure 6 demonstrates the way that data in sample B was classified by using the Euclidean distance approach. Since μ_1 and μ_2 were the same for samples A and B, the resulting $g_1(\mathbf{x})$ and $g_2(\mathbf{x})$ equations for sample set B were the same as in sample set A. Thus, the decision boundary for sample set B was also $x_2 = 5 - x_1$, as is demonstrated in Figure 6.

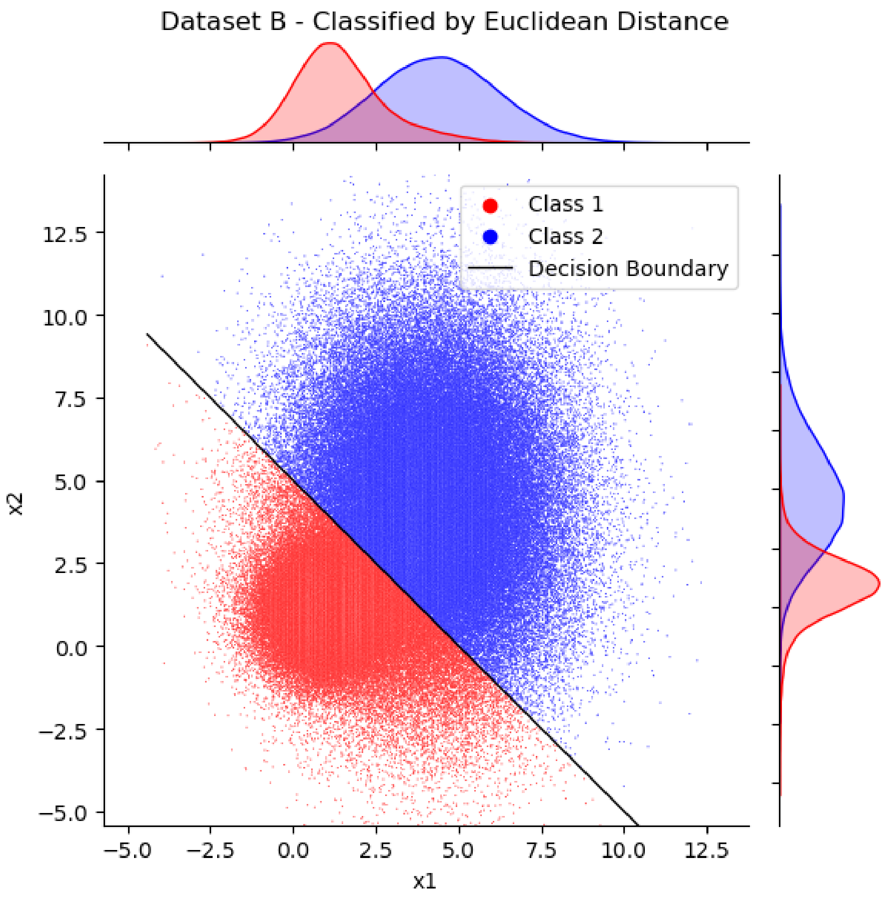


Figure 6: Sample set B, classified using the Euclidean distance classifier and assuming equal priors.

The misclassification rates for ω_1 and ω_2 in sample B were 1.63% and 19.34%, respectfully. Notably, the misclassification rate for ω_1 was much lower than the 8.14% misclassification rate found when using the case III discriminant. The misclassification rate for ω_2, however, more than doubled from 7.29% to 19.34%. The overall misclassification rate for sample set B when using the Euclidean distance classifier was 14.03%, which is close to double the error rate reported in Part 2. These results demonstrate that the Euclidean distance classifier is only optimum when all assumptions regarding the underlying data are met (i.e, data is Gaussian, $\Sigma_i = \sigma^2 I$, and priors are equal). Otherwise, it is best to chose a more complex model to appropriately classify the data when using Bayesian decision theory.

References

Duda, R. O., Hart, P. E., et al. (2006). *Pattern classification.* John Wiley & Sons.

Appendix

Code

Instructions to run code and generate figures:

All code used to generate the data and figures in this assignment is included in separate files which consist of the following:

'BayesianClassifier.py': Python script with all functions used for this assignment. The functions are separated and commented on to indicate the part to which they belong.

'CS679_ASSIGNMENT1_JaleesaHoule.ipynb': The actual data generation and analysis was performed in a Jupyter Notebook environment. A pdf of this notebook is also included in the pages below.

To run this code, download both the python script and the Jupyter Notebook file into the same directory. Open the Jupyter Notebook and select 'run all'. This should re-run all of the analyses conducted for this assignment. The code requires installation of packages Numpy, Pandas, Sympy, Matplotlib, and Seaborn. This code was run using Python 3.8.15.

In [1]:
```
import numpy as np
import pandas as pd
import sympy as sym
from sympy.calculus.util import continuous_domain
x1,x2 = sym.symbols('x1, x2')

import matplotlib.pyplot as plt
import seaborn as sns

#my own script
import BayesianClassifier as BC
```

In [24]:
```
!python -V
```

```
Python 3.8.15
```

Data Generation A

Using the parameters given, generate 60,000 random samples from N(μ1,Σ1) and 140,000 samples from N(μ2,Σ2) (i.e., 200,000 samples total). We will be referring to this data set as “data set A”.

In [2]:
```
mu1 = np.array([1,1])
cov1 = np.array([[1,0],[0,1]])
mu2 = np.array([4,4])
cov2 = np.array([[1,0],[0,1]])

mu= [mu1,mu2 ]
cov =[cov1, cov2]
n_samples= [60000,140000]
```

In [3]:
```
datasetA = BC.SampleData(mu,cov, n_samples) #initialize the object
```

In [4]:
```
datasetA.generate_samples(print_sample_params=True) #generate the samples
```

```
 *******************************************************************

Params of Randomly Generated Samples:


 Sample Means:

Sample 1 : [0.99473353 0.99933409]
Sample 2 : [3.99899976 3.99518989]

 Sample Covariances:

Sample 1 : [[ 0.99286331 -0.00905721]
 [-0.00905721  1.00033076]]
Sample 2 : [[ 0.99646347 -0.00154649]
 [-0.00154649  0.99717404]]

 *******************************************************************
```

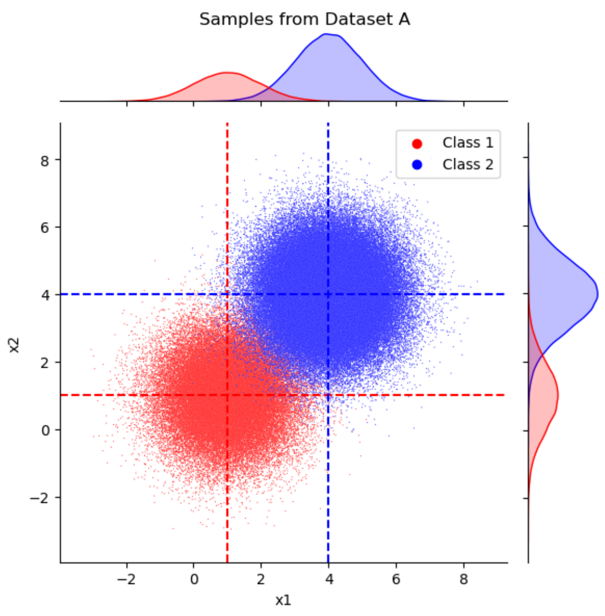
In [5]:
```
datasetA.convert_to_pandas_df() #make a df with the samples
datasetA.df
```

Out[5]:

	x1	x2	true_class
0	1.345584	1.821618	1
1	1.330437	-0.303157	1
2	1.905356	1.446375	1
3	0.463047	1.581118	1
4	1.364572	1.294132	1
...	...	...	...
139995	2.695839	4.471407	2
139996	1.631536	3.064225	2
139997	4.123511	3.249440	2
139998	3.401659	3.957829	2
139999	2.436539	5.826386	2

200000 rows × 3 columns

In [6]:
```
datasetA.make_plot(plot_type='seaborn', sort_type='raw', title='Samples from
```



Part 1

(a) Design a Bayes classifier for minimum error to classify the samples from set A. Which discriminant (i.e., case I, II, or III) would be optimum in this case and why? How would you set the prior probabilities P(ω1) and P(ω2)?

In [7]:
```
datasetA.determine_case_and_params(print_params=True) # chooses if mu,cov fa
datasetA.get_g_equations(print_params=True) # converts estimated params intc
datasetA.classify_2d_sample(print_params=True)
```

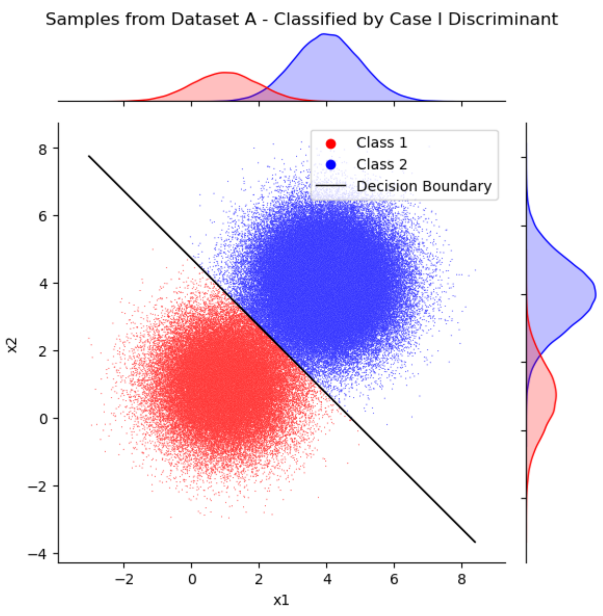
```
 *******************************************************************


 Case number: 1
 Classifier Params:
 [[[array([[1.],
       [1.]]), array([[4.],
       [4.]])]], [[array([[-2.2039728]]), array([[-16.35667494]])]]]

 *******************************************************************


 *******************************************************************


 Discriminant equations:

 g1, g2 =
 [[array([[1.0*x1 + 1.0*x2 - 2.20397280432594]], dtype=object), array([[4.0*
x1 + 4.0*x2 - 16.3566749439387]], dtype=object)]]

 *******************************************************************


 *******************************************************************


 x2=
 {4.71756737987093 - 1.0*x1}

 domain: Reals

 *******************************************************************
```

In [8]:
```
datasetA.x2_symbolic
```

Out[8]: $\{4.71756737987093 - 1.0 x_1\}$

(b) Plot both the Bayes decision boundary and the samples from data set A on the same plot to better visualize how the Bayes rule would classify the data in this case.

In [9]:
```
datasetA.make_plot(plot_type='seaborn', sort_type='sorted', title='Samples f
```



(c) Next, classify all 200,000 samples and report:

(i) the misclassification rate for each class separately (i.e., the percentage of misclassified samples for each class)

(ii) the total misclassification rate (i.e., the percentage of misclassified samples overall).

```
In [10]: datasetA.get_true_error(print_params=True)
```

```
 ******************************************************************



 Empirical Error Stats:

N samples misclassified: [1603, 1414]
Misclassification rate by class: [0.02671667 0.0101    ]
Total misclassification rate: 0.015085

 ******************************************************************
```

(d) Calculate the theoretical probability error (e.g., Bhattacharyya bound) and compare it with the misclassification rate from part (c). What do you observe?

```
In [11]: datasetA.Bhattacharyya_error_bound(print_params=True)
```

```
 ******************************************************************


 Bhattacharyya Error Bound Stats:

k value: 2.25
Probability of error <= 0.048299999247443921

 ******************************************************************
```

Data Generation B

Using the parameters given, generate 60,000 random samples from N(μ1,Σ1) and 140,000 samples from N(μ2,Σ2) (i.e., 200,000 samples total). We will be referring to this data set as “data set B”.

```
In [12]: mu1 = np.array([1,1])
         cov1 = np.array([[1,0],[0,1]])
         mu2 = np.array([4,4])
         cov2 = np.array([[4,0],[0,8]])

         mu= [mu1,mu2 ]
         cov =[cov1, cov2]
```

```
In [13]: datasetB = BC.SampleData(mu,cov, n_samples) #initialize the object
         datasetB.generate_samples(print_sample_params=True) #generate the samples
```

```
 ******************************************************************

Params of Randomly Generated Samples:


 Sample Means:

Sample 1 : [0.99473353 0.99933409]
Sample 2 : [3.99037978 3.99717091]

 Sample Covariances:

Sample 1 : [[ 0.99286331 -0.00905721]
 [-0.00905721  1.00033076]]
Sample 2 : [[ 3.98869615 -0.00874825]
 [-0.00874825  7.97170778]]

 ******************************************************************
```

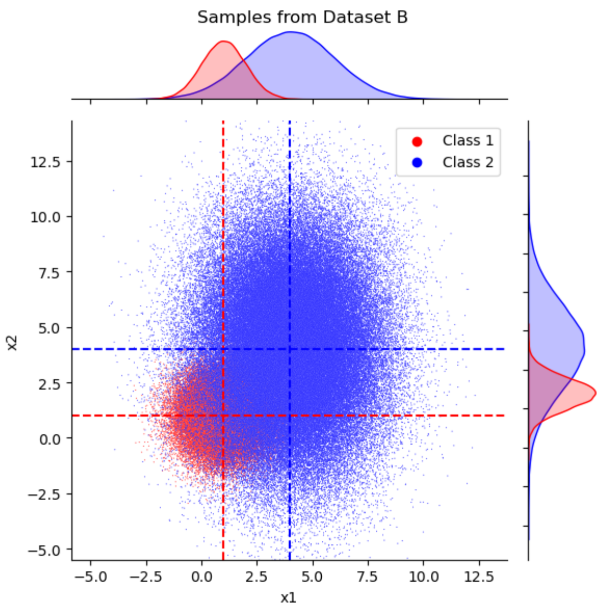
In [14]:
```
datasetB.convert_to_pandas_df() #make a df with the samples
datasetB.df
```

Out[14]:

	x1	x2	true_class
0	1.345584	1.821618	1
1	1.330437	-0.303157	1
2	1.905356	1.446375	1
3	0.463047	1.581118	1
4	1.364572	1.294132	1
...	...	...	...
139995	4.942814	0.311276	2
139996	2.128450	-2.699029	2
139997	2.498879	4.349343	2
139998	3.915658	2.307635	2
139999	7.652772	-0.422135	2

200000 rows × 3 columns

In [15]:
```
datasetB.make_plot(plot_type='seaborn', sort_type='raw', title='Samples from
```



Part 2

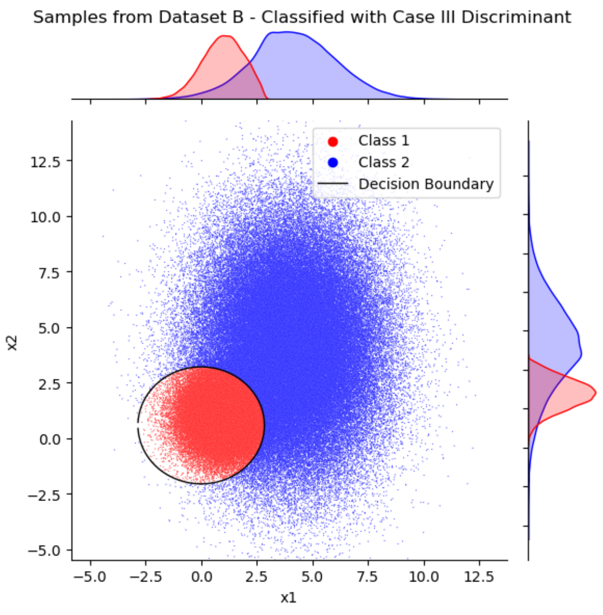
Repeat experiment 1 using data set B. How do your results from this experiment compare with your results from experiment 1 and why?

In [16]:
```
datasetB.determine_case_and_params(print_params=True) # chooses if mu,cov fa
datasetB.get_g_equations(print_params=True) # converts estimated params into
datasetB.classify_2d_sample(print_params=True)
```

```
 *****************************************************************


 Case number: 3
 Classifier Params:
 [[[array([[-0.5, -0. ],
       [-0. , -0.5]]), array([[-0.125 , -0.    ],
       [-0.    , -0.0625]])]], [[array([[1.],
       [1.]]), array([[1. ],
       [0.5]])]], [[array([[-2.2039728]]), array([[-5.0895429]])]]]

 *****************************************************************


 *****************************************************************


 Discriminant equations:

 g1, g2 =
 [[array([[-0.5*x1**2 + 1.0*x1 - 0.5*x2**2 + 1.0*x2 - 2.20397280432594]],
      dtype=object), array([[-0.125*x1**2 + 1.0*x1 - 0.0625*x2**2 + 0.5*x2 -
5.0895428953386]],
      dtype=object)]]

 *****************************************************************


 *****************************************************************


 x2=
 {0.571428571428571 - 2.63099209267196*sqrt(1.0 - 0.123826650284351*x1**2),
2.63099209267196*sqrt(1.0 - 0.123826650284351*x1**2) + 0.571428571428571}

 domain: Interval(-2.84179625536141, 2.84179625536141)

 *****************************************************************
```

In [17]:
```
datasetB.make_plot(plot_type='seaborn', sort_type='sorted', title='Samples f
```

```
/Volumes/MacBackup/opt/anaconda3/lib/python3.8/site-packages/pandas/core/arr
aylike.py:396: RuntimeWarning: invalid value encountered in sqrt
  result = getattr(ufunc, method)(*inputs, **kwargs)
```



In [18]:
```
datasetB.get_true_error(print_params=True)
```

```
 *****************************************************************


 Empirical Error Stats:

N samples misclassified: [4885, 10206]
Misclassification rate by class: [0.08141667 0.0729    ]
Total misclassification rate: 0.075455

 *****************************************************************
```

In [19]:
```
datasetB.Bhattacharyya_error_bound(print_params=True)
```

```
 *******************************************************************


 Bhattacharyya Error Bound Stats:

k value: 1.0437500886252833
Probability of error <= 0.16136700681940294

 *******************************************************************
```

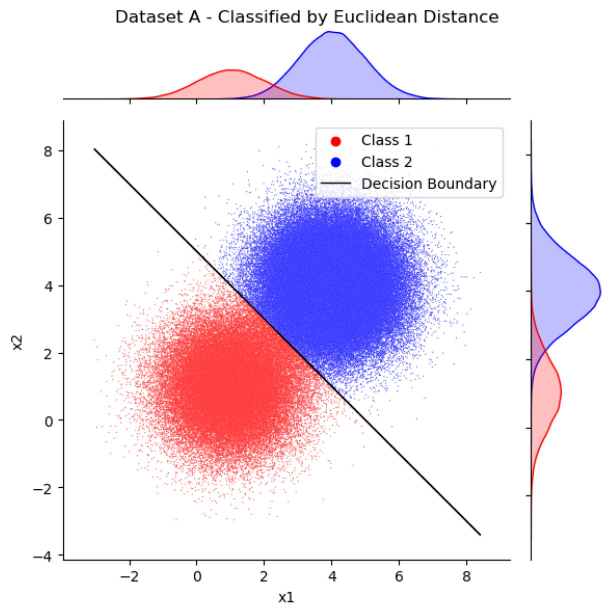
Part 3

Classify the samples from data set A using the Euclidean distance classifier and compare your results (i.e., misclassification rates) with those obtained from experiment 1. Explain your findings.

In [20]:
```
datasetA.classify_by_euclidean_distance()
datasetA.get_euclidean_error(print_params=True)
```

```
 *******************************************************************


 Discriminant equations:

g1, g2 =  [[array([[(1 - x1)*(x1 - 1) + (1 - x2)*(x2 - 1)]], dtype=object),
array([[(4 - x1)*(x1 - 4) + (4 - x2)*(x2 - 4)]], dtype=object)]]

 x2=
 {5 - x1}

 *******************************************************************


 Euclidean Distance Error Stats:

N samples misclassified: [979, 2410]
Probability of misclassification by class: [0.01631667 0.01721429]
Total error probability: 0.016945

 *******************************************************************
```

In [21]:
```
datasetA.make_plot(plot_type='seaborn', sort_type='euclidean', title='Datase
```



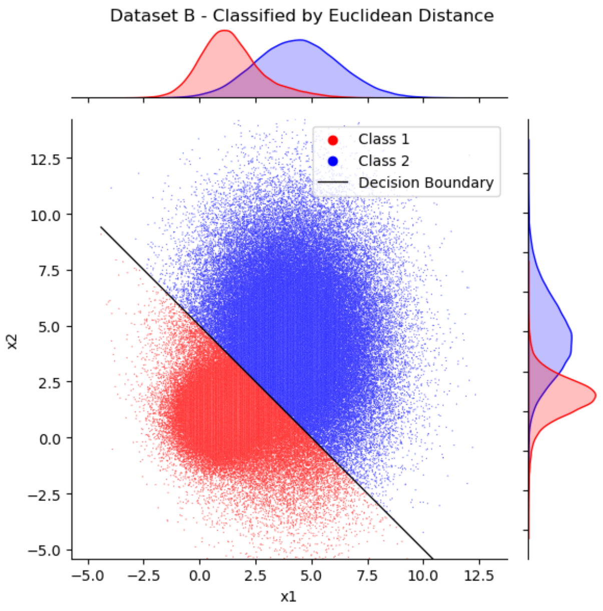
Part 4

Repeat experiment 3 using the samples from data set B. Compare and discuss your results with those obtained from experiments 2 and 3. Explain your findings.

In [22]:
```
datasetB.classify_by_euclidean_distance()
datasetB.get_euclidean_error(print_params=True)
```

```
 *****************************************************************


 Discriminant equations:

g1, g2 =  [[array([[(1 - x1)*(x1 - 1) + (1 - x2)*(x2 - 1)]], dtype=object),
array([[(4 - x1)*(x1 - 4) + (4 - x2)*(x2 - 4)]], dtype=object)]]

 x2=
 {5 - x1}

 *****************************************************************


 Euclidean Distance Error Stats:

N samples misclassified: [979, 27073]
Probability of misclassification by class: [0.01631667 0.19337857]
Total error probability: 0.14026

 *****************************************************************
```

```
In [23]: datasetB.make_plot(plot_type='seaborn', sort_type='euclidean', title='Datase
```



```
In [ ]:
```