CSE 802 Homework 3

By Abhiram Durgaraju

Problem 1)

a)

The likelihood function is the product for each in the above distribution:

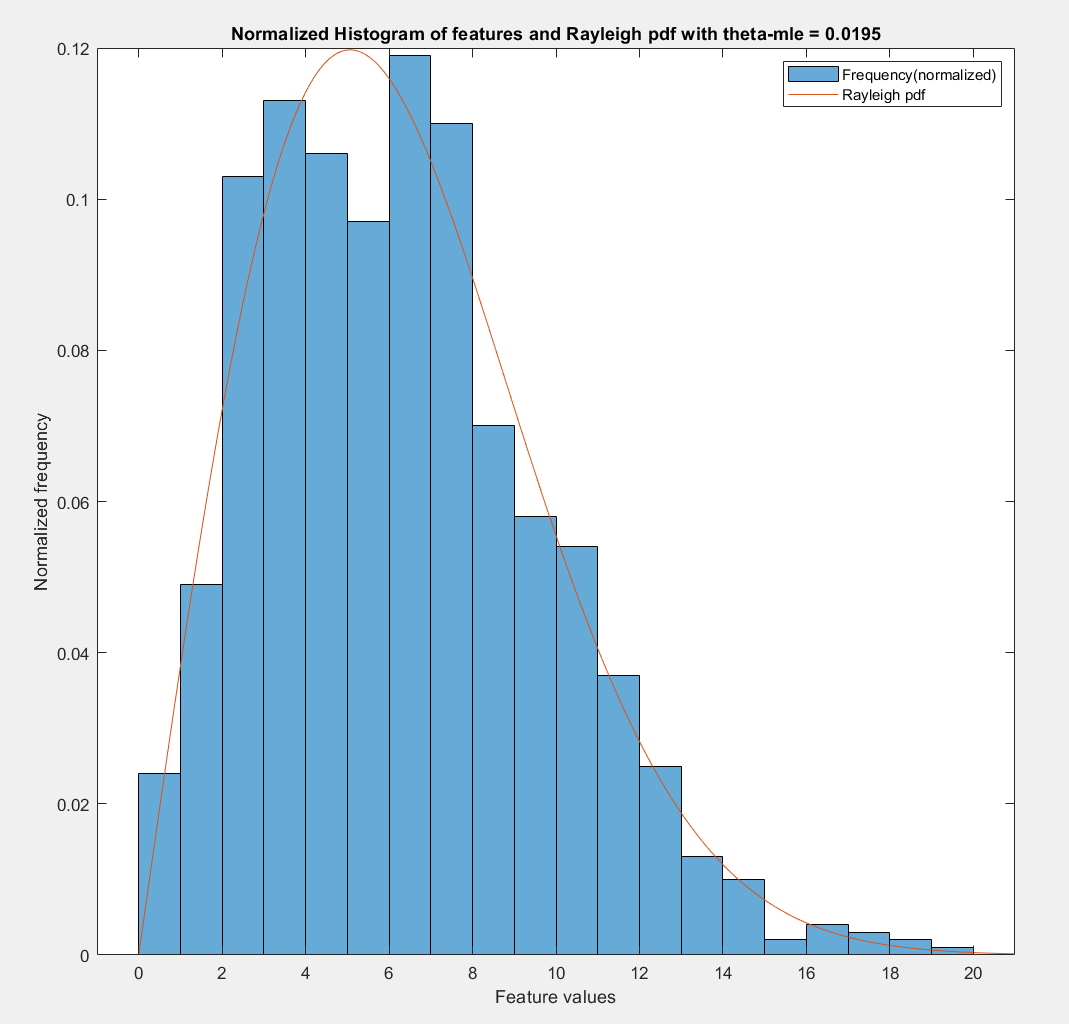
Convert to log likelihood to simplify the equation:

The logarithm converts the product function to a sum:

The maximum likelihood can be found by taking the derivative with respect to and setting it equal to zero:

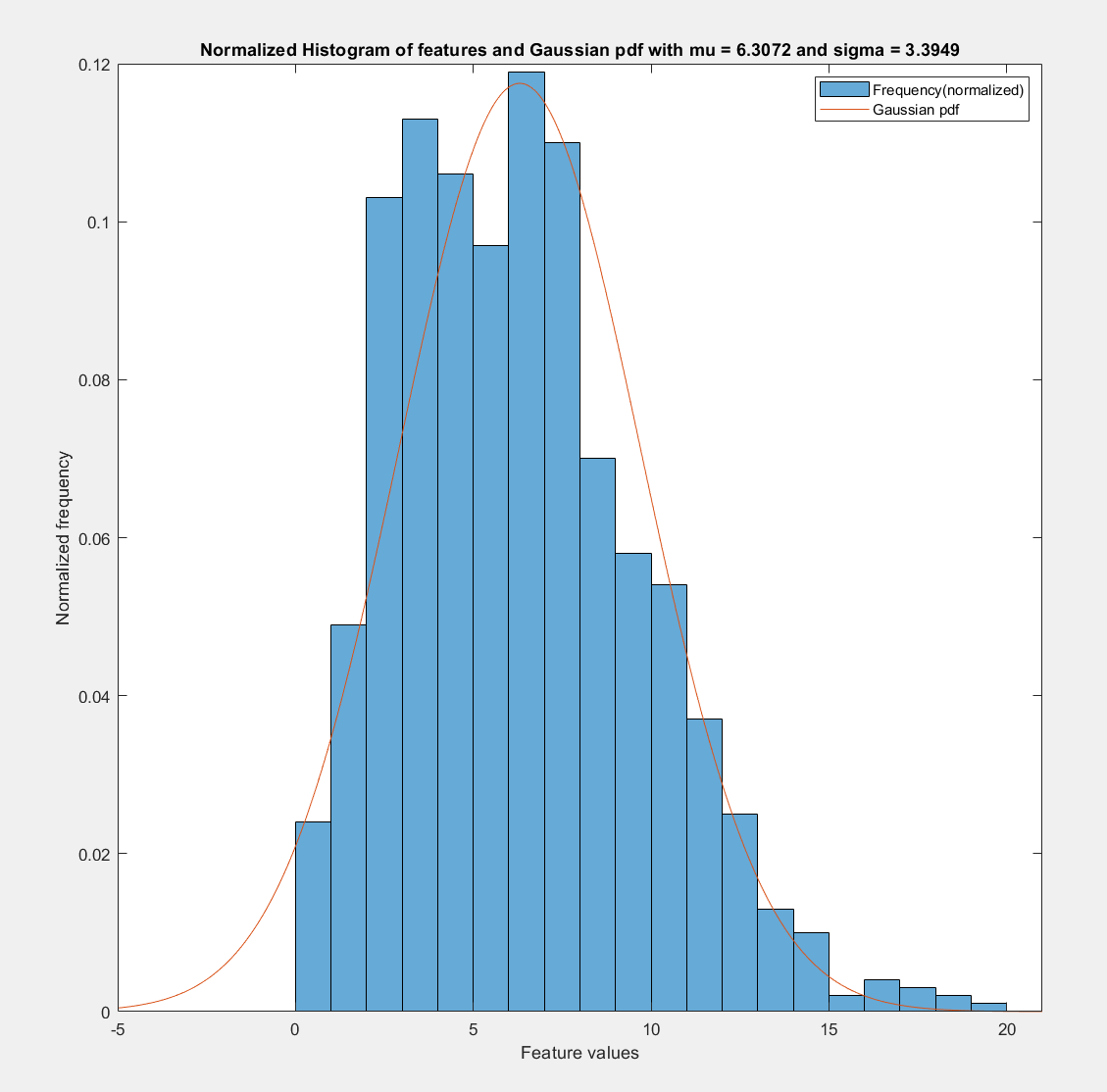
Since we are taking a derivative with respect to, can be treated as a constant, and therefore ignored.

b)



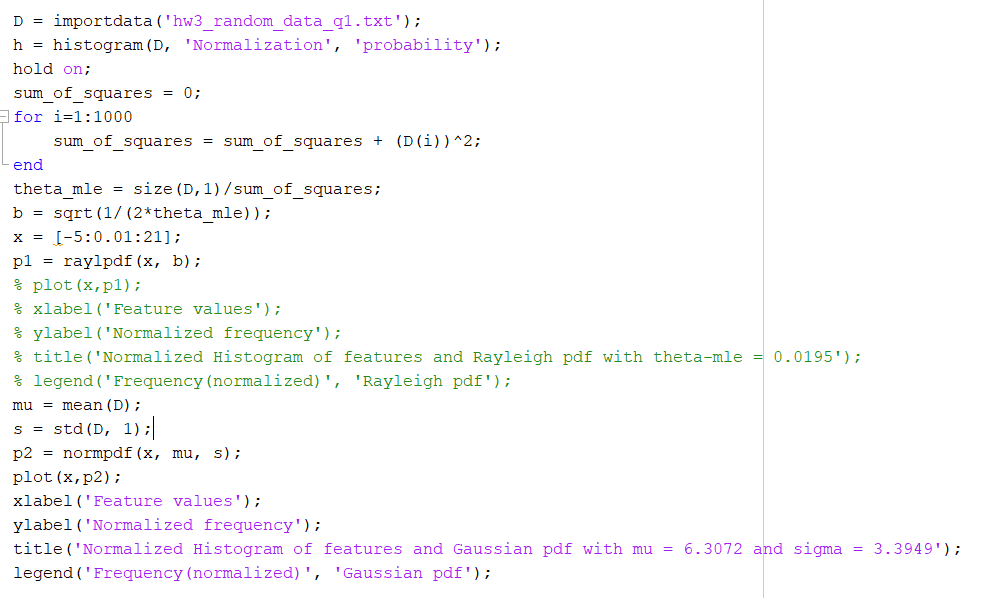
Using the derived formula, I found that

c)



The Gaussian pdf was plotted using a mean of 6.3072 and sigma of 3.3949. The MLE formula for mean is the same as MATLAB mean. The MLE formula for standard deviation is biased, and is calculated using std with weight of 1 in MATLAB.

d) The Rayleigh distribution fits the data much better. Since the majority of patterns lie within a short range of feature values near the beginning, it can be said that the distribution is skewed to the left. Therefore, the Gaussian distribution is not an ideal fit for this training data.



Problem 2)

a)

The likelihood function is the product of each where

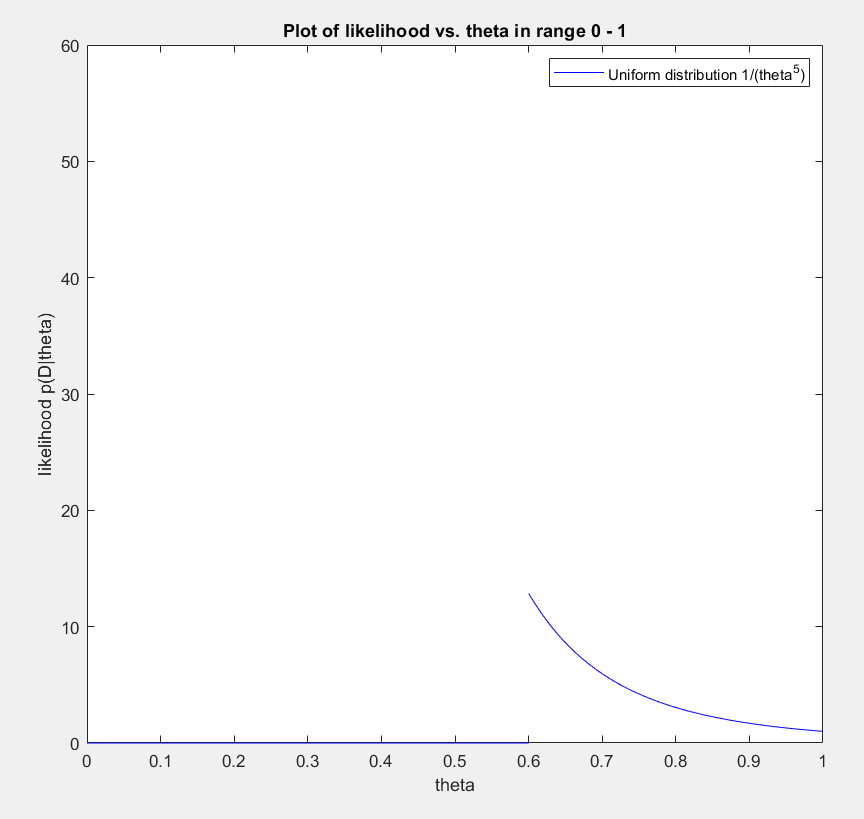
The log likelihood is given by:

is a decreasing function, as shown by the derivative. Therefore, it is maximized when is minimized. But we are given the constraint that for all . The smallest value of under the constraint that is simply the

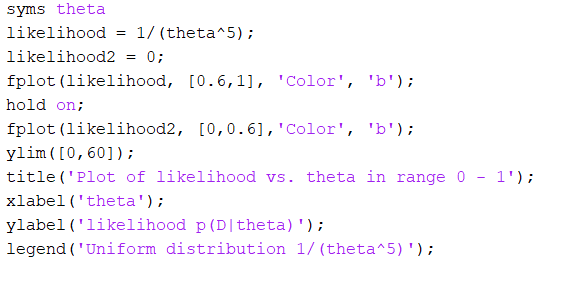
Therefore

b)

Since we are given that we know that:



As seen in the plot, the likelihood is maximized when is as small as possible. Given our constraint that , for all , the value of is maximum at . Therefore, we do not need to know the values of the other 4 points, as is purely dictated by the maximum value in the dataset, which is 0.6. When , . Therefore, we must pick



Problem 3)

Let us consider deriving , a specific component of MLE of . For , the Bernoulli distribution is as follows:

As before, let us take the log-likelihood as follows:

To maximize the log-likelihood, we need to take the derivative with respect to and set equal to zero:

As shown above, it clear that the parameter is simply the sample mean of. We can simply extend this principle to training samples to obtain, which is the sample mean of each pattern. So instead of dimensions of , we have training samples of for all :

Problem 4)

For 50 random samples:

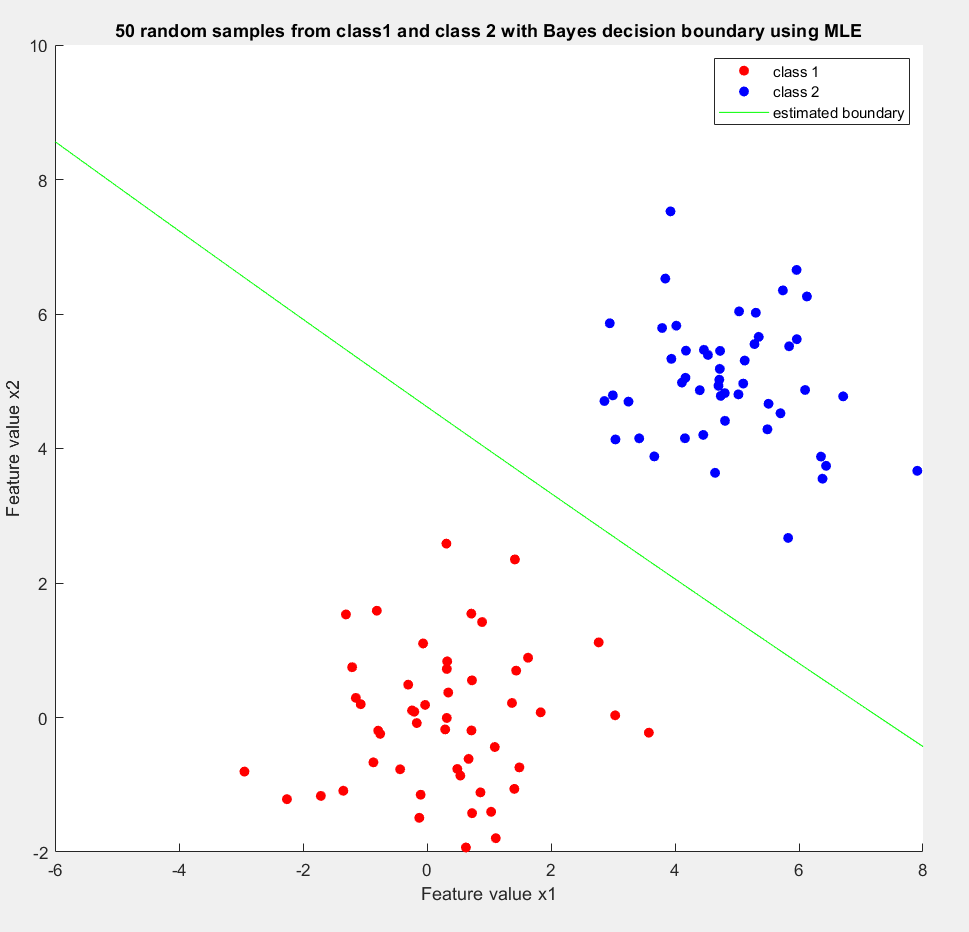
1. The maximum likelihood estimates, using equations (18) and (19) in textbook, via MATLAB are:
2. The Bayes decision boundary can be calculated when the posterior probability of the two classes are equal:

Using Bayes formula:

Since the priors are equal and is on both sides, the expression can be simplified to:

I used MATLAB to equate the two probability densities and solve for the line of intersection given 50 random samples. The Bayes decision boundary using estimated parameters is:

x2 = 24.325 - 3.4076\*(- 0.0011426\*x1^2 - 0.80902\*x1 + 33.44)^(1/2) - 0.88568\*x1

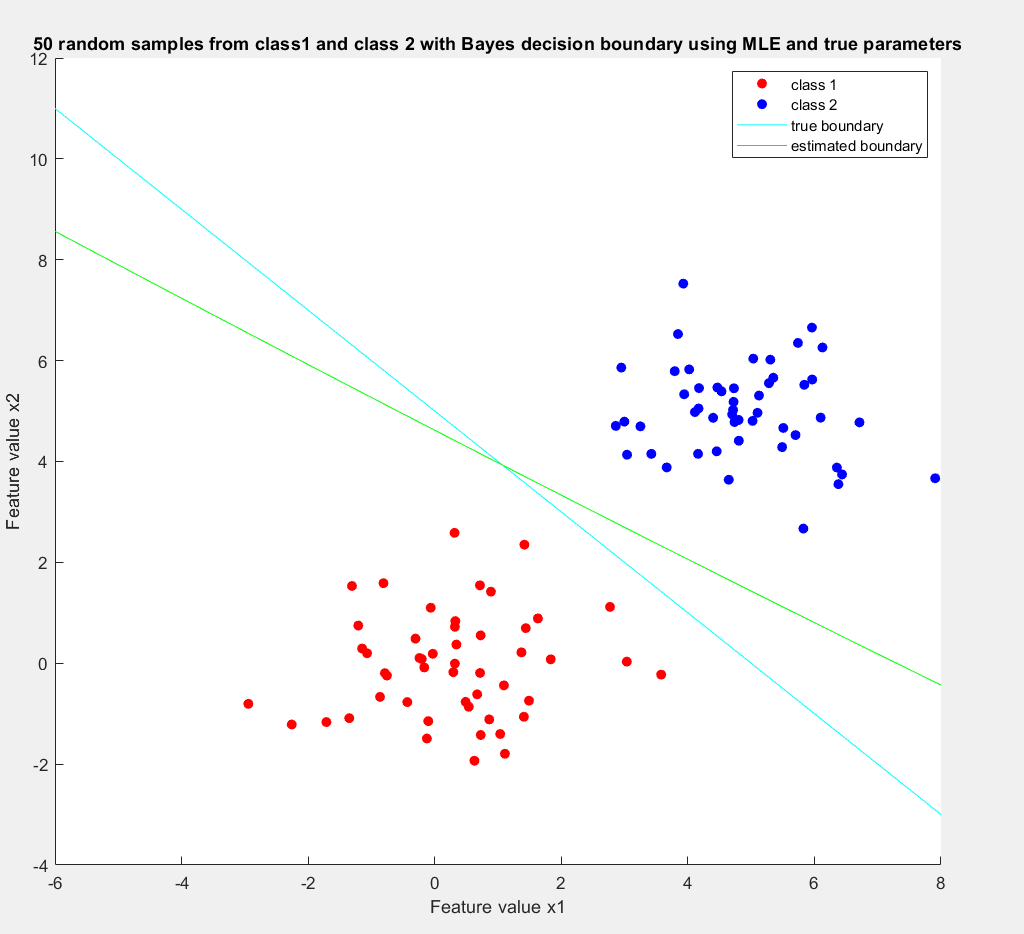


No samples were misclassified using estimated parameters. The empirical error is 0/100 = 0%

1. As in the previous part, since the priors are equal, we can say that the decision boundary can be found when:

Using the true values of mean and covariance, we have the following:

Take the log of both side and cancel all like terms:



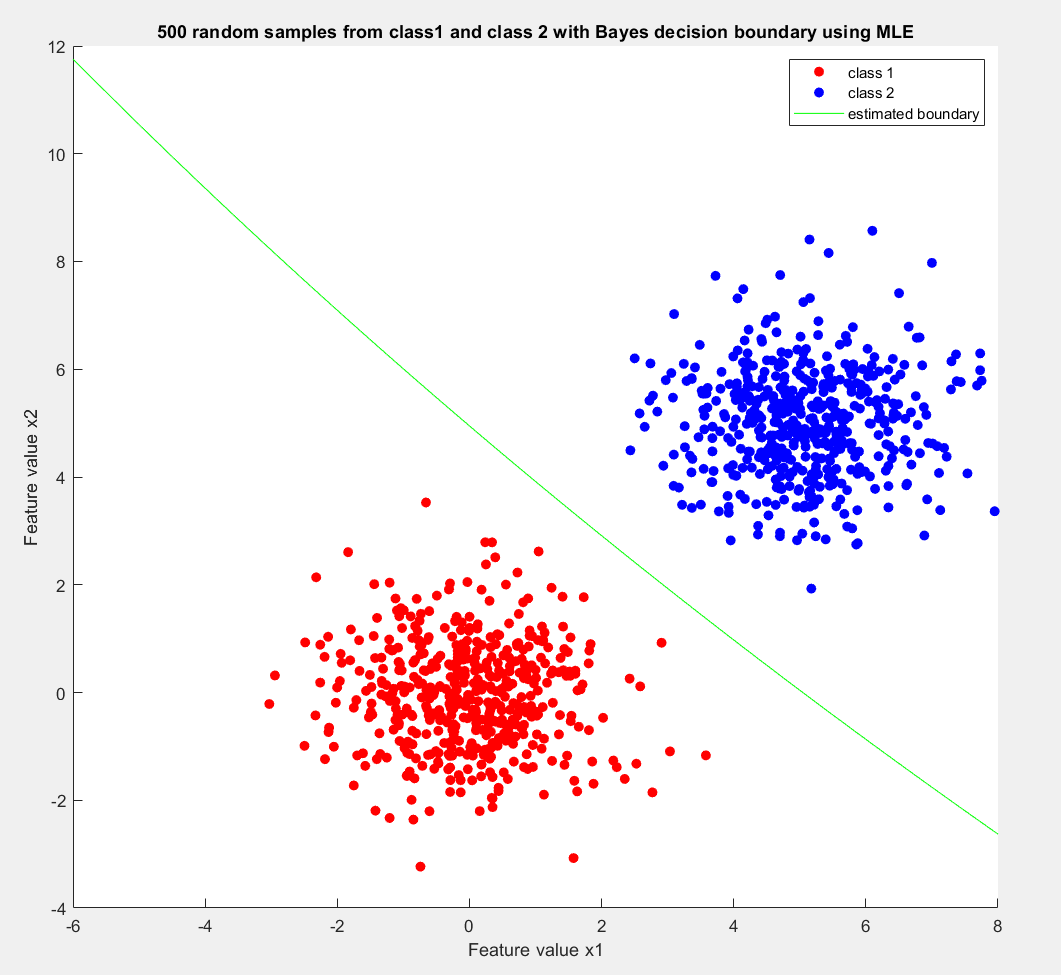
No samples were misclassified using the true parameters. The empirical error is 0/100 = 0%

For 500 random samples:

1. The maximum likelihood estimates, using equations (18) and (19) in textbook, via MATLAB are:

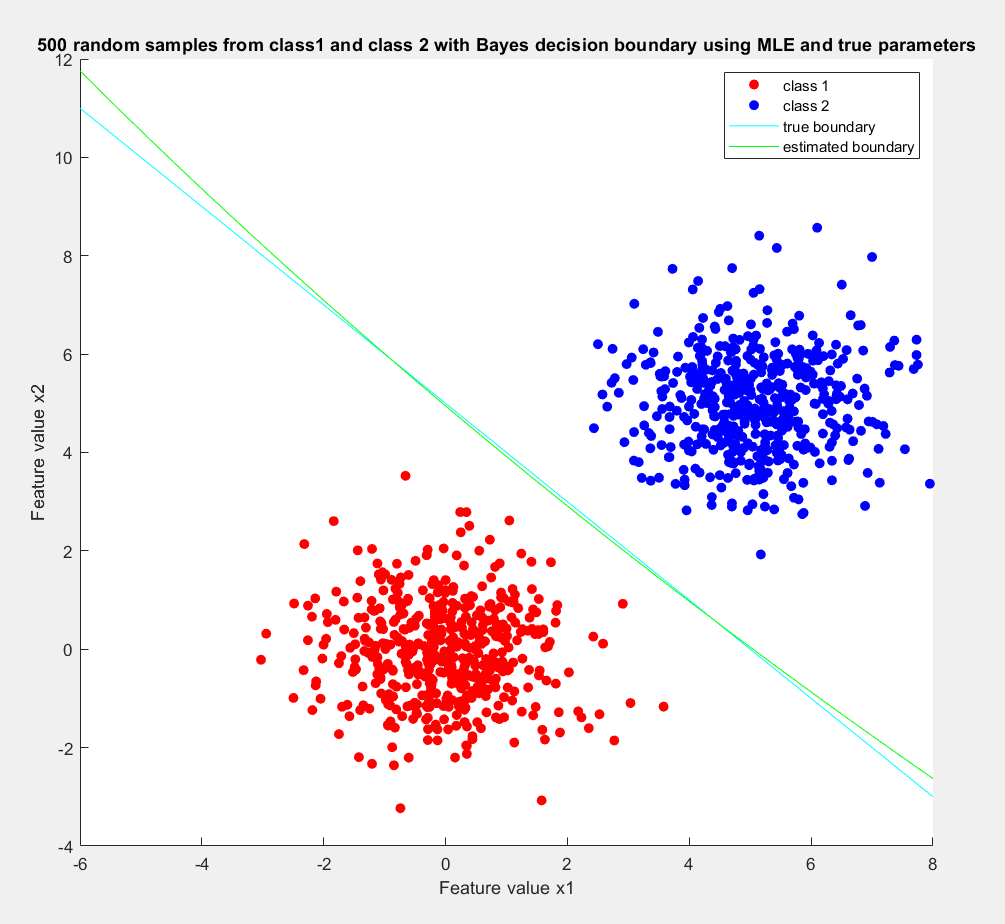
1. Using MATLAB,

x2 = 11.994\*x1 - 175.49\*(0.0047273\*x1^2 + 0.75767\*x1 + 25.999)^(1/2) + 899.75



The empirical error using estimated parameters is 0/1000 = 0%

The true boundary does not change based on sample size. It is still:

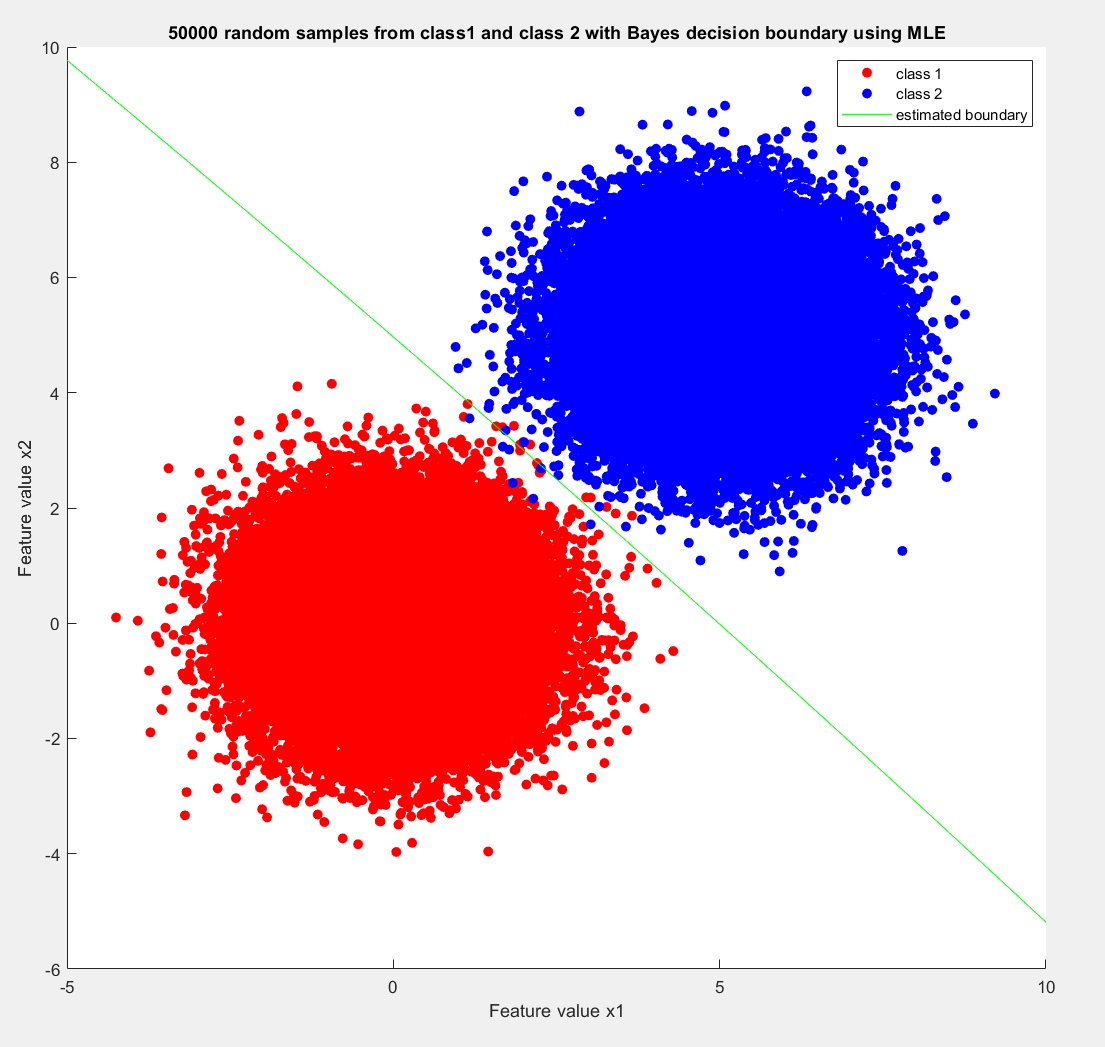


The empirical error rate using true parameters is 0/1000 = 0%

For 50000 random samples:

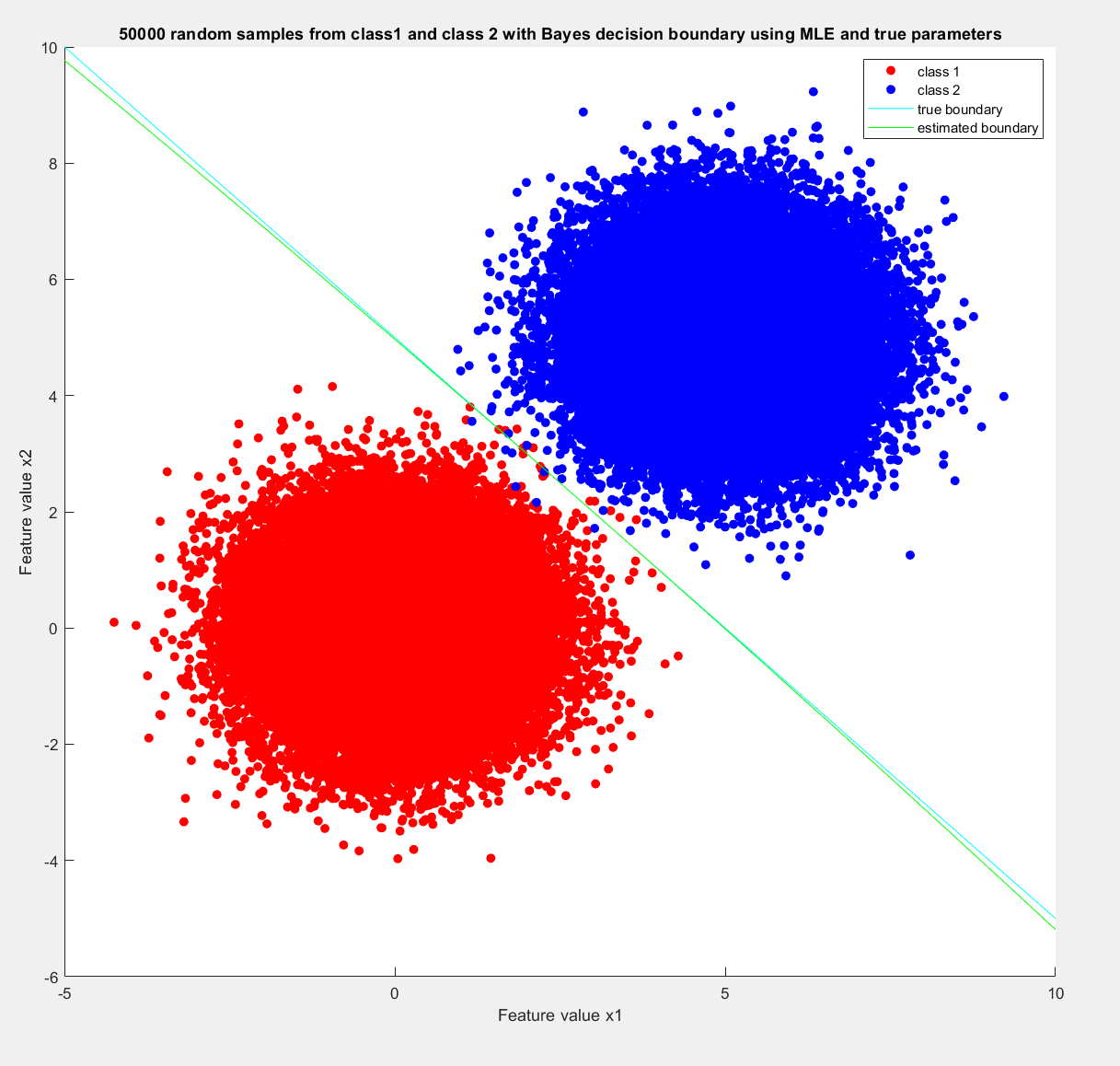
1. Using MATLAB, the Bayes decision boundary using estimated parameters is:

x2 = 4.4734\*x1 + 498.6\*(0.000045131\*x1^2 - 0.11062\*x1 + 25.588)^(1/2) - 2517.2



7 samples from class 2 were misclassified as class 1. 10 samples from class 1 were misclassified as class 2. A total of 17 samples were misclassified. The empirical error using estimated parameters is 17/100000 = 0.017%

1. The true boundary does not change based on sample size. It is still:



Misclassification using true parameters was the same as estimated parameters. 7 samples from class 2 were misclassified as class 1. 10 samples from class 1 were misclassified as class 2. A total of 17 samples were misclassified. The empirical error using estimated parameters is 17/100000 = 0.017%

1. As the number of representative training samples increases, the maximum likelihood estimated parameters get closer and closer to the true parameters. With 50000 samples, the difference between the estimated and true parameters is minimal. The true parameters do not change, as they do not depend on sample size.

As the number of representative training samples increases, the empirical error rate goes up. This is true for both the estimated boundary and true boundary. This makes sense intuitively because a larger sample size means more chances for mvrnd to output a pattern that is on the very edge of the Gaussian distribution, thereby being misclassified.



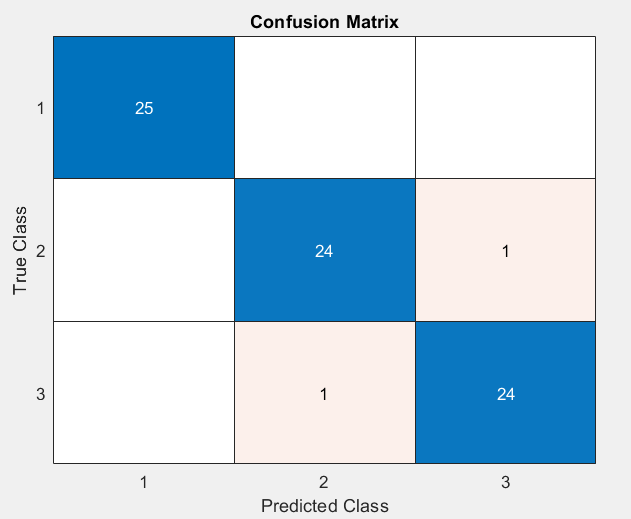
Problem 5)

1. The following were calculated using MATLAB:
2. Since the priors are equal and this is a 0-1 loss function, the maximum posterior rule can be simplified using Bayes formula:

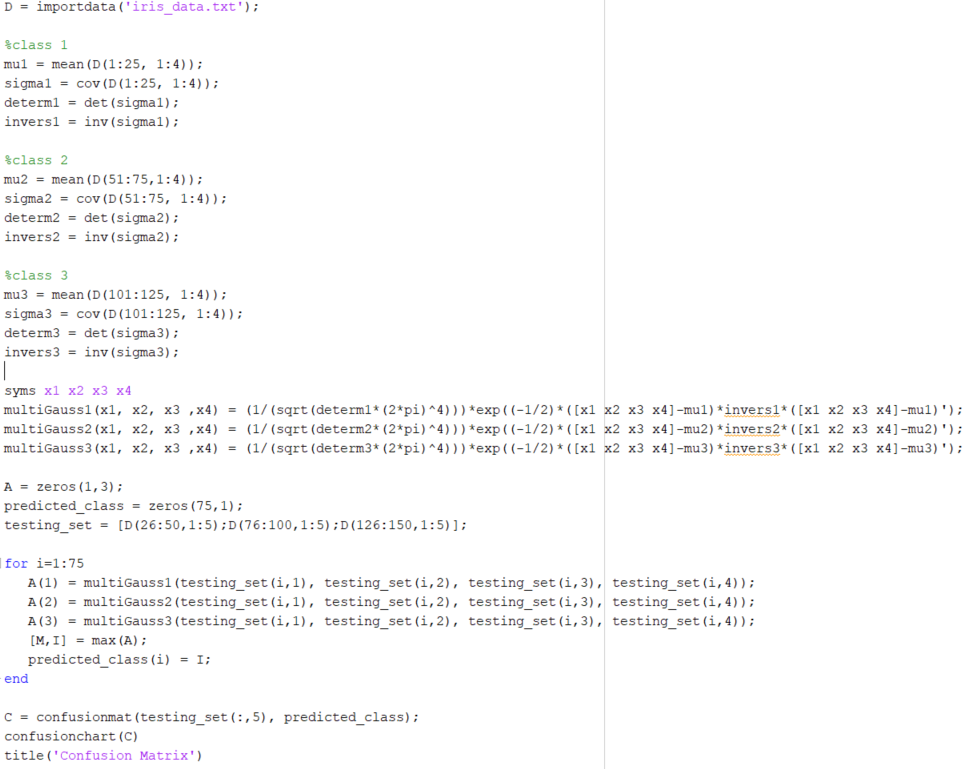
The priors are equal and is common to all three classes. So these can be cancelled out:

The code for my program is at shown at the end of this question.

c)



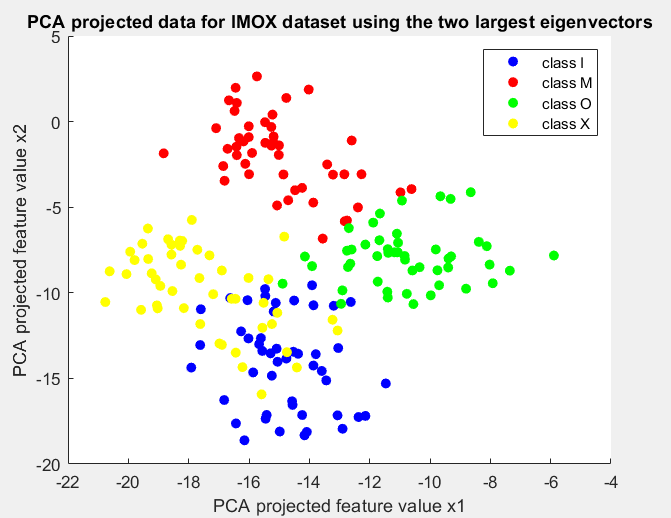
Empirical error rate = 2/75 = 2.666%



Problem 6)

1. Projection matrix using PCA:

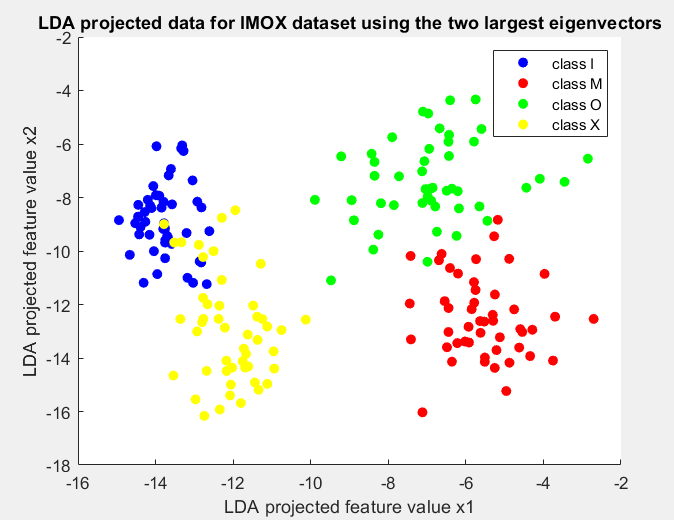
|  |
| --- |
| e1 e2 |
| 0.0145 -0.2153 |
| -0.0883 -0.4018 |
| -0.1640 -0.2741 |
| -0.0504 -0.0051 |
| -0.3403 -0.3764 |
| -0.4758 -0.5049 |
| -0.5572 0.3997 |
| -0.5567 0.4008 |



b)

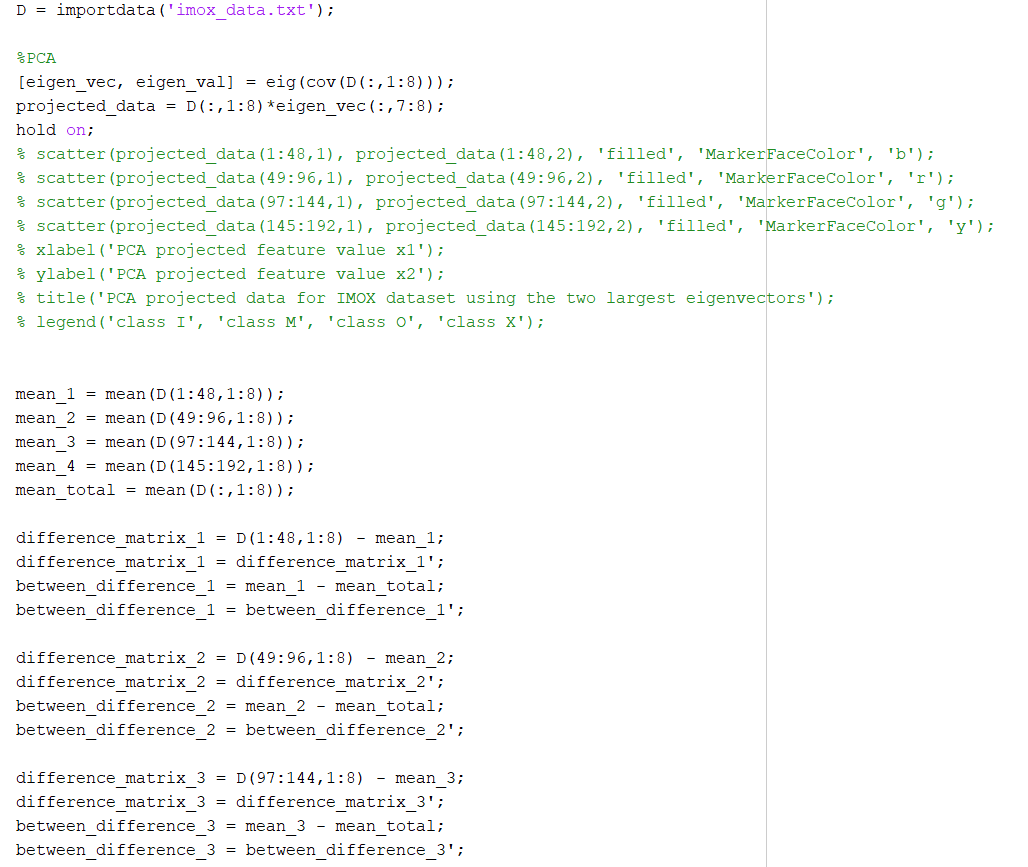
Projection matrix using LDA:

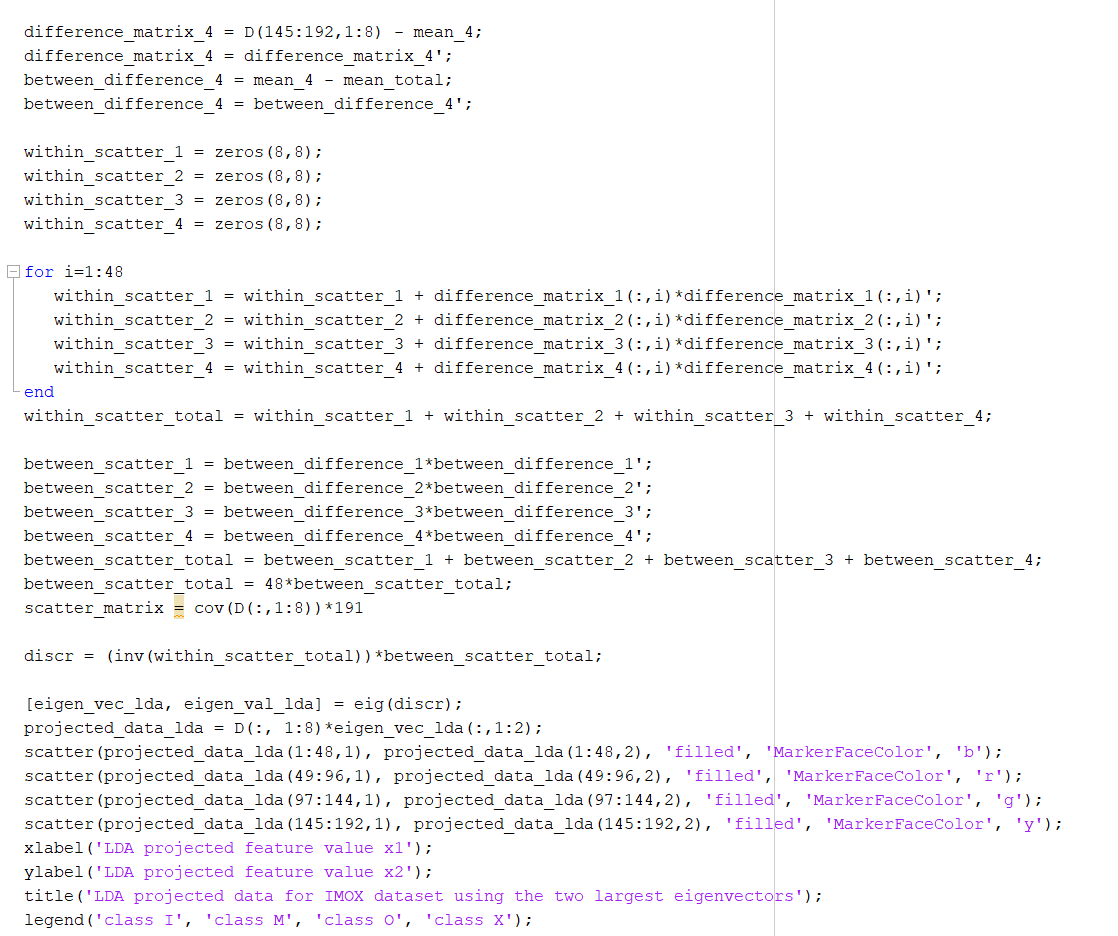
|  |
| --- |
| e1 e2 |
| -0.0354 -0.0237 |
| 0.0736 -0.0067 |
| 0.0452 0.0353 |
| -0.0507 0.0470 |
| -0.5458 -0.5475 |
| -0.8239 -0.0017 |
| -0.0377 -0.4865 |
| 0.1024 -0.6779 |



c)

Since LDA takes into account class means, it better separates the two classes. This is because we minimize the within class scatter matrix and maximize the between class scatter matrix. The projection matrix of LDA tries to maximize the difference between he means relative to the variance. Whereas, the projection matrix of PCA seeks to best represent the data by plotting the two axes of maximum variance.





Problem 7)

a)

For features and classes , their MLE estimates of mean and variances (biased) are shown in table below. They were calculated using MATLAB. Code is posted at the end of the problem.

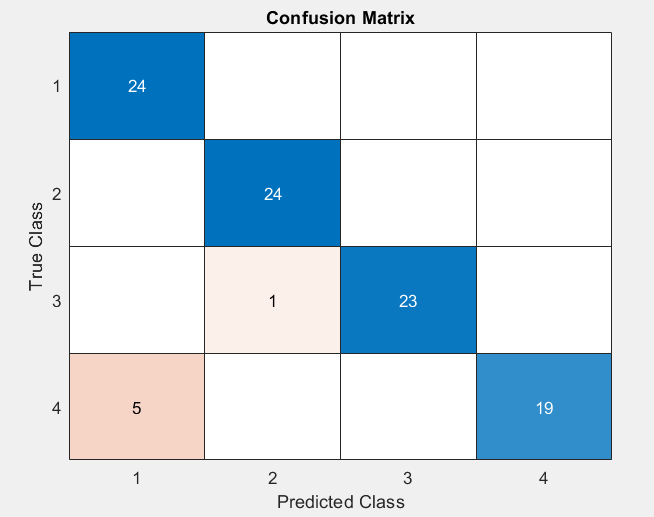
|  |  |  |  |
| --- | --- | --- | --- |
| i | j |  |  |
| 1 | 1 | 7.3333 7.5139 | |
| 2 | 1 | 9.2083 9.2899 | |
| 3 | 1 | 8.1875 12.5273 | |
| 4 | 1 | 5.9375 5.6836 | |
| 5 | 1 | 9.3125 1.7148 | |
| 6 | 1 | 11.4375 1.7878 | |
| 7 | 1 | 3.1458 1.8746 | |
| 8 | 1 | 3.7708 1.8016 | |
| 1 | 2 | 5.6667 1.8056 | |
| 2 | 2 | 5.1250 1.6927 | |
| 3 | 2 | 5.3750 1.8594 | |
| 4 | 2 | 6.0625 10.4753 | |
| 5 | 2 | 4.6458 0.9787 | |
| 6 | 2 | 4.6042 0.9891 | |
| 7 | 2 | 7.8958 12.2600 | |
| 8 | 2 | 9.4375 6.4961 | |
| 1 | 3 | 7.3125 0.7565 | |
| 2 | 3 | 7.2083 1.1233 | |
| 3 | 3 | 6.7292 0.7808 | |
| 4 | 3 | 5.9792 0.8121 | |
| 5 | 3 | 5.3333 1.4306 | |
| 6 | 3 | 5.4792 1.3746 | |
| 7 | 3 | 3.7292 2.0725 | |
| 8 | 3 | 4.2083 1.8733 | |
| 1 | 4 | 8.2708 4.8641 | |
| 2 | 4 | 8.0833 5.9931 | |
| 3 | 4 | 7.2500 5.4792 | |
| 4 | 4 | 6.0000 3.2917 | |
| 5 | 4 | 9.1875 0.9023 | |
| 6 | 4 | 9.6250 0.9010 | |
| 7 | 4 | 6.4167 6.5347 | |
| 8 | 4 | 7.4375 6.0794 | |

b)

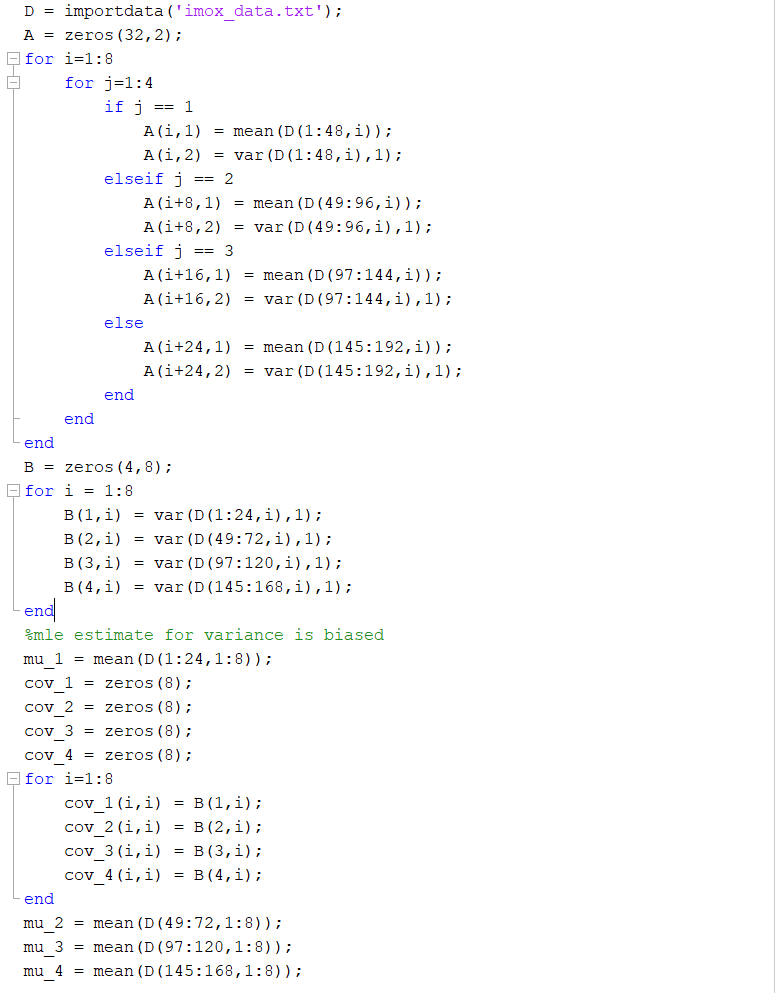
The classifier is the same as in problem #5. We try to maximize the posterior probability for the 4 classes:

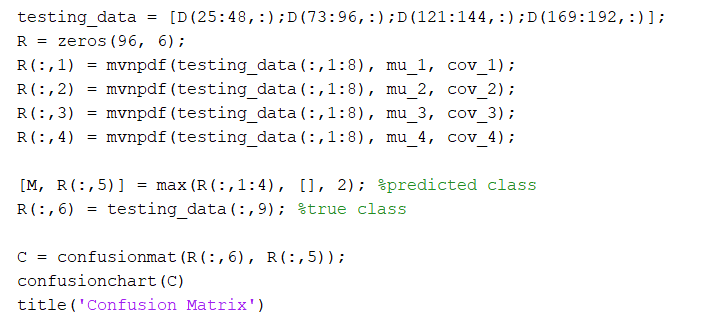
The priors are equal and is common to all three classes. So these can be cancelled out:

c)



The empirical error rate is 6/96 = 6.25%





Problem 8)

The criterion function J(.) will be invoked based on the number of features evaluated inf each iteration of the algorithm.

1. In SFS, we start with a null set and pick the best feature to add to our set and iterate 5 times. 15 features to evaluate to pick the first feature. 14 for the second. 13 for the third. 12 for the fourth. 11 for the second. Add them up for a total is **65 criterion function invokes.**
2. This can be thought of as performing SFS for *l* iterations and performing SBS for *r* iterations. If we do this algorithm twice, we end up with 4 features. Since the question asks to identify a subset of 5 feature or less, we must do another iteration of SFS at the end to reach 5 features.

First SFS:

(15+14+13+12+11) = 65

First SBS:

(5+4+3) = 12

Second SFS:

(13+12+11+10+9) = 55

Second SBS:

(7+6+5) = 18

Last SFS:

11

Total:

65 + 12 +55 + 18 + 11 = **161 criterion function invokes**

1. In SBS we start with the full feature set and remove the worst feature, and then repeat until 5 features remain.

15 features to pick and remove 1 during first round of SBS. 14 features to pick and remove 1 during the second round of SBS etc. We go on until we are left with 5 features.

15+14+13+12…6 = 105 criterion function invokes

But we need to also consider the subset of features with size less than 5, we need to consider size 4,3,2,1. So we need to choose between 5,4,3, and 2 features as well:

15+14+13+12…6+5+4+3+2 = **119 criterion function invokes**

1. We simply need to all the possible ways to form a subset of 5 features given a total of 15 features. Since the order we pick the features is not important, we should calculate the combinatorics instead of the permutation. We need to also consider the subset of features with size less than 5 as following: