CSE 802 Homework 3

By Abhiram Durgaraju

Problem 1)

a)

$$p(x|\theta) = 2\theta x e^{-\theta x^2}$$

The likelihood function is the product for each $p(x_i|\theta)$ in the above distribution:

$$L(\theta) = \prod_{i=1}^{n} 2\,\theta x_i e^{-\theta x_i^2}$$

Convert to log likelihood to simplify the equation:

$$l(\theta) = ln\left(\prod_{i=1}^{n} 2\,\theta x_i e^{-\theta x_i^2}\right)$$

The logarithm converts the product function to a sum:

$$\sum_{1=1}^{n}\left(\ln(2\theta) + \ln(x_i) - \theta x_i^2\right) =$$

$$\mathrm{n}\ln(2\theta) + \sum_{1=1}^{n}\left(\ln(x_i) - \theta x_i^2\right)$$

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

$$\frac{\partial}{\partial\theta} l(\theta) = \frac{n}{\theta} + \frac{\partial}{\partial\theta}\left(\sum_{1=1}^{n}\left(\ln(x_i) - \theta x_i^2\right)\right) = 0$$

Since we are taking a derivative with respect to θ, $\sum_{1=1}^{n} \ln(x_i)$ can be treated as a constant, and therefore ignored.

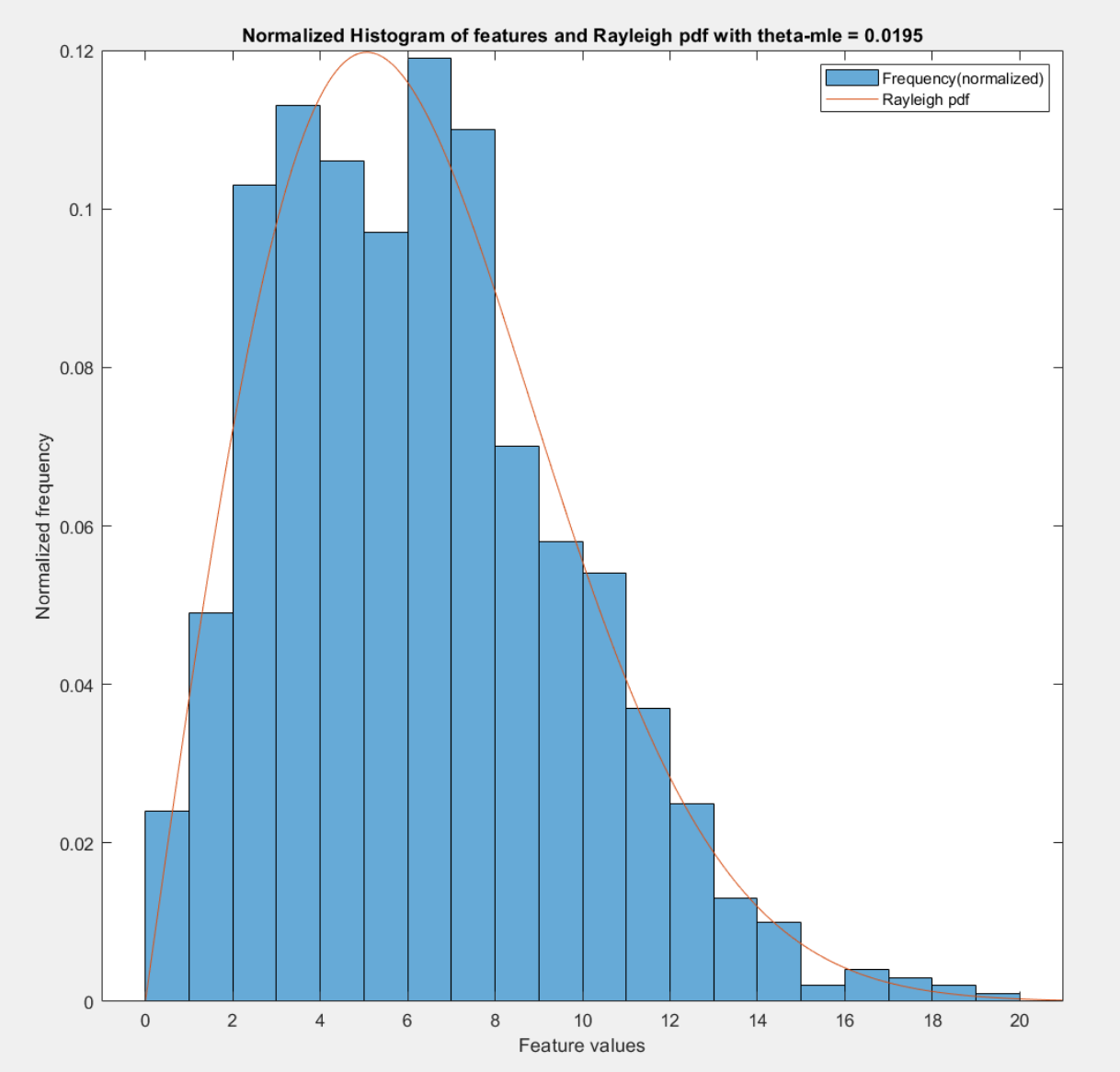
$$\frac{\partial}{\partial\theta} l(\theta) = \frac{n}{\theta} + \frac{\partial}{\partial\theta}\left(\sum_{1=1}^{n}\left(-\theta x_i^2\right)\right) = 0$$

$$\frac{\partial}{\partial\theta} l(\theta) = \frac{n}{\theta} + \sum_{1=1}^{n}\left(-x_i^2\right) = 0$$

$$\frac{n}{\theta} = \sum_{1=1}^{n}\left(x_i^2\right)$$

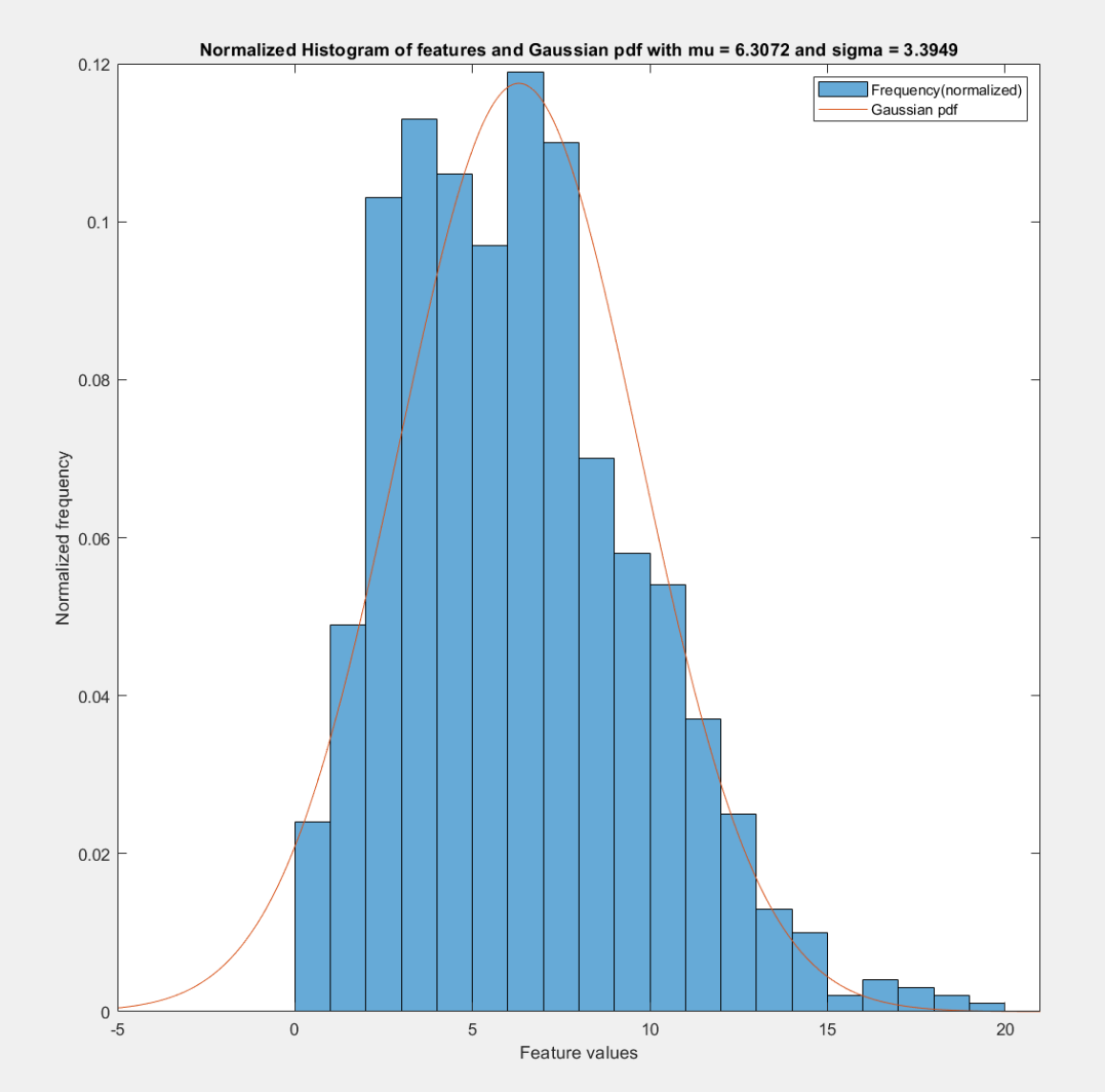
$$\hat{\theta}_{MLE} = \frac{n}{\sum_{1=1}^{n}\left(x_i^2\right)}$$

b)



Using the derived formula, I found that $\hat{\theta}_{MLE} = 0.0195$

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.

```
D = importdata('hw3_random_data_q1.txt');
h = histogram(D, 'Normalization', 'probability');
hold on;
sum_of_squares = 0;
for i=1:1000
    sum_of_squares = sum_of_squares + (D(i))^2;
end
theta_mle = size(D,1)/sum_of_squares;
b = sqrt(1/(2*theta_mle));
x = [-5:0.01:21];
p1 = raylpdf(x, b);
% plot(x,p1);
% xlabel('Feature values');
% ylabel('Normalized frequency');
% title('Normalized Histogram of features and Rayleigh pdf with theta-mle = 0.0195');
% legend('Frequency(normalized)', 'Rayleigh pdf');
mu = mean(D);
s = std(D, 1);
p2 = normpdf(x, mu, s);
plot(x,p2);
xlabel('Feature values');
ylabel('Normalized frequency');
title('Normalized Histogram of features and Gaussian pdf with mu = 6.3072 and sigma = 3.3949');
legend('Frequency(normalized)', 'Gaussian pdf');
```

Problem 2)

a)

$$p(x|\theta) \sim U(0,\theta) = \begin{cases} 1/\theta\,, & 0 \le x \le \theta \\ 0, & \text{otherwise} \end{cases}$$

The likelihood function is the product of each $p(x_i|\theta)$ where $i = 1,2,3,\dots n$

$$L(\theta) = \begin{cases} \prod_{i=1}^{n} 1/\theta\,, & 0 \le x_i \le \theta \\ 0, & \text{otherwise} \end{cases}$$

$$L(\theta) = \begin{cases} \theta^{-n}, & 0 \le x_i \le \theta \\ 0, & \text{otherwise} \end{cases}$$

The log likelihood is given by:

$$l(\theta) = -n log(\theta), \qquad 0 \le x_i \le \theta$$

$$\frac{\partial}{\partial\theta} l(\theta) = \frac{-n}{\theta}, \qquad 0 \le x_i \le \theta$$

$L(\theta)$ is a decreasing function, as shown by the derivative. Therefore, it is maximized when θ is minimized. But we are given the constraint that $\theta \ge x_i \ge 0$ for all $i = 1,2,3,\dots n$. The smallest value of θ under the constraint that $\theta \ge x_i \ge 0$ is simply the $\max(x_1, x_2, \dots x_n) = \max(D)$

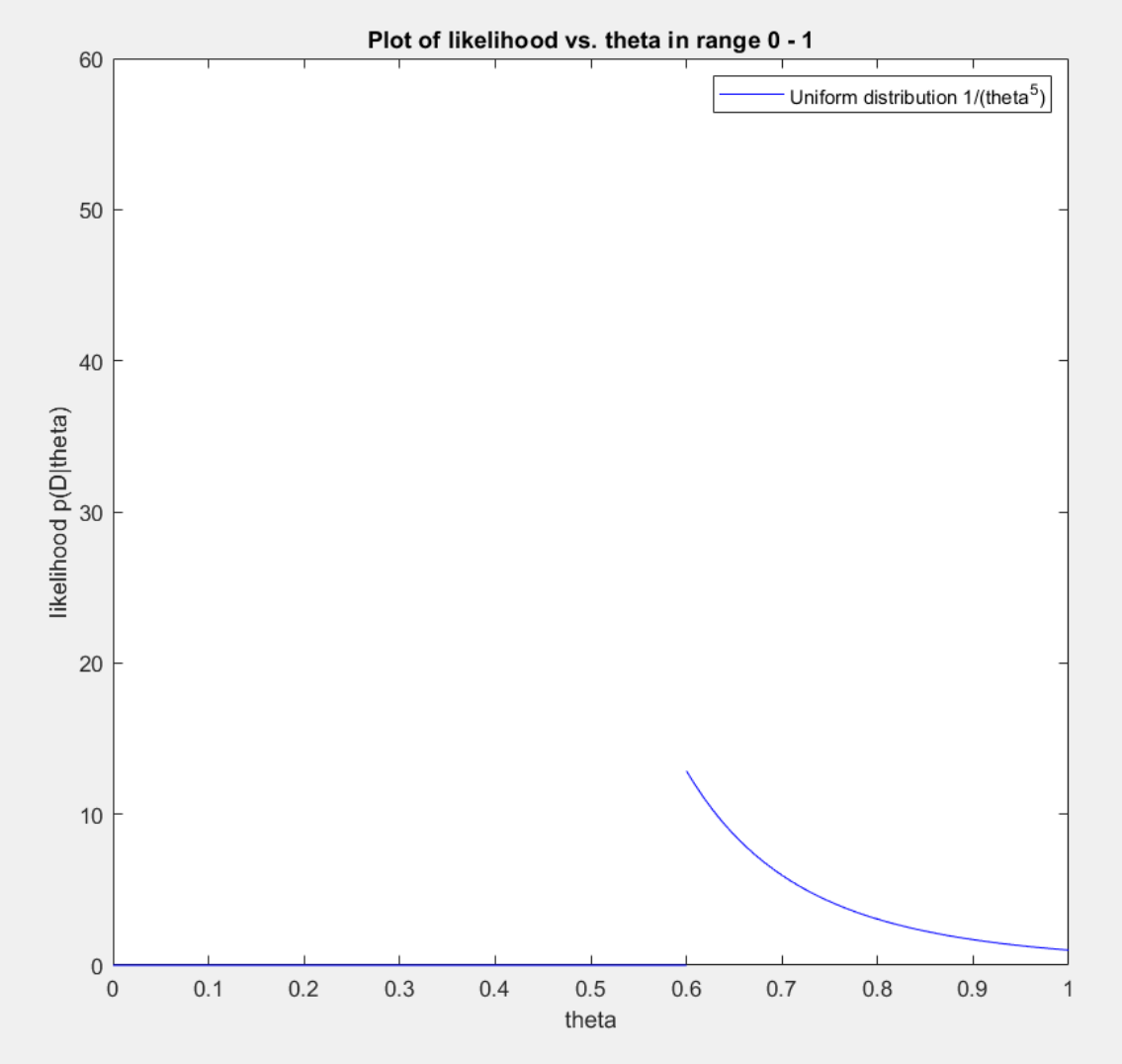
Therefore $\hat{\theta}_{MLE} = \max(D)$

b)

Since we are given that $n = 5$, we know that:

$$L(\theta) = \begin{cases} \prod_{i=1}^{5} 1/\theta\,, & 0 \le x_i \le \theta \\ 0, & \text{otherwise} \end{cases}$$

$$L(\theta) = \frac{1}{\theta^5}$$



As seen in the plot, the likelihood is maximized when θ is as small as possible. Given our constraint that $0 \le x_i \le \theta$, for all $i = 1,2,3, \ldots n$, the value of θ is maximum at $\hat{\theta}_{MLE} = \max(D)$. Therefore, we do not need to know the values of the other 4 points, as $\hat{\theta}_{MLE}$ is purely dictated by the maximum value in the dataset, which is 0.6. When $\theta \le 0.6$, $p(D|\theta) = 0$. Therefore, we must pick $\hat{\theta}_{MLE} = \max(D)$

```
syms theta
likelihood = 1/(theta^5);
likelihood2 = 0;
fplot(likelihood, [0.6,1], 'Color', 'b');
hold on;
fplot(likelihood2, [0,0.6],'Color', 'b');
ylim([0,60]);
title('Plot of likelihood vs. theta in range 0 - 1');
xlabel('theta');
ylabel('likelihood p(D|theta)');
legend('Uniform distribution 1/(theta^5)');
```

Problem 3)

Let us consider deriving θ_1, a specific component of MLE of $\widehat{\boldsymbol{\theta}}$. For θ_1, the Bernoulli distribution is as follows:

$$p(x|\theta_1) = \prod_{i=1}^{d} \theta_1^{x_i}(1-\theta_1)^{1-x_i}$$

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

$$l(\theta_1) = \ln\left(\prod_{i=1}^{d} \theta_1^{x_i}(1-\theta_1)^{1-x_i}\right)$$

$$l(\theta_1) = \sum_{i=1}^{d}(x_i \ln(\theta_1) + (1-x_i)\ln(1-\theta_1))$$

$$l(\theta_1) = \ln(\theta_1)\sum_{i=1}^{d} x_i + \ln(1-\theta_1)\left(d - \sum_{i=1}^{d} x_i\right)$$

To maximize the log-likelihood, we need to take the derivative with respect to θ_1 and set equal to zero:

$$\frac{\partial}{\partial\theta} l(\theta) = \frac{1}{\theta_1}\sum_{i=1}^{d} x_i - \frac{1}{1-\theta_1}\left(d - \sum_{i=1}^{d} x_i\right) = 0$$

$$\frac{1}{\theta_1}\sum_{i=1}^{d} x_i - \frac{d}{1-\theta_1} + \frac{\sum_{i=1}^{d} x_i}{1-\theta_1} = 0$$

$$\left(\sum_{i=1}^{d} x_i\right)\left(\frac{1}{\theta_1} + \frac{1}{1-\theta_1}\right) - \frac{d}{1-\theta_1} = 0$$

$$\left(\sum_{i=1}^{d} x_i\right)\left(\frac{1}{\theta_1} + \frac{1}{1-\theta_1}\right) = \frac{d}{1-\theta_1}$$

$$\left(\sum_{i=1}^{d} x_i\right)\left(\frac{1}{\theta_1(1-\theta_1)}\right) = \frac{d}{1-\theta_1}$$

$$\left(\frac{\sum_{i=1}^{d} x_i}{\theta_1(1-\theta_1)}\right) = \frac{d}{1-\theta_1}$$

$$d\theta_1(1-\theta_1) = (1-\theta_1)\sum_{i=1}^{d} x_i$$

$$d\theta_1 = \sum_{i=1}^{d} x_i$$

$$\theta_1 = \frac{1}{d}\sum_{i=1}^{d} x_i$$

As shown above, it clear that the parameter θ_1 is simply the sample mean of $\boldsymbol{x}_1 = (x_1, x_2, .. x_d)^t$. We can simply extend this principle to n training samples to obtain $\widehat{\boldsymbol{\theta}}$, which is the sample mean of each pattern. So instead of d dimensions of $\boldsymbol{x}_1$, we have n training samples of $\boldsymbol{x_k}$ for all $k = 1,2,3 \ldots n$:

$$\widehat{\boldsymbol{\theta}} = \frac{1}{n}\sum_{k=1}^{n} x_k$$

Problem 4)

For 50 random samples:

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

$$\boldsymbol{\mu_1} = [0.2840, -0.0379]$$
$$\boldsymbol{\mu_2} = [4.8460, 5.0086]$$
$$\boldsymbol{\Sigma_1} = \begin{bmatrix} 1.5621 & 0.1462 \\ 0.1462 & 1.0614 \end{bmatrix}$$
$$\boldsymbol{\Sigma_2} = \begin{bmatrix} 1.1637 & -0.1591 \\ -0.1591 & 0.8231 \end{bmatrix}$$

b) The Bayes decision boundary x can be calculated when the posterior probability of the two classes are equal:

$$P(\omega_1|x) = P(\omega_2|x)$$

Using Bayes formula:

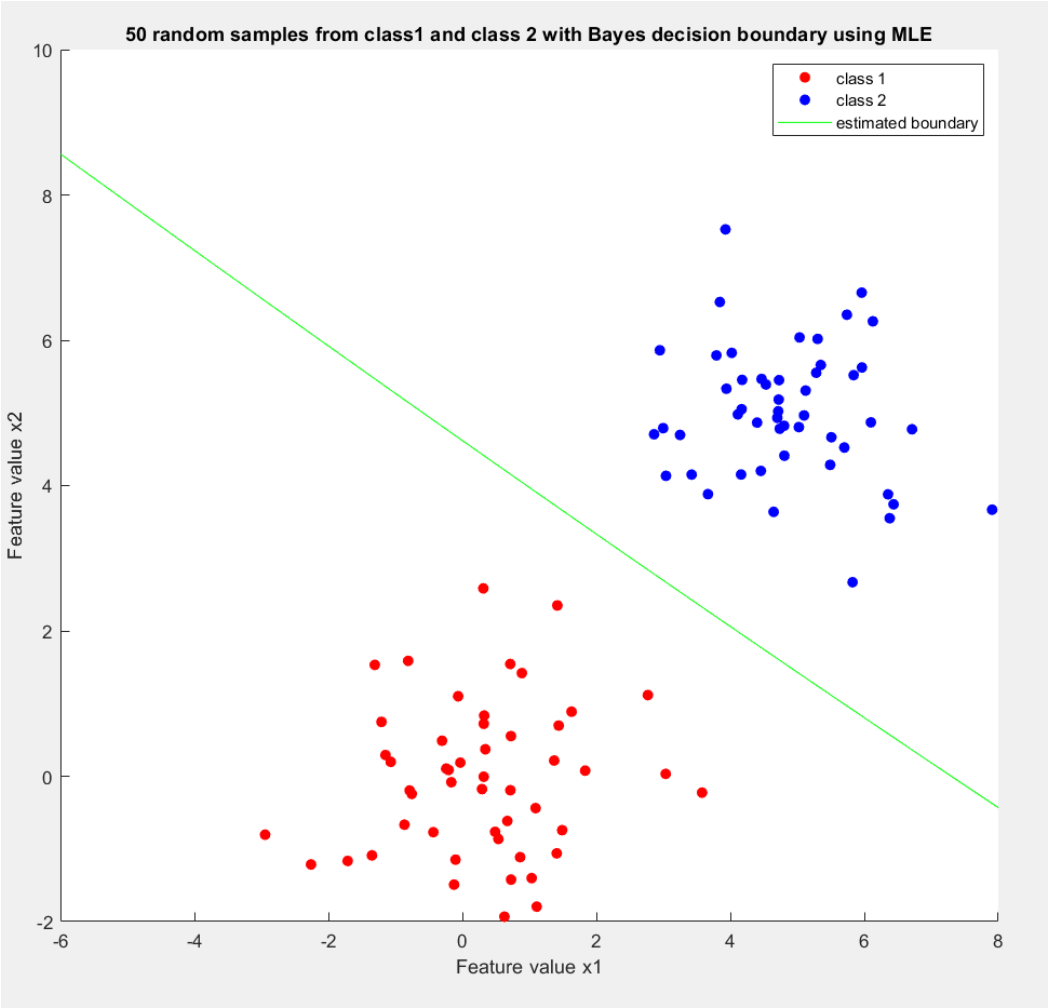
$$\frac{p(x|\omega_1)P(\omega_1)}{p(x)} = \frac{p(x|\omega_2)P(\omega_2)}{p(x)}$$

Since the priors are equal and $p(x)$ is on both sides, the expression can be simplified to:

$$p(x|\omega_1) = p(x|\omega_2)$$

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%

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

$$p(x|\omega_1) = p(x|\omega_2)$$

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

$$p(x|\omega_1) = \frac{1}{(2\pi)^{2/2}(1)^{1/2}} e^{\left(-\frac{1}{2}\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}-\begin{bmatrix}0\\0\end{bmatrix}\right)^t \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}-\begin{bmatrix}0\\0\end{bmatrix}\right)\right)}$$

$$p(x|\omega_2) = \frac{1}{(2\pi)^{2/2}(1)^{1/2}} e^{\left(-\frac{1}{2}\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}-\begin{bmatrix}5\\5\end{bmatrix}\right)^t \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}-\begin{bmatrix}5\\5\end{bmatrix}\right)\right)}$$

$$\frac{1}{(2\pi)^{2/2}(1)^{1/2}} e^{\left(-\frac{1}{2}\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}-\begin{bmatrix}0\\0\end{bmatrix}\right)^t \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}-\begin{bmatrix}0\\0\end{bmatrix}\right)\right)} = \frac{1}{(2\pi)^{2/2}(1)^{1/2}} e^{\left(-\frac{1}{2}\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}-\begin{bmatrix}5\\5\end{bmatrix}\right)^t \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}-\begin{bmatrix}5\\5\end{bmatrix}\right)\right)}$$

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

$$\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\right)^t \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\right) = \left(\begin{bmatrix}x_1\\x_2\end{bmatrix}-\begin{bmatrix}5\\5\end{bmatrix}\right)^t \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}-\begin{bmatrix}5\\5\end{bmatrix}\right)$$

$$\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\right)^t \left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\right) = \left(\begin{bmatrix}x_1-5\\x_2-5\end{bmatrix}\right)^t \left(\begin{bmatrix}x_1-5\\x_2-5\end{bmatrix}\right)$$

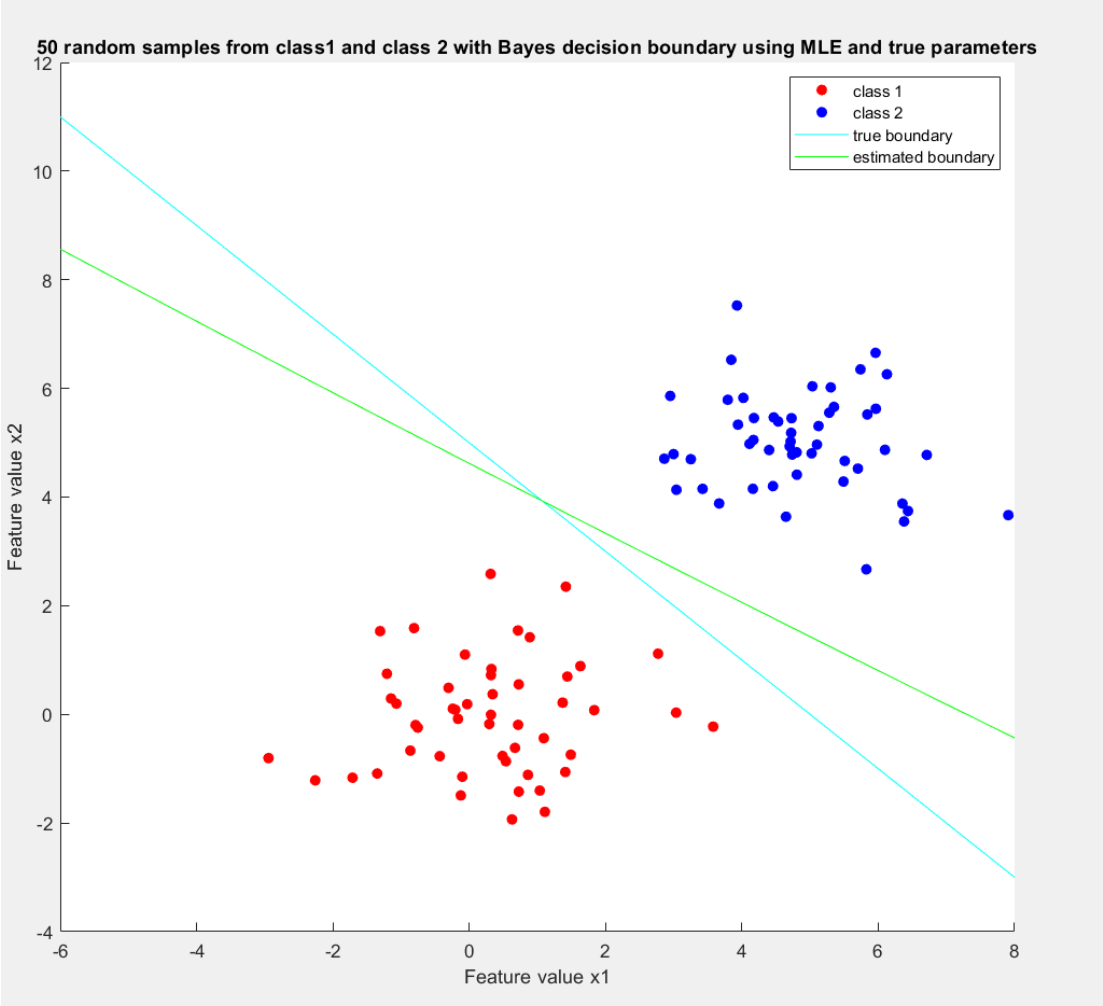
$$x_1{}^2 + x_2{}^2 = (x_1-5)^2 + (x_2-5)^2$$

$$x_1{}^2 + x_2{}^2 = x_1{}^2 - 10x_1 + 25 + x_2{}^2 - 10x_2 + 25$$

$$10x_1 + 10x_2 = 50$$

$$x_2 = \frac{50 - 10x_1}{10}$$

$$x_2 = 5 - x_1$$



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

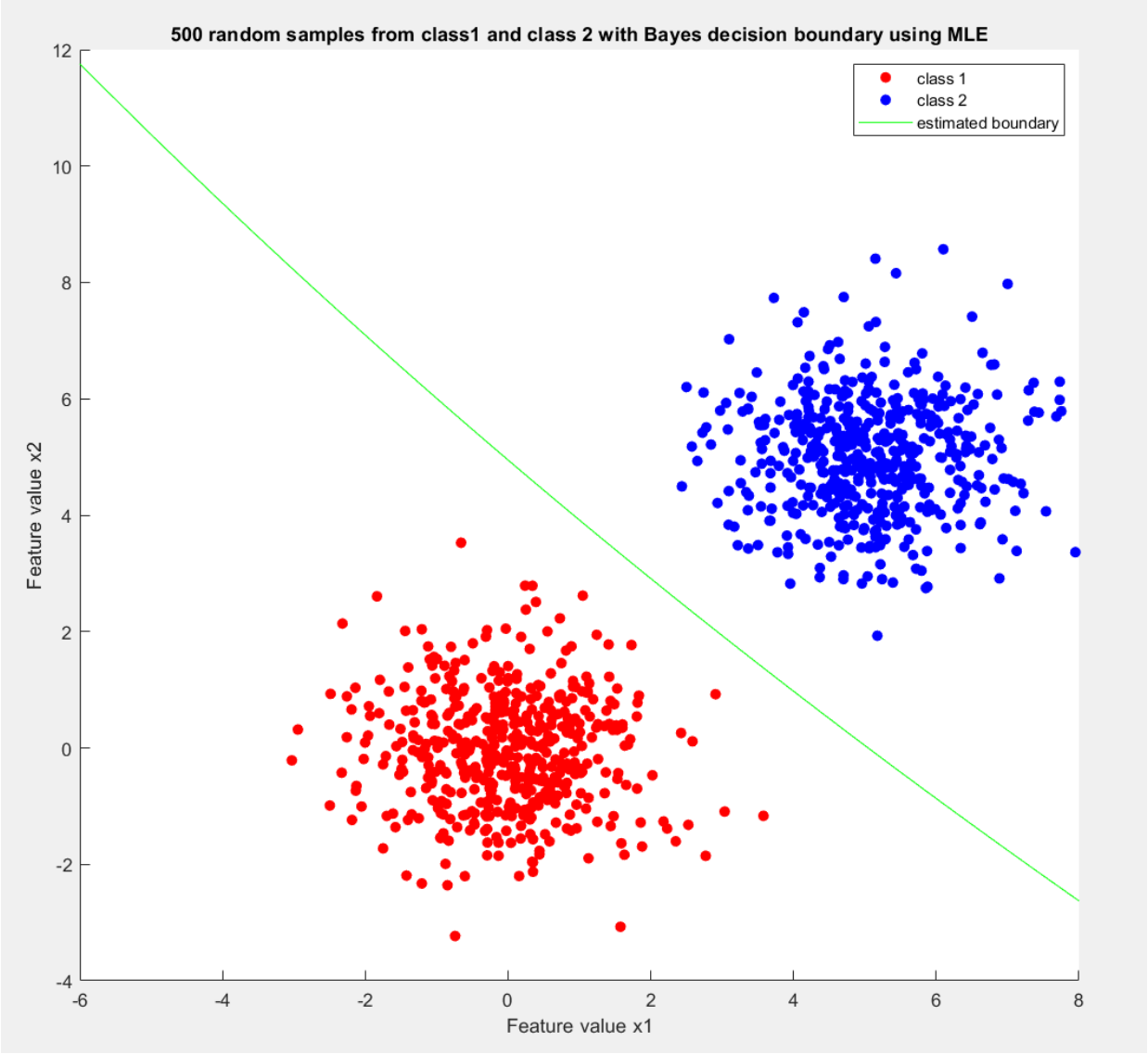
For 500 random samples:

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

$$\boldsymbol{\mu_1} = [\,-0.0219, -0.0433]$$
$$\boldsymbol{\mu_2} = [5.0634, 5.0104]$$
$$\boldsymbol{\Sigma_1} = \begin{bmatrix} 1.0058 & -0.0603 \\ -0.0603 & 0.9878 \end{bmatrix}$$
$$\boldsymbol{\Sigma_2} = \begin{bmatrix} 1.0122 & 0.0074 \\ 0.0074 & 0.9788 \end{bmatrix}$$

b) Using MATLAB,

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

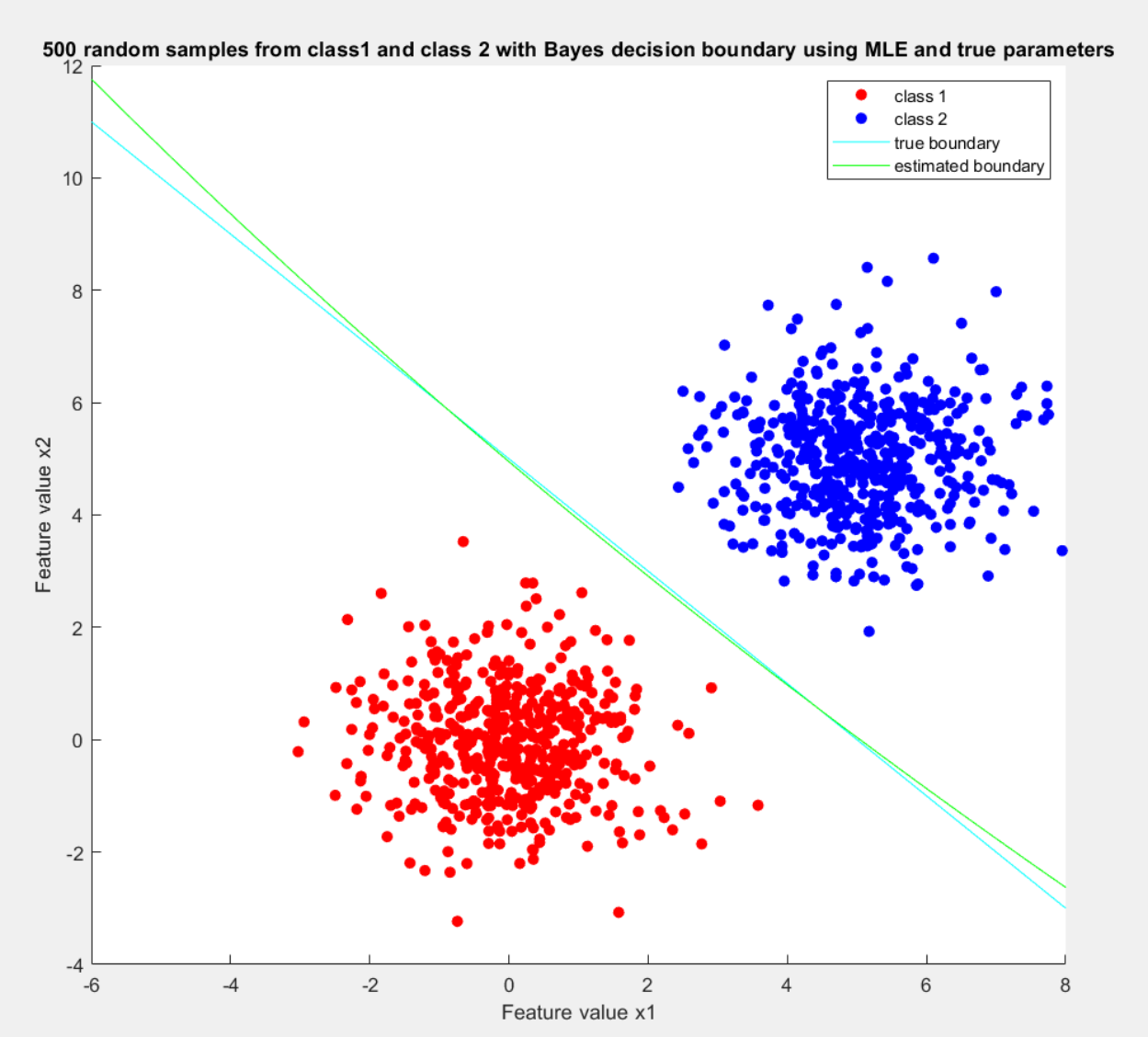


The empirical error using estimated parameters is 0/1000 = 0%

c)

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

$$x_2 = 5 - x_1$$



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

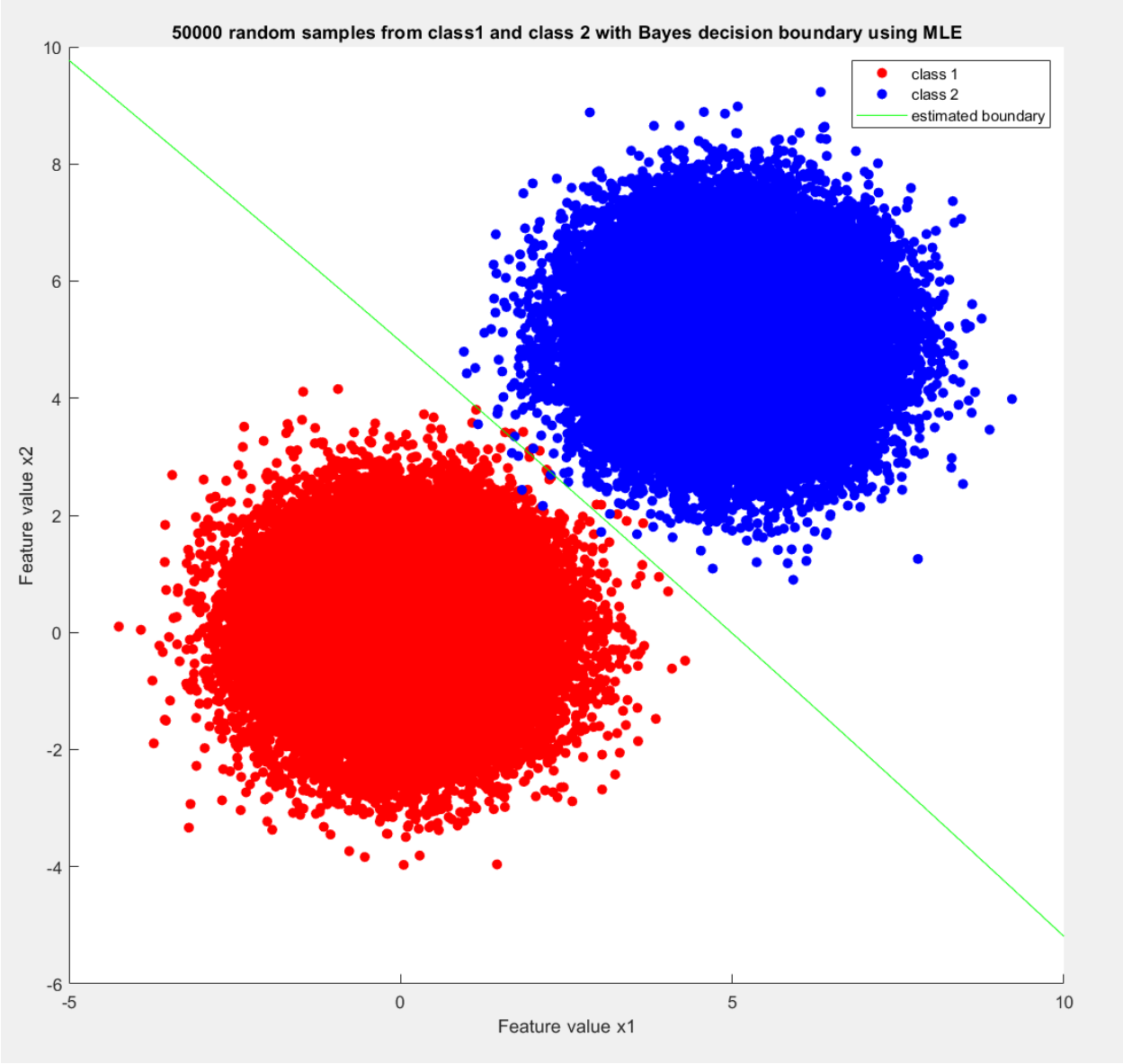
For 50000 random samples:

a)

$$\boldsymbol{\mu_1} = [-0.0030, 0.0014]$$
$$\boldsymbol{\mu_2} = [4.9983, 5.0016]$$
$$\boldsymbol{\Sigma_1} = \begin{bmatrix} 0.9916 & 0.0014 \\ 0.0014 & 0.9959 \end{bmatrix}$$
$$\boldsymbol{\Sigma_2} = \begin{bmatrix} 1.0093 & -0.0076 \\ -0.0076 & 0.9980 \end{bmatrix}$$

b) Using MATLAB, the Bayes decision boundary using estimated parameters is:

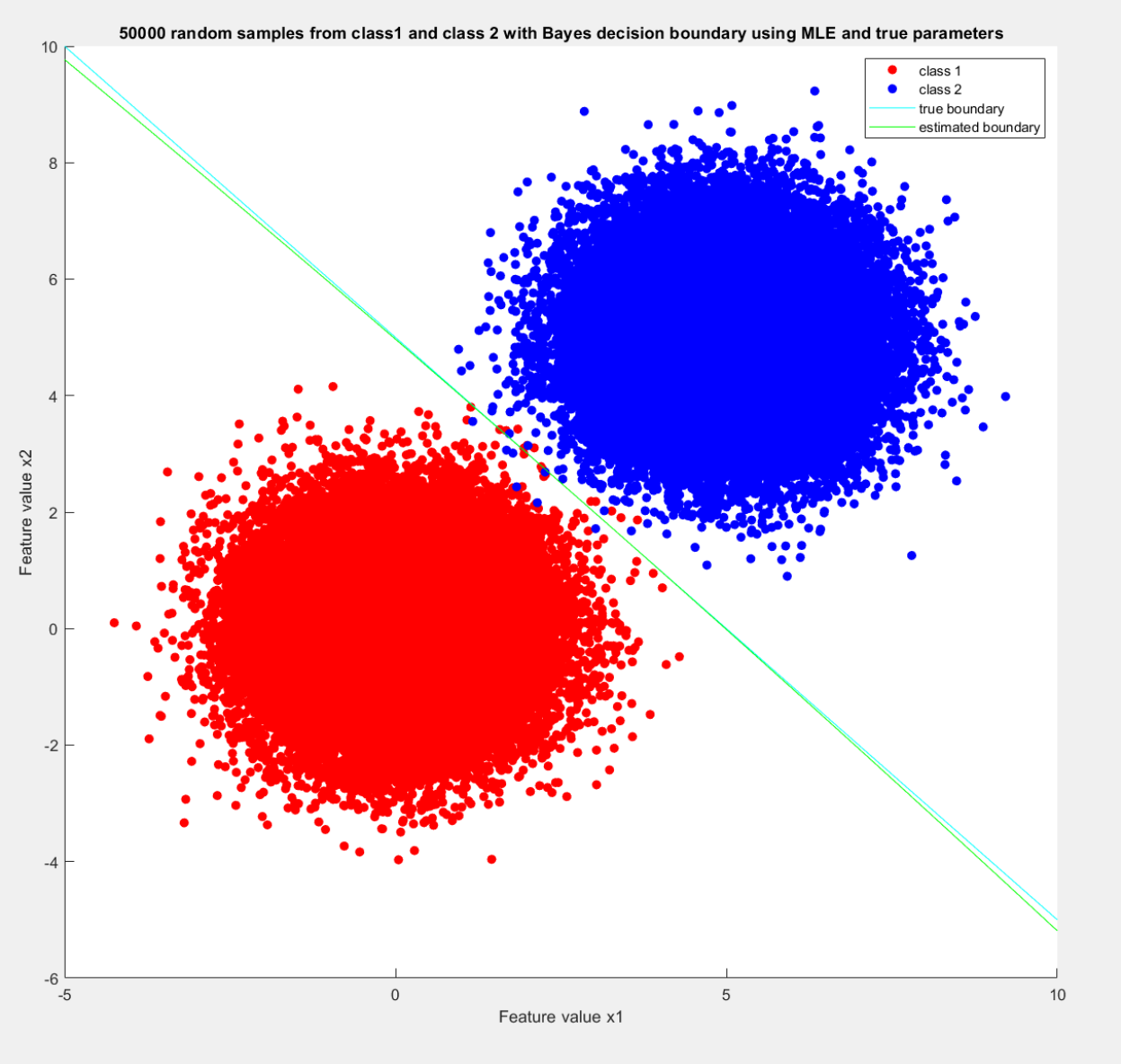
x2 = 4.4734*x1 + 498.6*(0.000045131*x1^2 - 0.11062*x1 + 25.588)^(1/2) - 2517.2



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%

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

$$x_2 = 5 - x_1$$



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%

d) 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.

```
rng('default');
mu = [0, 0];
sigma = [1 0; 0 1];
mu2 = [5, 5];
sigma2 = [1 0;0 1];
determ = det(sigma);
determ2 = det(sigma2);
invers = inv(sigma);
invers2 = inv(sigma2);
R1 = mvnrnd(mu, sigma, 50000);
R2 = mvnrnd(mu2, sigma2, 50000);
mu_mle = mean(R1);
mu2_mle = mean(R2);
cov_mle = cov(R1, 1);
cov2_mle = cov(R2, 1);
determ_mle = det(cov_mle);
determ2_mle = det(cov2_mle);
invers_mle = inv(cov_mle);
invers2_mle = inv(cov2_mle);

syms x1 x2 real
multiGauss = (1/(sqrt(determ*(2*pi)^2)))*exp((-1/2)*([x1 x2]-mu)*invers*([x1 x2]-mu)') == (1/(sqrt(determ2*(2*pi)^2)))*exp((-1/2)*([x1 x2]-mu2)*invers2*([x1 x2]-mu2)');
multiGauss_mle = (1/(sqrt(determ_mle*(2*pi)^2)))*exp((-1/2)*([x1 x2]-mu_mle)*invers_mle*([x1 x2]-mu_mle)') == ...
    (1/(sqrt(determ2_mle*(2*pi)^2)))*exp((-1/2)*([x1 x2]-mu2_mle)*invers2_mle*([x1 x2]-mu2_mle)');
S = solve(multiGauss, x2);
S_mle = solve(multiGauss_mle, x2);

scatter(R1(:, 1), R1(:, 2), 'filled', 'MarkerFaceColor', 'red');
hold on;
scatter(R2(:, 1), R2(:, 2), 'filled','MarkerFaceColor', 'blue');
f1 = fplot(S);
f1.Color = 'c';
f2 = fplot(S_mle(2));
f2.Color = 'g';
digits(5);
vpa(S_mle(2))
xlabel('Feature value x1');
ylabel('Feature value x2');
title('50000 random samples from class1 and class 2 with Bayes decision boundary using MLE and true parameters');
legend('class 1', 'class 2', 'true boundary', 'estimated boundary');
```

Problem 5)

a) The following were calculated using MATLAB:

$$\boldsymbol{\mu_1} = [\,5.0280, 3.4800, 1.4600, 0.2480]$$
$$\boldsymbol{\mu_2} = [6.0120, 2.7760, 4.3120, 1.3440]$$
$$\boldsymbol{\mu_3} = [6.5760, 2.9280, 5.6400, 2.0440]$$
$$\boldsymbol{\Sigma_1} = \begin{bmatrix} 0.1604 & 0.1181 & 0.0241 & 0.0194 \\ 0.1181 & 0.1358 & 0.0062 & 0.0222 \\ 0.0241 & 0.0062 & 0.0392 & 0.0066 \\ 0.0194 & 0.0222 & 0.0066 & 0.0109 \end{bmatrix}$$
$$\boldsymbol{\Sigma_2} = \begin{bmatrix} 0.3003 & 0.1095 & 0.1865 & 0.0520 \\ 0.1095 & 0.1244 & 0.0886 & 0.0465 \\ 0.1865 & 0.0886 & 0.1969 & 0.0640 \\ 0.0520 & 0.0465 & 0.0640 & 0.0426 \end{bmatrix}$$
$$\boldsymbol{\Sigma_3} = \begin{bmatrix} 0.5244 & 0.1215 & 0.4323 & 0.0619 \\ 0.1215 & 0.1304 & 0.0988 & 0.0604 \\ 0.4323 & 0.0988 & 0.4175 & 0.0673 \\ 0.0619 & 0.0604 & 0.0673 & 0.0651 \end{bmatrix}$$

b) Since the priors are equal and this is a 0-1 loss function, the maximum posterior rule can be simplified using Bayes formula:

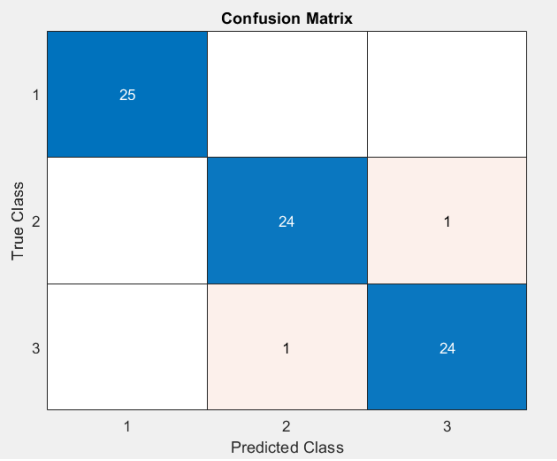
$$\mathrm{j} = \operatorname{argmax}\left(P(\omega_1|x), P(\omega_2|x), P(\omega_3|x)\right) = \operatorname{argmax}\left(\frac{p(x|\omega_1)P(\omega_1)}{p(x)}, \frac{p(x|\omega_2)P(\omega_2)}{p(x)}, \frac{p(x|\omega_3)P(\omega_3)}{p(x)}\right)$$

The priors are equal and $p(x)$ is common to all three classes. So these can be cancelled out:

$$\mathrm{j} = \operatorname{argmax}(p(\omega_1|x), p(\omega_2|x), p(\omega_3|x))$$

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

c)



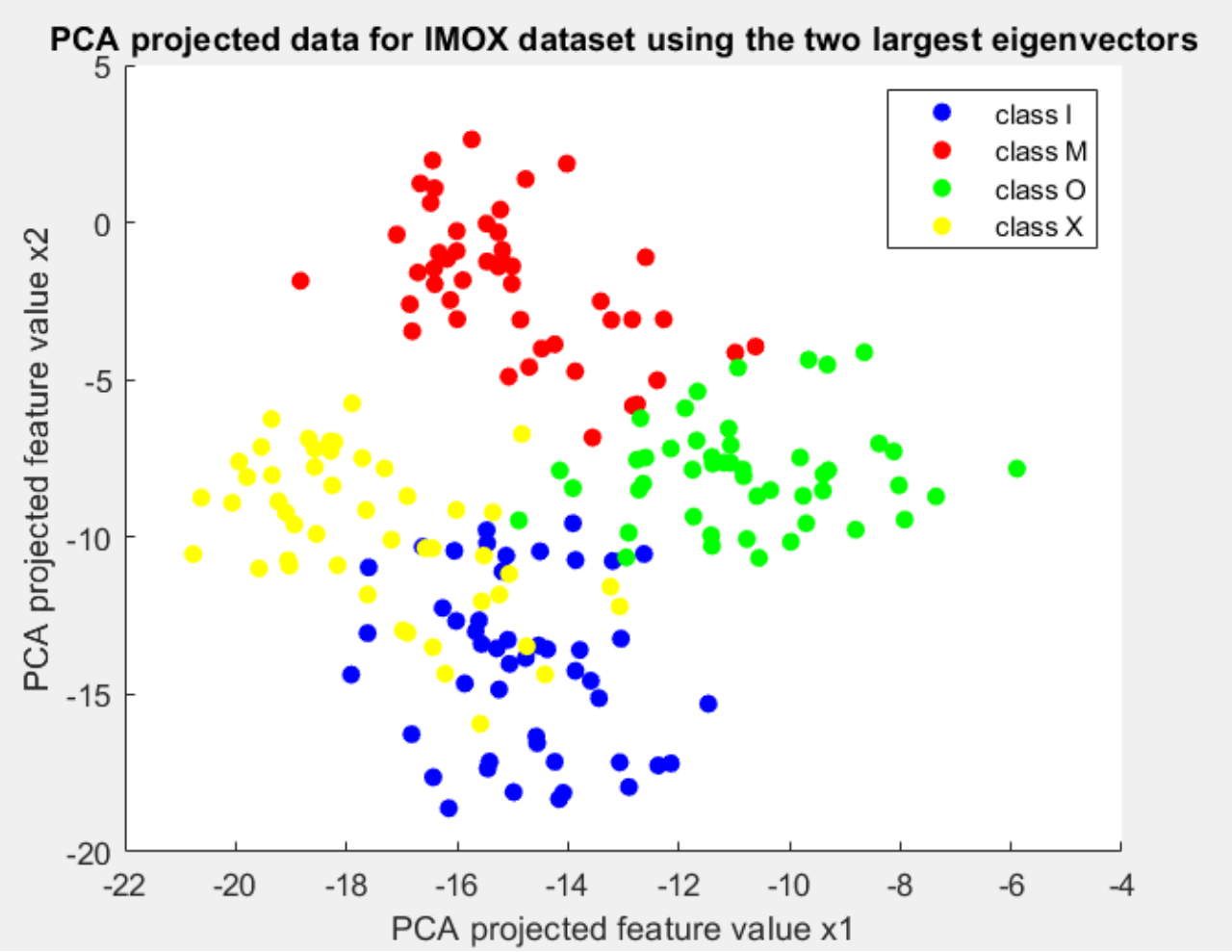
Empirical error rate = 2/75 = 2.666%

```
D = importdata('iris_data.txt');

%class 1
mu1 = mean(D(1:25, 1:4));
sigma1 = cov(D(1:25, 1:4));
determ1 = det(sigma1);
invers1 = inv(sigma1);

%class 2
mu2 = mean(D(51:75,1:4));
sigma2 = cov(D(51:75, 1:4));
determ2 = det(sigma2);
invers2 = inv(sigma2);

%class 3
mu3 = mean(D(101:125, 1:4));
sigma3 = cov(D(101:125, 1:4));
determ3 = det(sigma3);
invers3 = inv(sigma3);

syms x1 x2 x3 x4
multiGauss1(x1, x2, x3 ,x4) = (1/(sqrt(determ1*(2*pi)^4)))*exp((-1/2)*([x1 x2 x3 x4]-mu1)*invers1*([x1 x2 x3 x4]-mu1)');
multiGauss2(x1, x2, x3 ,x4) = (1/(sqrt(determ2*(2*pi)^4)))*exp((-1/2)*([x1 x2 x3 x4]-mu2)*invers2*([x1 x2 x3 x4]-mu2)');
multiGauss3(x1, x2, x3 ,x4) = (1/(sqrt(determ3*(2*pi)^4)))*exp((-1/2)*([x1 x2 x3 x4]-mu3)*invers3*([x1 x2 x3 x4]-mu3)');

A = zeros(1,3);
predicted_class = zeros(75,1);
testing_set = [D(26:50,1:5);D(76:100,1:5);D(126:150,1:5)];

for i=1:75
   A(1) = multiGauss1(testing_set(i,1), testing_set(i,2), testing_set(i,3), testing_set(i,4));
   A(2) = multiGauss2(testing_set(i,1), testing_set(i,2), testing_set(i,3), testing_set(i,4));
   A(3) = multiGauss3(testing_set(i,1), testing_set(i,2), testing_set(i,3), testing_set(i,4));
   [M,I] = max(A);
   predicted_class(i) = I;
end

C = confusionmat(testing_set(:,5), predicted_class);
confusionchart(C)
title('Confusion Matrix')
```

Problem 6)

a) Projection matrix using PCA:

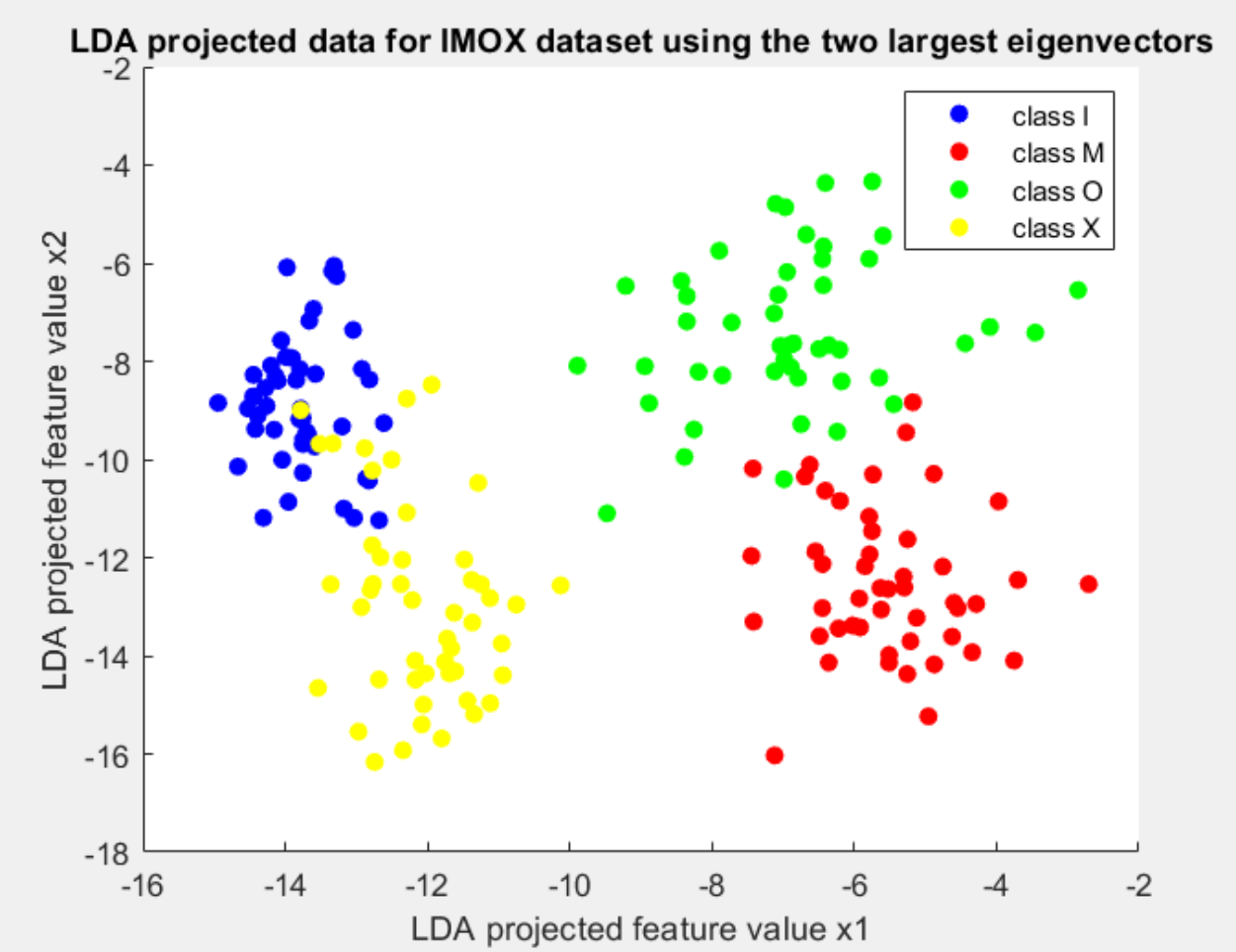
e1	e2
0.0145	-0.2153
-0.0883	-0.4018
-0.1640	-0.2741
-0.0504	-0.0051
-0.3403	-0.3764
-0.4758	-0.5049
-0.5572	0.3997
-0.5567	0.4008



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.

```
D = importdata('imox_data.txt');

%PCA
[eigen_vec, eigen_val] = eig(cov(D(:,1:8)));
projected_data = D(:,1:8)*eigen_vec(:,7:8);
hold on;
% scatter(projected_data(1:48,1), projected_data(1:48,2), 'filled', 'MarkerFaceColor', 'b');
% scatter(projected_data(49:96,1), projected_data(49:96,2), 'filled', 'MarkerFaceColor', 'r');
% scatter(projected_data(97:144,1), projected_data(97:144,2), 'filled', 'MarkerFaceColor', 'g');
% scatter(projected_data(145:192,1), projected_data(145:192,2), 'filled', 'MarkerFaceColor', 'y');
% xlabel('PCA projected feature value x1');
% ylabel('PCA projected feature value x2');
% title('PCA projected data for IMOX dataset using the two largest eigenvectors');
% legend('class I', 'class M', 'class O', 'class X');


mean_1 = mean(D(1:48,1:8));
mean_2 = mean(D(49:96,1:8));
mean_3 = mean(D(97:144,1:8));
mean_4 = mean(D(145:192,1:8));
mean_total = mean(D(:,1:8));

difference_matrix_1 = D(1:48,1:8) - mean_1;
difference_matrix_1 = difference_matrix_1';
between_difference_1 = mean_1 - mean_total;
between_difference_1 = between_difference_1';

difference_matrix_2 = D(49:96,1:8) - mean_2;
difference_matrix_2 = difference_matrix_2';
between_difference_2 = mean_2 - mean_total;
between_difference_2 = between_difference_2';

difference_matrix_3 = D(97:144,1:8) - mean_3;
difference_matrix_3 = difference_matrix_3';
between_difference_3 = mean_3 - mean_total;
between_difference_3 = between_difference_3';
```

```
difference_matrix_4 = D(145:192,1:8) - mean_4;
difference_matrix_4 = difference_matrix_4';
between_difference_4 = mean_4 - mean_total;
between_difference_4 = between_difference_4';

within_scatter_1 = zeros(8,8);
within_scatter_2 = zeros(8,8);
within_scatter_3 = zeros(8,8);
within_scatter_4 = zeros(8,8);

for i=1:48
   within_scatter_1 = within_scatter_1 + difference_matrix_1(:,i)*difference_matrix_1(:,i)';
   within_scatter_2 = within_scatter_2 + difference_matrix_2(:,i)*difference_matrix_2(:,i)';
   within_scatter_3 = within_scatter_3 + difference_matrix_3(:,i)*difference_matrix_3(:,i)';
   within_scatter_4 = within_scatter_4 + difference_matrix_4(:,i)*difference_matrix_4(:,i)';
end
within_scatter_total = within_scatter_1 + within_scatter_2 + within_scatter_3 + within_scatter_4;

between_scatter_1 = between_difference_1*between_difference_1';
between_scatter_2 = between_difference_2*between_difference_2';
between_scatter_3 = between_difference_3*between_difference_3';
between_scatter_4 = between_difference_4*between_difference_4';
between_scatter_total = between_scatter_1 + between_scatter_2 + between_scatter_3 + between_scatter_4;
between_scatter_total = 48*between_scatter_total;
scatter_matrix = cov(D(:,1:8))*191

discr = (inv(within_scatter_total))*between_scatter_total;

[eigen_vec_lda, eigen_val_lda] = eig(discr);
projected_data_lda = D(:, 1:8)*eigen_vec_lda(:,1:2);
scatter(projected_data_lda(1:48,1), projected_data_lda(1:48,2), 'filled', 'MarkerFaceColor', 'b');
scatter(projected_data_lda(49:96,1), projected_data_lda(49:96,2), 'filled', 'MarkerFaceColor', 'r');
scatter(projected_data_lda(97:144,1), projected_data_lda(97:144,2), 'filled', 'MarkerFaceColor', 'g');
scatter(projected_data_lda(145:192,1), projected_data_lda(145:192,2), 'filled', 'MarkerFaceColor', 'y');
xlabel('LDA projected feature value x1');
ylabel('LDA projected feature value x2');
title('LDA projected data for IMOX dataset using the two largest eigenvectors');
legend('class I', 'class M', 'class O', 'class X');
```

Problem 7)

a)

For features $i = 1,2,3 \ldots 8$ and classes $j = 1,2,3,4$, 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	μ_{ij}	σ_{ij}
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)

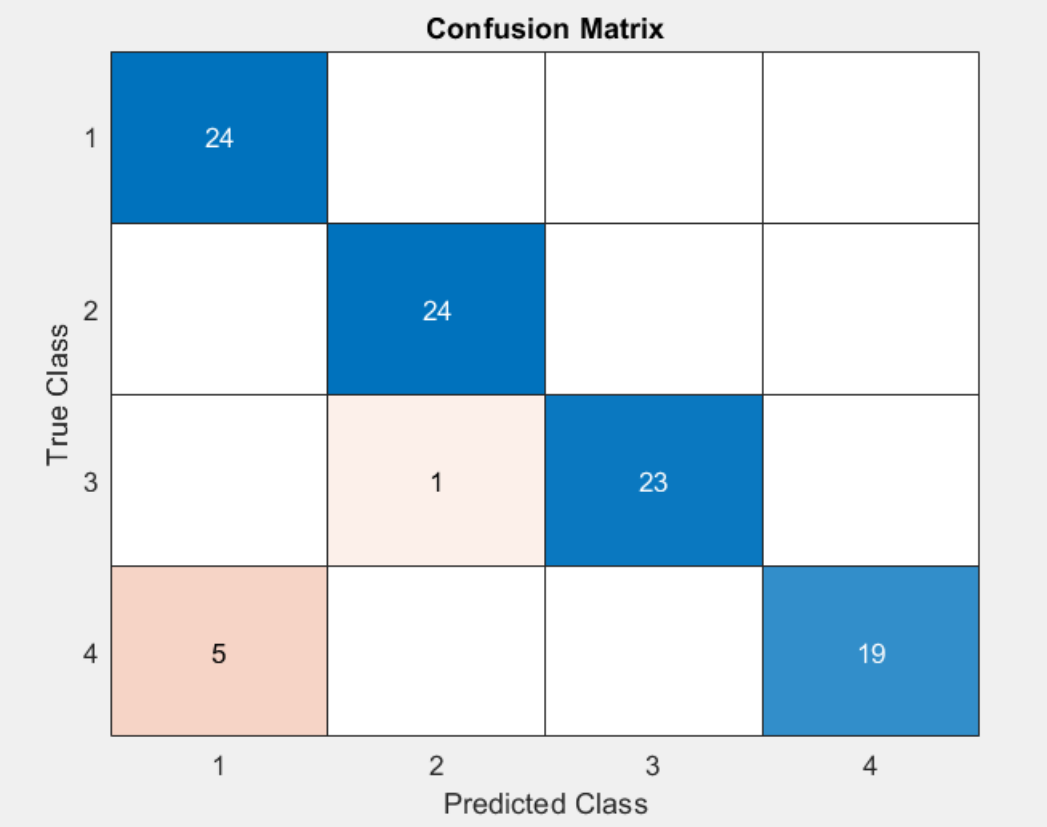
The classifier is the same as in problem #5. We try to maximize the posterior probability for the 4 classes:

$$\begin{aligned} \mathrm{j} &= \operatorname{argmax}\left(P(\omega_1|x), P(\omega_2|x), P(\omega_3|x), P(\omega_4|x)\right) \\ &= \operatorname{argmax}\left(\frac{p(x|\omega_1)P(\omega_1)}{p(x)}, \frac{p(x|\omega_2)P(\omega_2)}{p(x)}, \frac{p(x|\omega_3)P(\omega_3)}{p(x)}, \frac{p(x|\omega_4)P(\omega_4)}{p(x)}\right) \end{aligned}$$

The priors are equal and $p(x)$ is common to all three classes. So these can be cancelled out:

$$\mathrm{j} = \operatorname{argmax}(p(\omega_1|x), p(\omega_2|x), p(\omega_3|x), p(x|\omega_4))$$

c)



The empirical error rate is 6/96 = 6.25%

```
D = importdata('imox_data.txt');
A = zeros(32,2);
for i=1:8
    for j=1:4
        if j == 1
            A(i,1) = mean(D(1:48,i));
            A(i,2) = var(D(1:48,i),1);
        elseif j == 2
            A(i+8,1) = mean(D(49:96,i));
            A(i+8,2) = var(D(49:96,i),1);
        elseif j == 3
            A(i+16,1) = mean(D(97:144,i));
            A(i+16,2) = var(D(97:144,i),1);
        else
            A(i+24,1) = mean(D(145:192,i));
            A(i+24,2) = var(D(145:192,i),1);
        end
    end
end
B = zeros(4,8);
for i = 1:8
    B(1,i) = var(D(1:24,i),1);
    B(2,i) = var(D(49:72,i),1);
    B(3,i) = var(D(97:120,i),1);
    B(4,i) = var(D(145:168,i),1);
end
%mle estimate for variance is biased
mu_1 = mean(D(1:24,1:8));
cov_1 = zeros(8);
cov_2 = zeros(8);
cov_3 = zeros(8);
cov_4 = zeros(8);
for i=1:8
    cov_1(i,i) = B(1,i);
    cov_2(i,i) = B(2,i);
    cov_3(i,i) = B(3,i);
    cov_4(i,i) = B(4,i);
end
mu_2 = mean(D(49:72,1:8));
mu_3 = mean(D(97:120,1:8));
mu_4 = mean(D(145:168,1:8));
```

```
testing_data = [D(25:48,:);D(73:96,:);D(121:144,:);D(169:192,:)];
R = zeros(96, 6);
R(:,1) = mvnpdf(testing_data(:,1:8), mu_1, cov_1);
R(:,2) = mvnpdf(testing_data(:,1:8), mu_2, cov_2);
R(:,3) = mvnpdf(testing_data(:,1:8), mu_3, cov_3);
R(:,4) = mvnpdf(testing_data(:,1:8), mu_4, cov_4);

[M, R(:,5)] = max(R(:,1:4), [], 2); %predicted class
R(:,6) = testing_data(:,9); %true class

C = confusionmat(R(:,6), R(:,5));
confusionchart(C)
title('Confusion Matrix')
```

Problem 8)

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

a) 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.**

b) 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**

c) 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**

d) 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:

$$\binom{15}{5} + \binom{15}{4} + \binom{15}{3} + \binom{15}{2} + \binom{15}{1} = 3003 + 1365 + 455 + 105 + 15$$
$$= \mathbf{4943\ criterion\ function\ invokes}$$