CS 273 Homework 1

Zachary DeStefano, 15247592

Due Date: Tuesday, January 13, 2015

Problem 1

Problem 1, Part a

The number of features is 4
The number of observations is 148

Problem 1, Part b

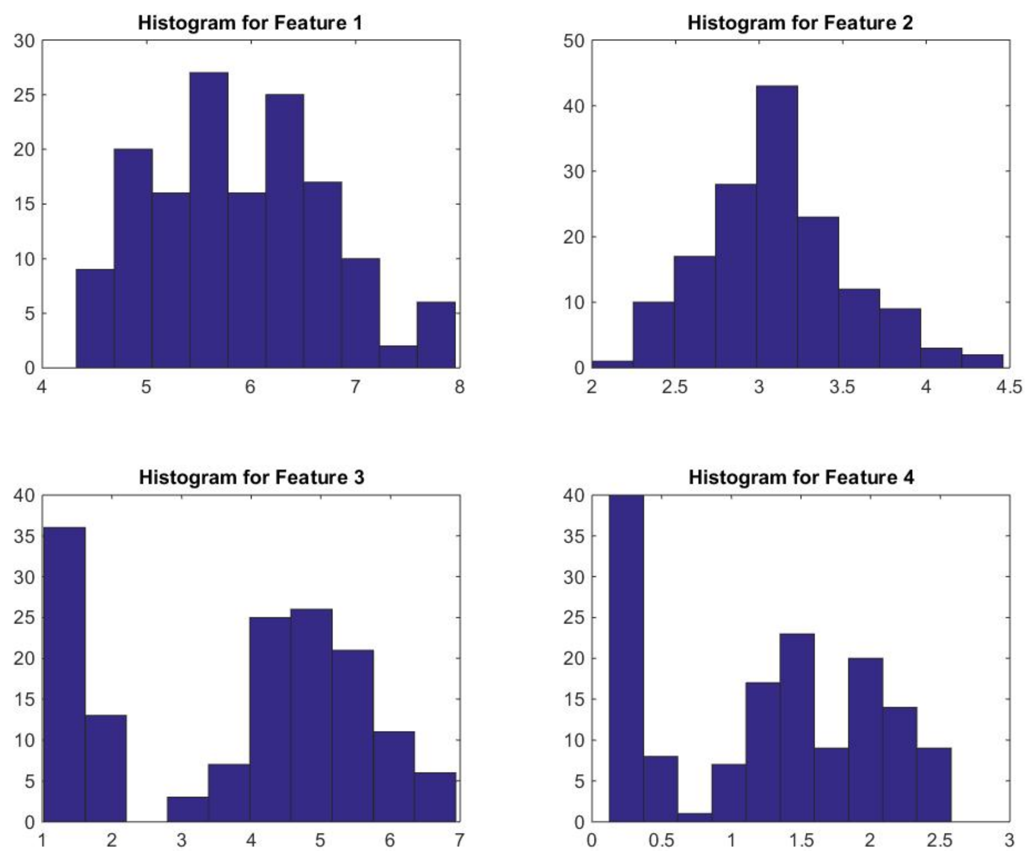

Figure 1: Histograms for each feature

Problem 1, Part c

The mean of feature 1 is 5.9001
The mean of feature 2 is 3.0989
The mean of feature 3 is 3.8196
The mean of feature 4 is 1.2526

Problem 1, Part d

The variance of feature 1 is 0.6993
The variance of feature 2 is 0.1916
The variance of feature 3 is 3.0976
The variance of feature 4 is 0.5797

The standard deviation of feature 1 is 0.8362
The standard deviation of feature 2 is 0.4378
The standard deviation of feature 3 is 1.7600
The standard deviation of feature 4 is 0.7613

Problem 1, Part e

Here is the code for part E. The initial parts of the code covers previous parts of this problem.

```
iris = load('data/iris.txt');
y = iris(:,end);
X = iris(:,1:end-1);

%part A
numFeatures = size(X,2);
numDataPoints = size(X,1);

%put features into vectors
feature1 = X(:,1);
feature2 = X(:,2);
feature3 = X(:,3);
feature4 = X(:,4);

%part B
figure
```

```
subplot(2,2,1)
hist(feature1)
title('Histogram for Feature 1')
subplot(2,2,2)
hist(feature2)
title('Histogram for Feature 2')
subplot(2,2,3)
hist(feature3)
title('Histogram for Feature 3')
subplot(2,2,4)
hist(feature4)
title('Histogram for Feature 4')

%part C
mean1 = mean(feature1);
mean2 = mean(feature2);
mean3 = mean(feature3);
mean4 = mean(feature4);


%part D

%compute the variance
var1 = var(feature1);
var2 = var(feature2);
var3 = var(feature3);
var4 = var(feature4);

%compute the standard deviation
std1 = std(feature1);
std2 = std(feature2);
std3 = std(feature3);
std4 = std(feature4);

%part E
% Normalizes the data
normalize1 = (feature1-mean1)/std1;
normalize2 = (feature2-mean2)/std2;
normalize3 = (feature3-mean3)/std3;
normalize4 = (feature4-mean4)/std4;
```

Problem 1, Part f

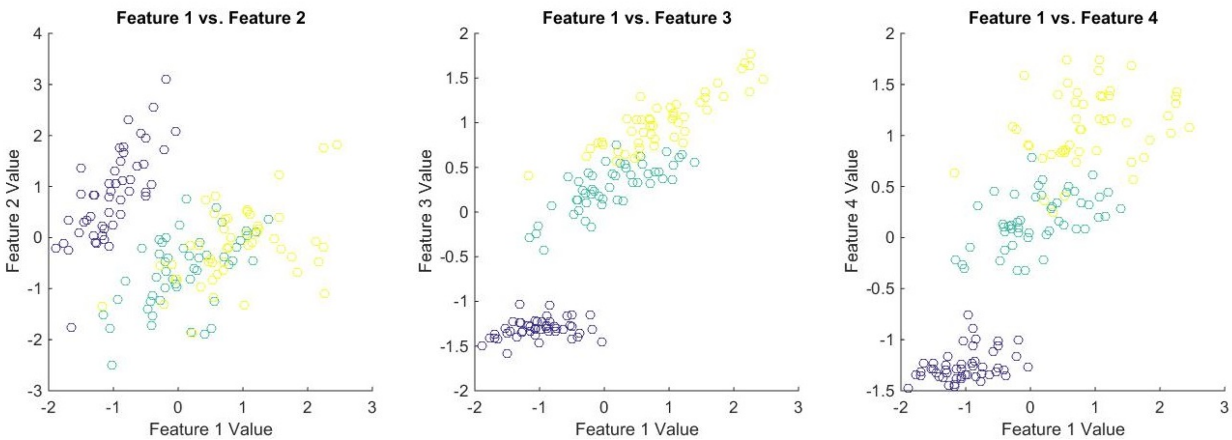

Figure 2: The scatter plots for Problem 1f

This is the code to make those plots. It is a continuation of the code posted for part e.

```
size = 30;
figure
subplot(1,3,1)
scatter(normalize1,normalize2,size,y);
title('Feature 1 vs. Feature 2');
xlabel('Feature 1 Value');
ylabel('Feature 2 Value');
subplot(1,3,2)
scatter(normalize1,normalize3,size,y);
title('Feature 1 vs. Feature 3');
xlabel('Feature 1 Value');
ylabel('Feature 3 Value');
subplot(1,3,3)
scatter(normalize1,normalize4,size,y);
title('Feature 1 vs. Feature 4');
xlabel('Feature 1 Value');
ylabel('Feature 4 Value');
```

Problem 2

Problem 2, Part a

These are the plots for part a

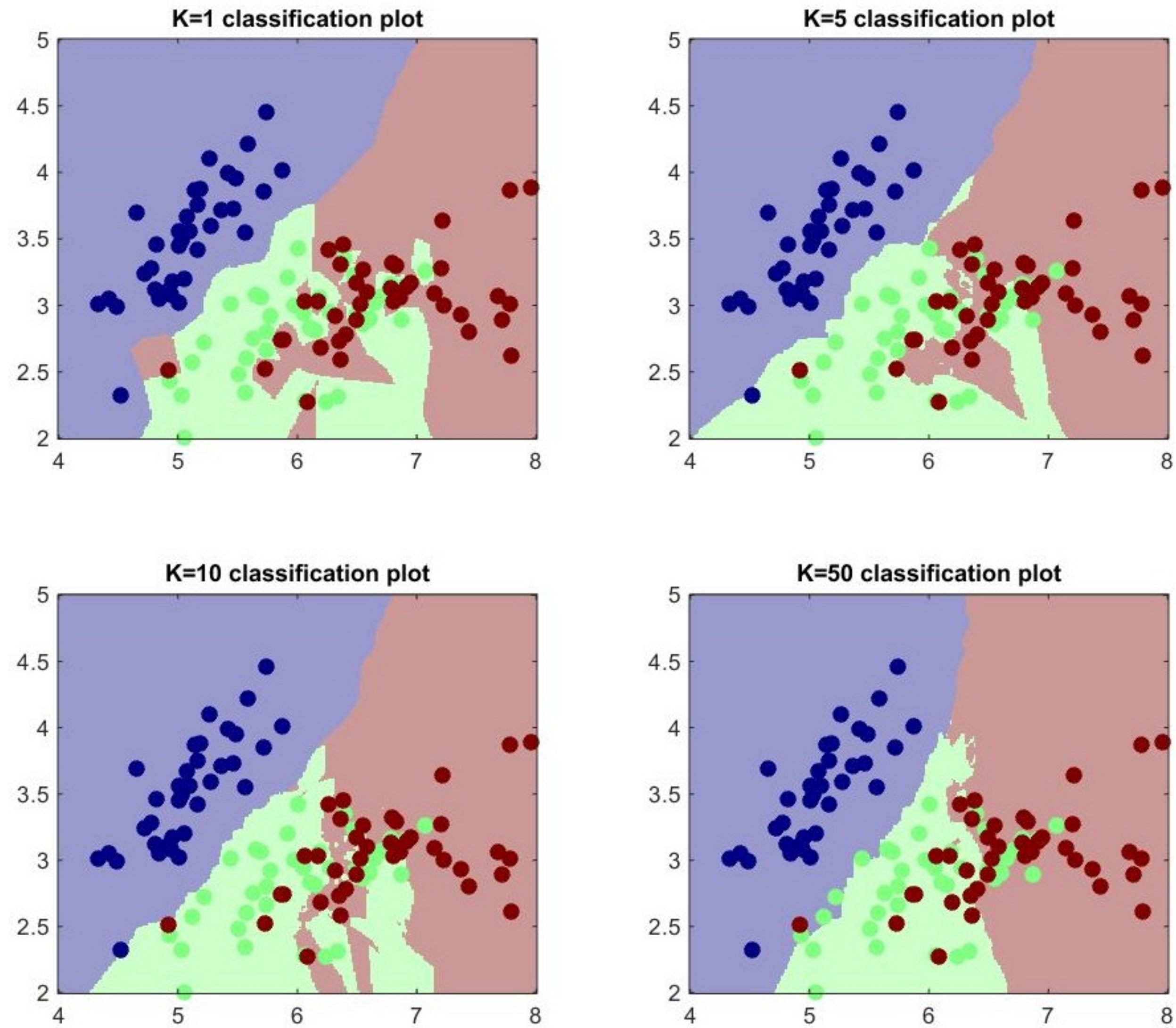

Figure 3: The scatter plots for Problem 2a

Here is the code I used to generate those plots

```
%InitialPart
iris=load('data/iris.txt');
y=iris(:,end);
X=iris(:,1:end-1);

[X y] = shuffleData(X,y); % shuffle data randomly
[Xtr Xte Ytr Yte] = splitData(X,y, .75); % split data into 75/25 train/test

%gets the first 2 features
XtrFirstTwo = Xtr(:,1:2);
XteFirstTwo = Xte(:,1:2);

%partA
figure
Kvals = [1,5,10,50];
for i=1:4
   K = Kvals(i);

   %train the classifier
   knn = knnClassify( XtrFirstTwo, Ytr, K );

   % make 2D classification plot
   subplot(2,2,i)
   plotClassify2D( knn, XtrFirstTwo, Ytr );
   title(strcat('K=',num2str(K),' classification plot'));
end
```

Problem 2, Part b

Here is the training error (in Red) and the test error (in green) as the value of K increases.
Based on this plot, I would recommend $K = 50$

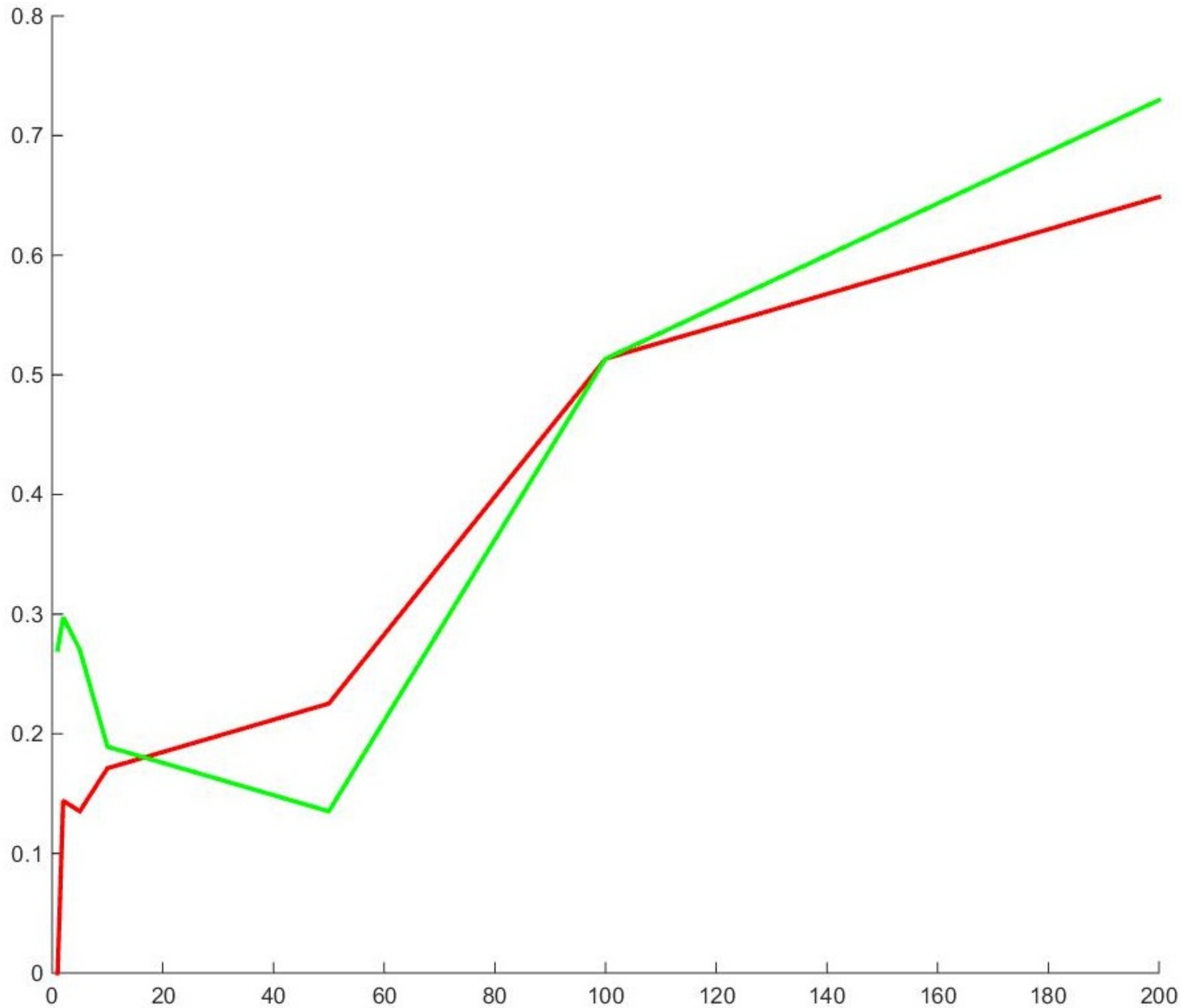

Figure 4: The semilog plot for Problem 2

This is the rest of the code for problem 2, the part which was used to make the plots for part b.

```
%part B
Kvals=[1,2,5,10,50,100,200];
errTrain=zeros(1,length(Kvals));
errTest = zeros(1,length(Kvals));
for i=1:length(Kvals)
    K = Kvals(i);
    learner = knnClassify( XtrFirstTwo, Ytr, K );
    YhatTr = predict(learner,XtrFirstTwo);
    errTrain(i)=length(find(YhatTr~=Ytr))/length(Ytr);

    YhatTe = predict(learner,XteFirstTwo);
    errTest(i)=length(find(YhatTe~=Yte))/length(Yte);
end;
figure
hold on
semilogx(Kvals,errTrain,'-','LineWidth',2,'Color','red');
semilogx(Kvals,errTest,'-','LineWidth',2,'Color','green');
hold off
```

Problem 3

Problem 3, Part a

The class probabilities are as follows:
$p(y = 1) = 0.4$
$p(y = -1) = 0.6$

The feature probabilites are as follows:

i	$p(x_i = 0\|y = 1)$	$p(x_i = 1\|y = 1)$	$p(x_i = 0\|y = 1)$	$p(x_i = 1\|y = 1)$
1	0.25	0.75	0.5	0.5
2	1	0	0.1667	0.8333
3	0.25	0.75	0.3333	0.6667
4	0.5	0.5	0.1667	0.8333
5	0.75	0.25	0.6667	0.3333

Here is the code I used to get those values

```
xyData=[
0 0 1 1 0 -1;
1 1 0 1 0 -1;
0 1 1 1 1 -1;
1 1 1 1 0 -1;
0 1 0 0 0 -1;
1 0 1 1 1 1;
0 0 1 0 0 1;
1 0 0 0 0 1;
1 0 1 1 0 1;
1 1 1 1 1 -1];

X=xyData(:,1:5);
y=xyData(:,6);

%Part A

%this gets p(y==1) and p(y==-1)
indicesY1 = find(y==1);
indicesYminus1 = find(y==-1);
probY1 = length(indicesY1)/length(y);
probYminus1 = 1-probY1;
```

```
%in order to get class probabilities, we go over entries where
%       y==1 and y==-1
%
% We will get the following matrices
%   probXwhereY1:
%       row i has probabilities for x_i
%       column 1 is p(x_i=0|y=1)
%       column 2 is p(x_i=1|y=1)
%   probXwhereYminus1:
%       row i has probabilities for x_i
%       column 1 is p(x_i=0|y=-1)
%       column 2 is p(x_i=1|y=-1)
probXwhereY1 = zeros(5,2);
probXwhereYminus1 = zeros(5,2);
for i = 1:5
    Xi = X(:,i);

    XiwhereY1 = Xi(indicesY1); %x_i values for entries where y=1
    probXwhereY1(i,1) = length(find(XiwhereY1==0))/length(XiwhereY1);
    probXwhereY1(i,2) = 1-probXwhereY1(i,1);

    XiwhereYminus1 = Xi(indicesYminus1); %x_i values for entries where y=-1
    probXwhereYminus1(i,1) = length(find(XiwhereYminus1==0))/length(XiwhereYminus1)
    probXwhereYminus1(i,2) = 1-probXwhereYminus1(i,1);
end
```

Problem 3, Part b

With $x = (0, 0, 0, 0, 0)$ it holds that
$p(y = 1) = 0.8351$
$p(y = -1) = 0.1649$
Thus $y = 1$ is the predicted classification.

With $x = (1, 1, 0, 1, 0)$ it holds that
$p(y = 1) = 0$
$p(y = -1) = 1$
Thus $y = -1$ is the predicted classification.

Problem 3, Part c

The values showed in my answer for part b are the normalized values, thus
$p(y = 1) = 0$ for $x = (1, 1, 0, 1, 0)$ as noted previously.

Problem 3, Part b and c code

Here is the code that I used to compute the values in part b and c.
This is a continuation of the code from part a.

```
%part B
%Using Naive Bayes, p(y|x) = p(x|y)p(y)/p(x)
%   p(x) = sum_y( p(x|y)p(y) )
%   p(x|y)=p(x_1|y)p(x_2|y)...p(x_5|y)

%We need to find the y prob of x=(0 0 0 0 0)
xTest1 = [0 0 0 0 0];
[probY1withXtest1,probYminus1withXtest1] = prob3Classifier(probXwhereY1,probXwhereY

bestYclassification1 = 1;
if(probY1withXtest1<probYminus1withXtest1)
    bestYclassification1 = -1;
end

xTest2 = [1 1 0 1 0];
[probY1withXtest2,probYminus1withXtest2] = prob3Classifier(probXwhereY1,probXwhereY

bestYclassification2 = 1;
if(probY1withXtest2<probYminus1withXtest2)
    bestYclassification2 = -1;
end

%part C
probY1withXtest2
```

I wrote a function called *prob3Classifier* that used the classifier made in part a and computed the posterior probabilities. Here is the code for it:

```
function [ probY1withXtest1,probYminus1withXtest1 ] = prob3Classifier(
probXwhereY1,probXwhereYminus1, probY1, probYminus1,xTest1 )
%PROB3CLASSIFIER says the probabilities of the classifications of the test
%               data
% Input is the following matrices:
%   probXwhereY1:
%       row i has probabilities for x_i
%       column 1 is p(x_i=0|y=1)
%       column 2 is p(x_i=1|y=1)
%   probXwhereYminus1:
%       row i has probabilities for x_i
%       column 1 is p(x_i=0|y=-1)
%       column 2 is p(x_i=1|y=-1)
%
%probY1withXtest1 is p(y=1|x) where x is the test vector
%probYminus1withXtest1 is p(y=-1|x) where x is the test vector

probXtestWhereY1 = zeros(1,5);
probXtestWhereYminus1 = zeros(1,5);
for i = 1:5
   probXtestWhereY1(i) = probXwhereY1(i,xTest1(i)+1);
   probXtestWhereYminus1(i) = probXwhereYminus1(i,xTest1(i)+1);
end
probXtestWithY1 = prod(probXtestWhereY1)*probY1;
probXtestWithYminus1 = prod(probXtestWhereYminus1)*probYminus1;

%finally here is p(y=1|x)
probY1withXtest1 = probXtestWithY1/(probXtestWithY1+probXtestWithYminus1);

%here is p(y=-1|x)
probYminus1withXtest1 = probXtestWithYminus1/(probXtestWithY1+probXtestWithYminus1)

end
```

Problem 3, part d

We have 5 features with 2 values each, so if we want a Bayes classifier then there are $2^5 = 32$ feature vectors which we would need to find classification probabilities for. We only have 10 observations meaning that not all the feature vectors have a classification probability. We thus would not have a trained Bayes classifier and there could be input values that it could not compute a probability for.

Another issue with using a Bayes classifier is that each row is unique so the probability for each class given a feature vector would simply be 0 or 1 which is likely not accurate.

Problem 4

Problem 4, Part a

The mean vectors for the first two features are as follows:

For class $y = 0$ it is the following: $(5.0094, 3.4460)$
For class $y = 1$ it is the following: $(5.9934, 2.7838)$
For class $y = 2$ it is the following: $(6.5927, 3.0392)$

The covariance matrices for the first two features are as follows:

For class $y = 0$ it is the following:

0.1239	0.1038
0.1038	0.1542

For class $y = 1$ it is the following:

0.2781	0.0921
0.0921	0.1008

For class $y = 2$ it is the following:

0.3956	0.1377
0.1377	0.1277

The code to get these values is as follows:

```
iris=load('data/iris.txt');
y=iris(:,end);
X=iris(:,1:end-1);

[X y] = shuffleData(X,y); % shuffle data randomly
[Xtr Xte Ytr Yte] = splitData(X,y, .75); % split data into 75/25 train/test

%gets the first 2 features
XtrFirstTwo = Xtr(:,1:2);
XteFirstTwo = Xte(:,1:2);

%Part A

%classes are 0,1,2 for Y
%
%The indices giving y=0,1, and 2 for training data
Ytr0Indices = find(Ytr==0);
Ytr1Indices = find(Ytr==1);
Ytr2Indices = find(Ytr==2);

XtrClass0 = XtrFirstTwo(Ytr0Indices,:);
XtrClass1 = XtrFirstTwo(Ytr1Indices,:);
XtrClass2 = XtrFirstTwo(Ytr2Indices,:);

XtrMeanClass0 = mean(XtrClass0);
XtrMeanClass1 = mean(XtrClass1);
XtrMeanClass2 = mean(XtrClass2);

XtrCovClass0 = cov(XtrClass0);
XtrCovClass1 = cov(XtrClass1);
XtrCovClass2 = cov(XtrClass2);
```

Problem 4, Part b

Here is the scatter plot of the feature data and their classes
TODO: Label what class corresponds to what color

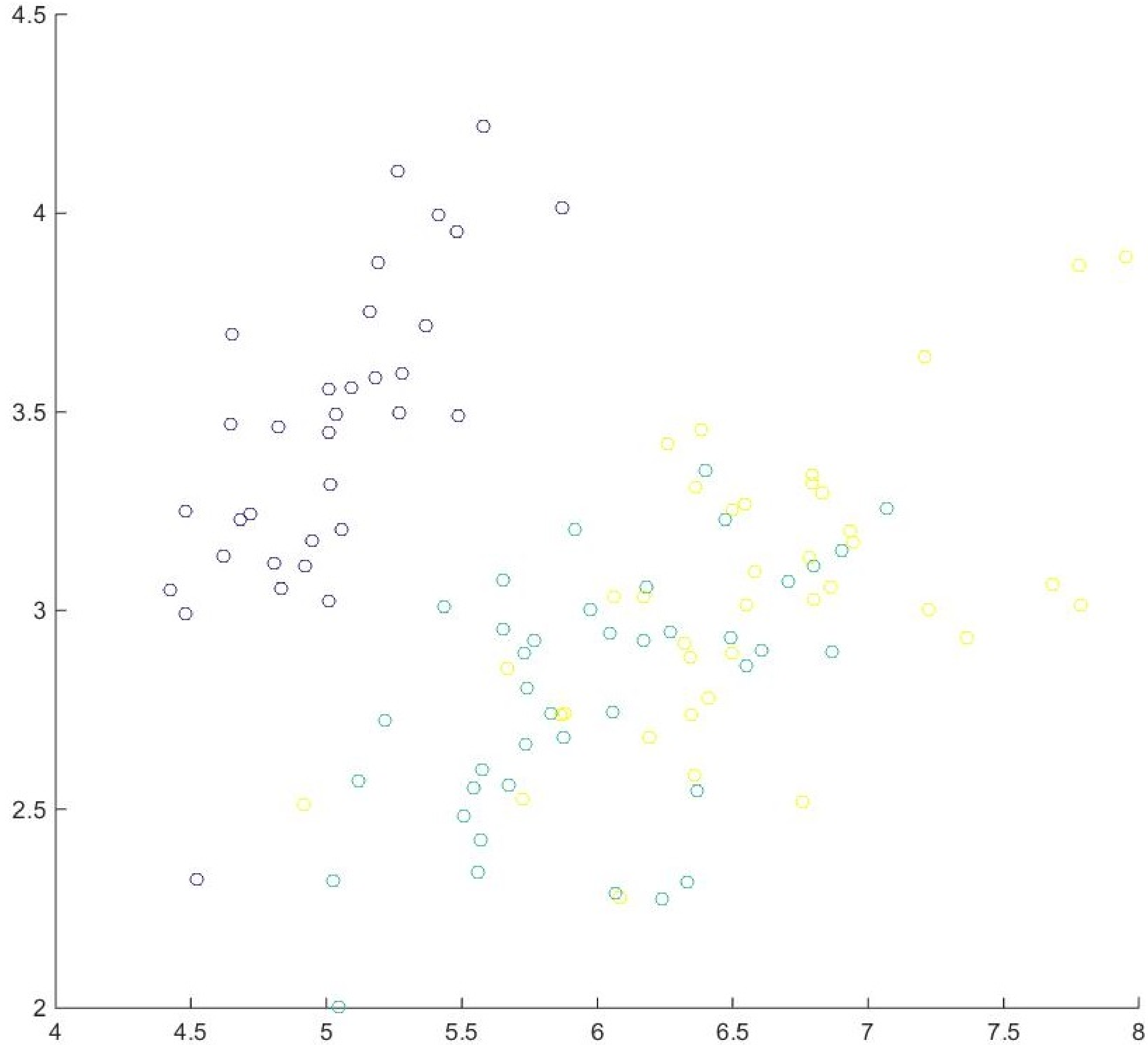

Figure 5: Scatter Plot of features and class labels

Problem 4, Part c

The contours of the Gaussian plot are shown below

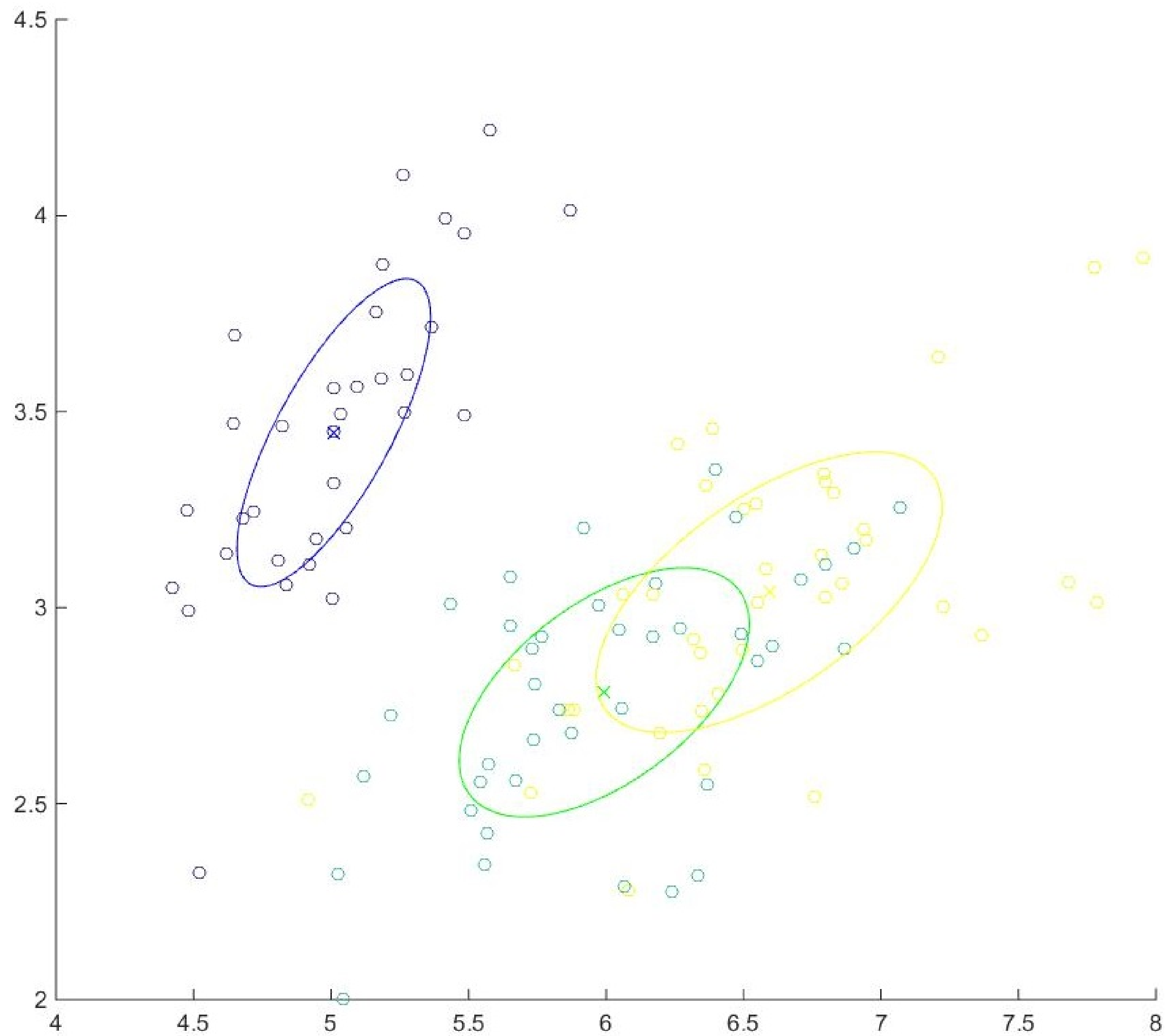

Figure 6: Scatter Plot of features and class labels with contours for Gaussian

Problem 4, Part d code

Here is the plot of the boundaries and contours of the Gaussian classifier

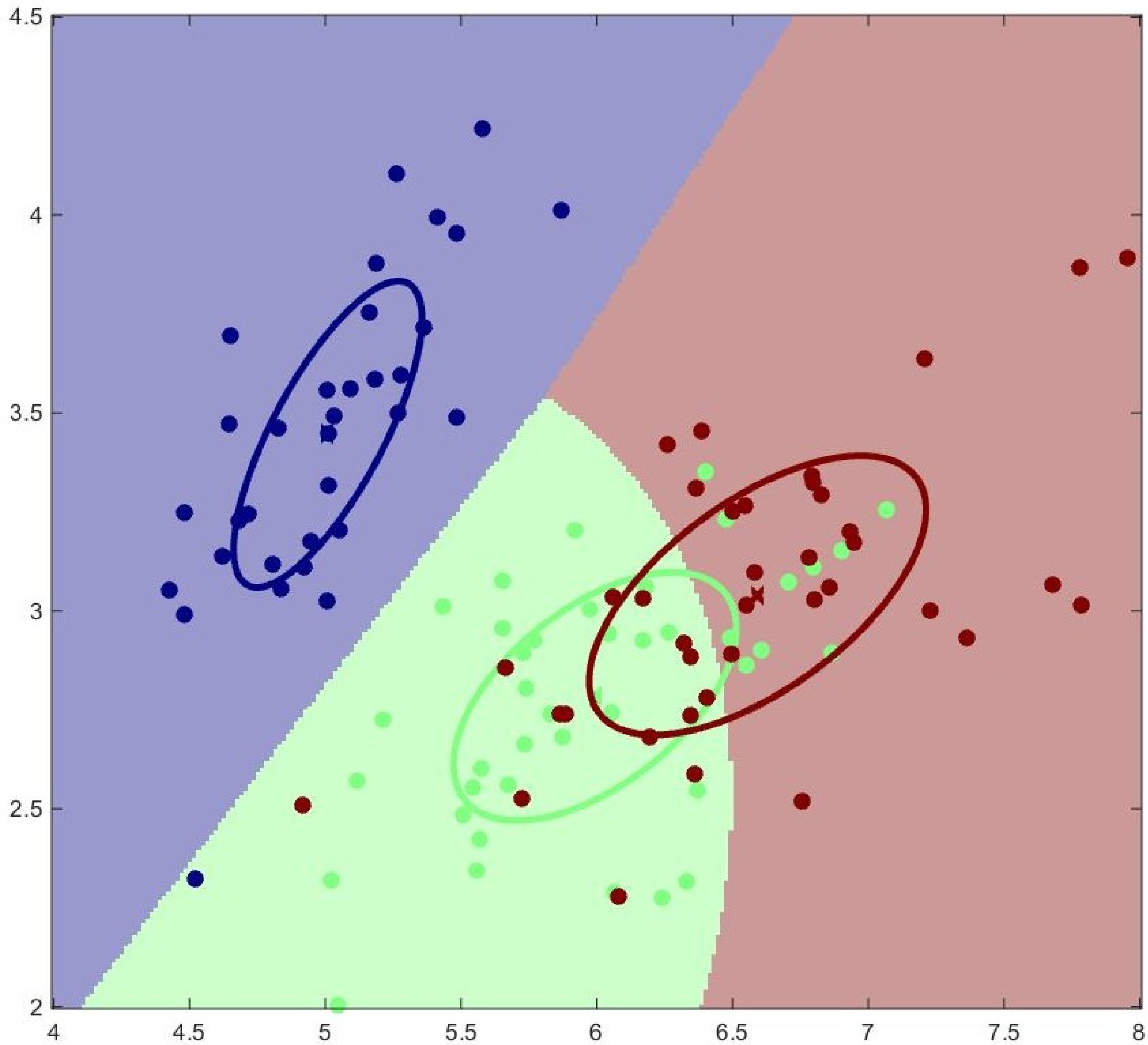

Figure 7: Scatter Plot of features and class labels with boundaries of Gaussian classifier

Problem 4, Part b,c,d code

Here is the code to make the plots for part b,c,and d

```
%%
%Part B
size = 30;
figure
scatter(XtrFirstTwo(:,1),XtrFirstTwo(:,2),size,Ytr);

%Part C
figure
hold on
scatter(XtrFirstTwo(:,1),XtrFirstTwo(:,2),size,Ytr);
plotGauss2D(XtrMeanClass0,XtrCovClass0,'b-');
plotGauss2D(XtrMeanClass1,XtrCovClass1,'g-');
plotGauss2D(XtrMeanClass2,XtrCovClass2,'y-');
hold off

%Part D
bc = gaussBayesClassify( XtrFirstTwo, Ytr );
figure
plotClassify2D(bc, XtrFirstTwo, Ytr);
```

Problem 4, Part e

The training error rate ends up being 0.2252
The test error rate ends up being 0.1081

This is the code I used to calculate that. It is a continuation of the code from previous parts of this problem:

```
%Part E
yTrHat = predict(bc,XtrFirstTwo);
trainError = length(find(yTrHat~=Ytr))/length(Ytr);

yTeHat = predict(bc,XteFirstTwo);
testError = length(find(yTeHat~=Yte))/length(Yte);
```

Problem 4, Part f

For this problem, the training error rate ends up being 0.0270
The test error rate ends up being 0

Here is the code I used to calculate that
It only needs the code at the beginning that obtained *Xtr* and *Ytr*

```
XtrAllClass0 = Xtr(Ytr0Indices,:);
XtrAllClass1 = Xtr(Ytr1Indices,:);
XtrAllClass2 = Xtr(Ytr2Indices,:);

XtrAllMeanClass0 = mean(XtrAllClass0);
XtrAllMeanClass1 = mean(XtrAllClass1);
XtrAllMeanClass2 = mean(XtrAllClass2);

XtrAllCovClass0 = cov(XtrAllClass0);
XtrAllCovClass1 = cov(XtrAllClass1);
XtrAllCovClass2 = cov(XtrAllClass2);

bcTrAll = gaussBayesClassify( Xtr, Ytr );
yTrAllHat = predict(bcTrAll,Xtr);
trainErrorAll = length(find(yTrAllHat~=Ytr))/length(Ytr);

yTeAllHat = predict(bcTrAll,Xte);
testErrorAll = length(find(yTeAllHat~=Yte))/length(Yte);
```