EECE 5644: Assignment 1

Jeffrey Chen

October 23, 2024

---

1 Question 1

1.1 Part A

The samples of this question is attached as samples_1a.csv.

Figure 1 here shows the distribution of the data and their true labels

1. To convert this to a likelihood-ratio test, knowing that

$$P(L=0) = 0.35$$
$$P(L=1) = 0.65$$
$$> if D(x) = 1$$
$$< if D(x) = 0$$

we would need to start out with the given equation:

$$\frac{P(x|L=1)}{P(x|L=0)} \overset{?}{>=} \gamma$$

we also know that

$$\gamma = \frac{\lambda_{10} - \lambda_{00}}{\lambda_{01} - \lambda 11} \cdot \frac{P(x|L=0)}{P(x|L=1)}$$

we can plug this in to get the likelihood-ratio test:

$$\frac{P(x|L=1)}{P(x|L=0)} \underset{D(x)=0}{\overset{D(x)=1}{\gtrless}} \frac{\lambda_{10} - \lambda_{00}}{\lambda_{01} - \lambda_{11}} \cdot \frac{P(x|L=0)}{P(x|L=1)}$$

we can also replace it with the given class conditional pdfs to get the final equation:

$$\frac{g(x|m_1, C1)}{g(x|m_0, C0)} \underset{D(x)=0}{\overset{D(x)=1}{\gtrless}} \frac{\lambda_{10} - \lambda_{00}}{\lambda_{01} - \lambda_{11}} \cdot \frac{P(x|L=0)}{P(x|L=1)}$$

solving for γ, we get, which we will use in part 2 and part 3:

$$\gamma > 0.53846$$

2. Using this classifier and applying it to the 10k sample generated, by varying the threshold of γ to compute each pair of $P(D = 1|L = 1; \gamma)$ and $P(D = 1|L = 0; \gamma)$ which produced the ROC curve shown in figure 2.

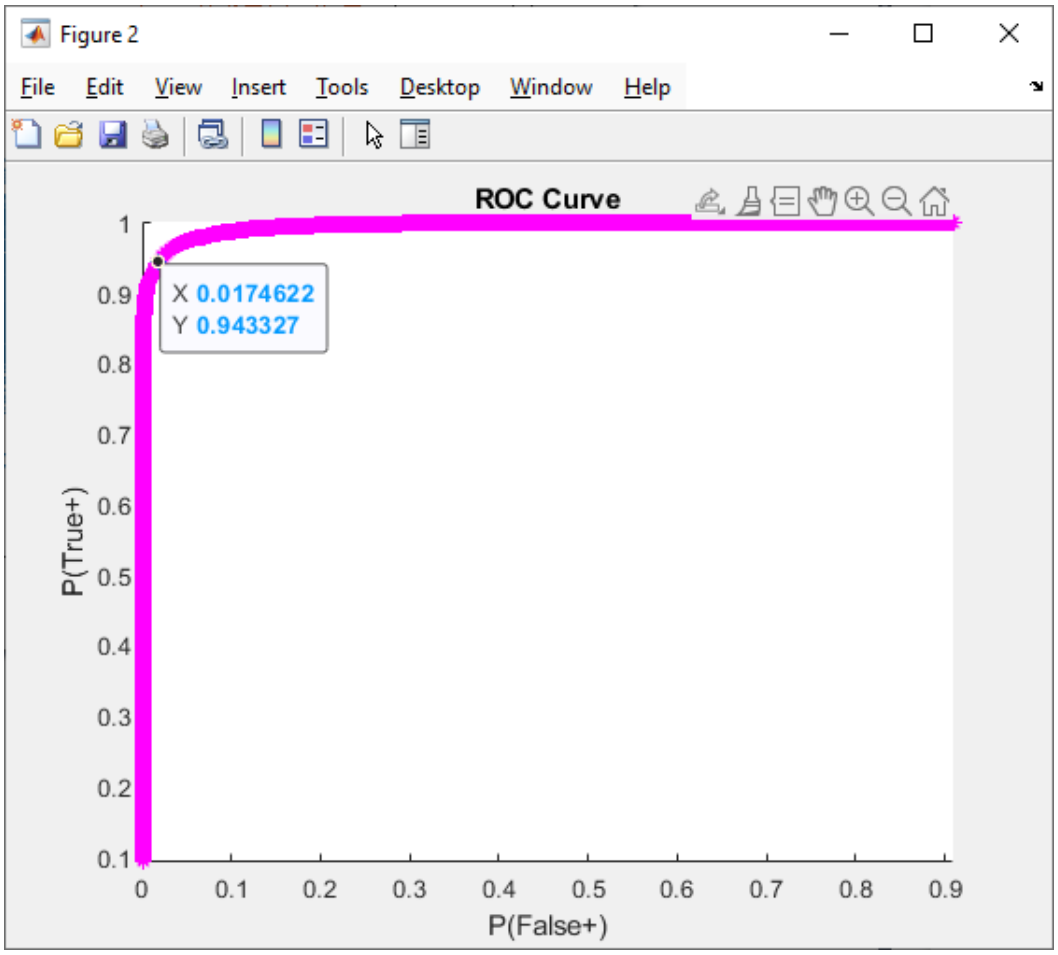

Figure 2 shows the ROC curve of the following distribution. The blue x indicates the experimental threshold value of the minimum probability of error.

3. Knowing that

$$P(error, \gamma) = P(D = 1|L = 0; \gamma)P(L = 0) + P(D = 0|L = 1; \gamma)P(L = 1)$$

we can use this to find the P(error) of both the theoretical and the experimental results. From this we get:

$$\gamma_t = 0.5385, P(error)_t = 0.0292$$
$$\gamma_e = 0.5235, P(error)_e = 0.0348$$

Looking at both the theoretical and experimental values, it seems that while they aren't the same, they seem close enough where we can conclude they are accurate. More samples can be used to get a more accurate theoretical and experimental values but this may lead to overfitting.

1.2 Part B

For this part, the same steps in part A were used but all we did was assume a Naive Bayesian Classifier. Using this approach, the class conditional pdfs would only have diagonal values while the other values were zeros, similar to an identity matrix. Below show the matrices we used:

$$C_0 = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \quad C_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

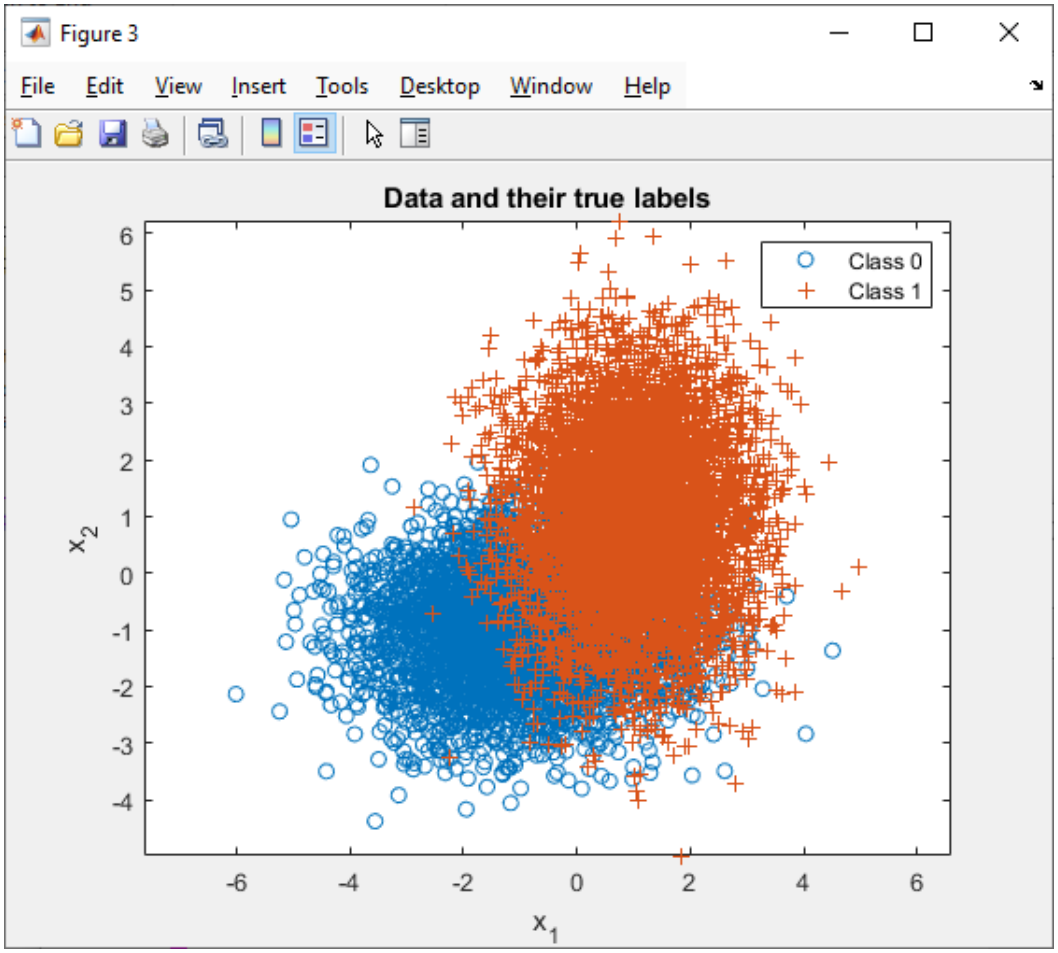

Figure 3 here shows the distribution of the Naive Bayesian Classifier

1. Part A remains unchanged.

$$\frac{g(x|m_1, C1)}{g(x|m_0, C0)} \underset{D(x)=0}{\overset{D(x)=1}{\gtrless}} \frac{\lambda_{10} - \lambda_{00}}{\lambda_{01} - \lambda_{11}} \cdot \frac{P(x|L=0)}{P(x|L=1)}$$

$$\gamma > 0.53846$$

2. Using this Naive Bayesian Classifier approach, there seem to be more false positives than the Gaussian approach. Looking at the ROC, while only slightly, there seems to be more false positives in the graph. But looking clearly at the most optimized dot, we can see there is about a 0.013 increase in the probability of a false positive.

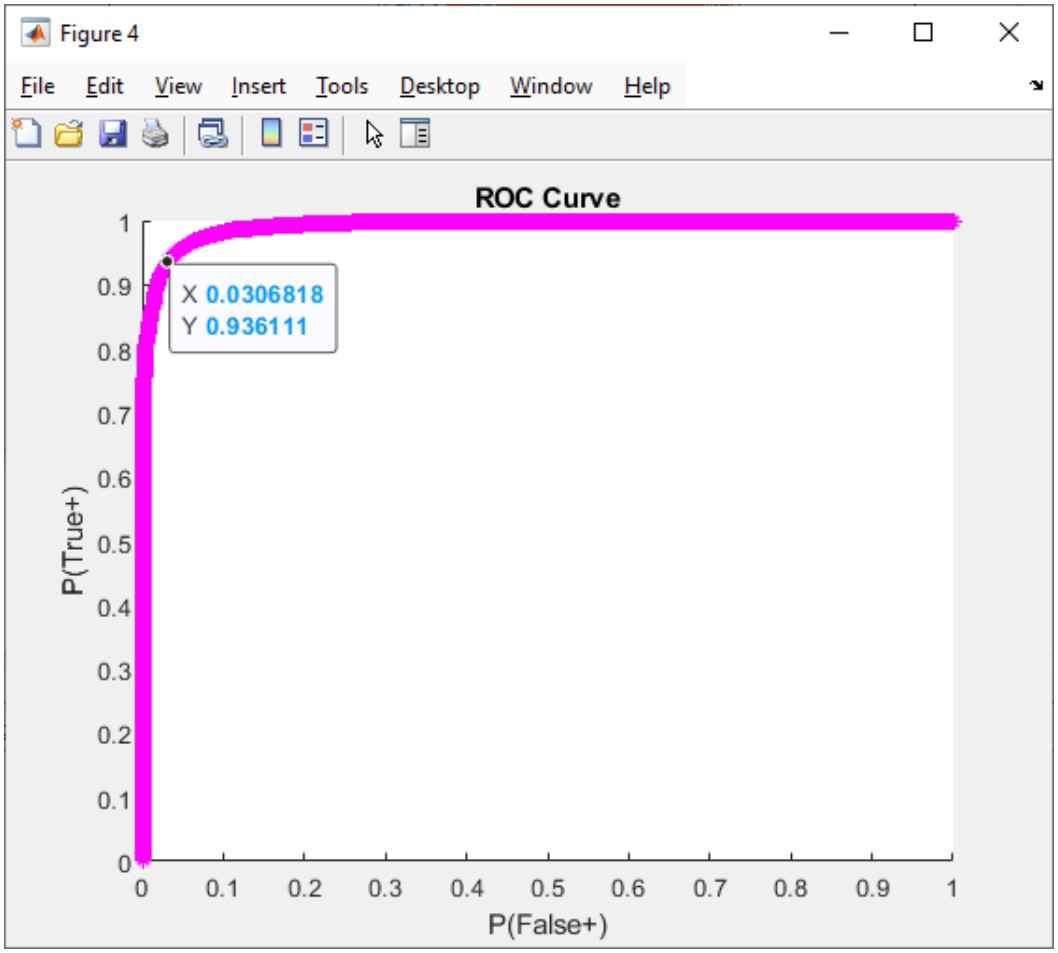

Figure 4 here shows the ROC for the Naive Bayesian Classifier approach.

3. Using the same function, in part a:

$$P(error, \gamma) = P(D = 1|L = 0; \gamma)P(L = 0) + P(D = 0|L = 1; \gamma)P(L = 1)$$

we get that:

$$\gamma_{tn} = 0.5385, P(error)_{tn} = 0.0406$$

$$\gamma_{en} = 0.5432, P(error)_{en} = 0.0423$$

While the γ is relatively the same, there is a higher error rate compared to the original Gaussian method.

1.3 Part C

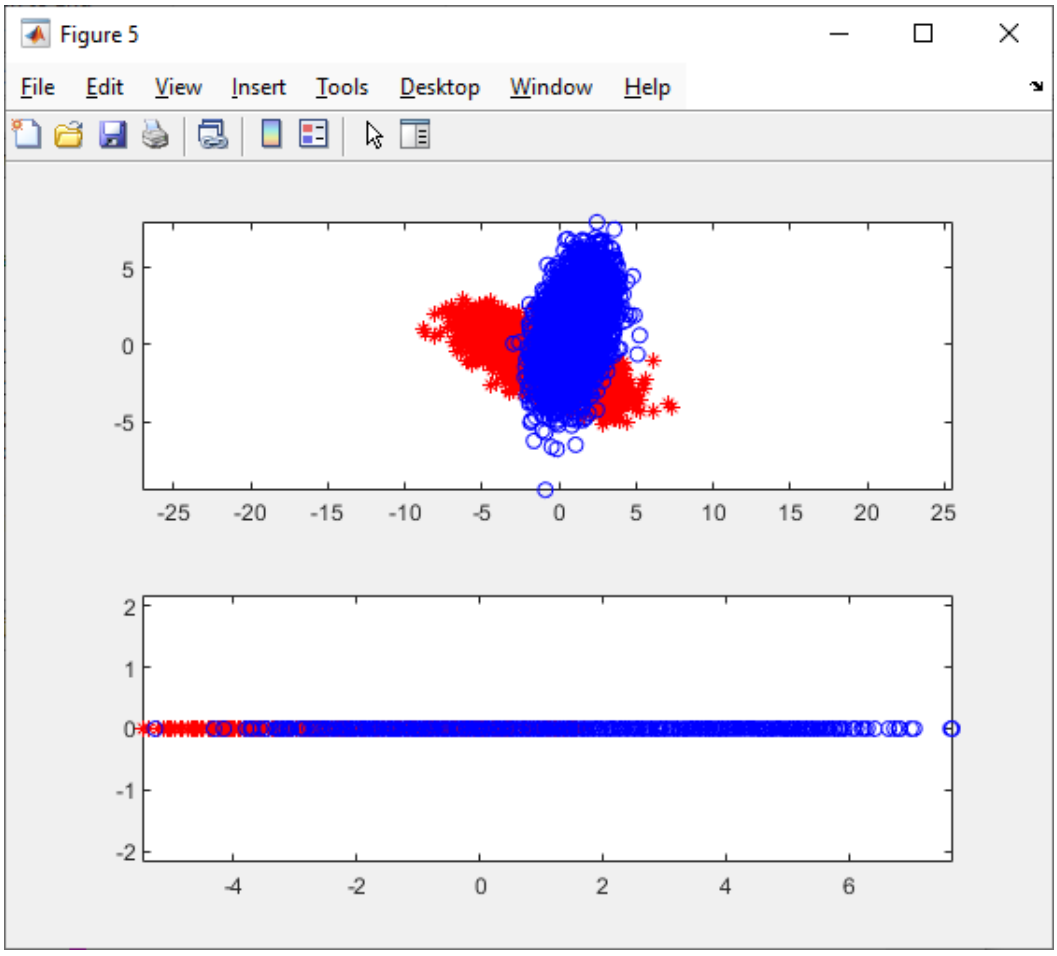

Figure 5 here shows the data before and after linear projection

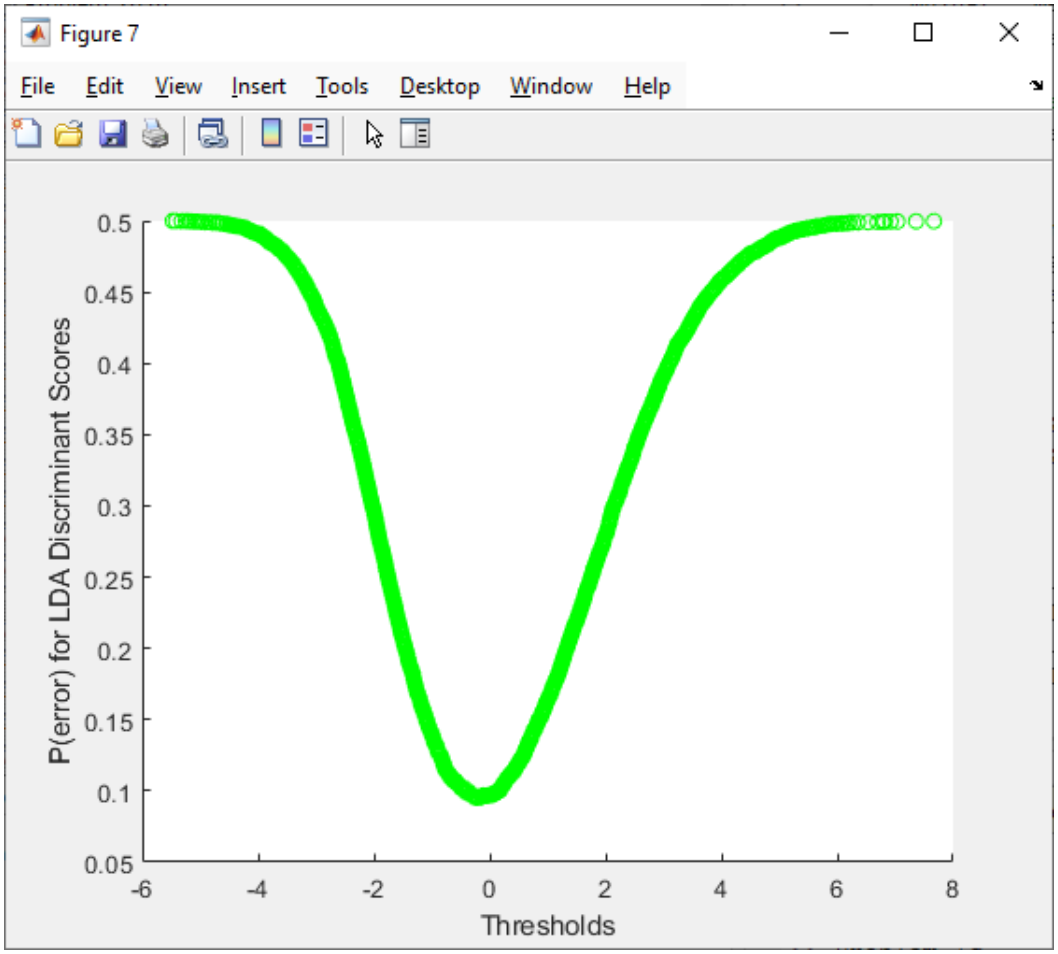

Figure 7 here shows the $P(error)$ for the LDA Discriminant Scores

1. Estimating the class conditional pdf mean and the covariance matrices, we get:

$$\hat{\mu_0} = \begin{bmatrix} -1.0902 \\ -0.8982 \\ -1.1050 \\ -0.9974 \end{bmatrix} \quad \hat{\sigma_0} = \begin{bmatrix} 3.8644 & -1.4631 & 0.9490 & -0.9760 \\ -1.4631 & 1.3751 & -0.8361 & 0.4186 \\ 0.9490 & -0.8361 & 0.7980 & -0.4090 \\ -0.9760 & 0.4186 & -0.4090 & 4.4048 \end{bmatrix}$$

$$\hat{\mu_1} = \begin{bmatrix} 1.1432 \\ 0.9278 \\ 0.9199 \\ 0.7041 \end{bmatrix} \quad \hat{\sigma_1} = \begin{bmatrix} 1.2426 & 0.7912 & -0.4170 & 0.6311 \\ 0.7912 & 4.0080 & 0.9506 & -0.0762 \\ -0.4170 & 0.9506 & 1.2110 & -0.2652 \\ 0.6311 & -0.0762 & -0.2652 & 9.1634 \end{bmatrix}$$

For the w_{LDA}, we get

$$w_{LDA} = \begin{bmatrix} 0.3634 \\ 0.3266 \\ 0.8581 \\ 0.1581 \end{bmatrix}$$

2. Below in figure 6, we can see the ROC curve. We can see that $P(False+) = 0.0678$, which was higher than both of the classifiers. Given this, we can say that the Fischer LDA classifier is more prone to error compared to the other classifiers. We can also see in this ROC curve that there does seem to have a larger area of error.

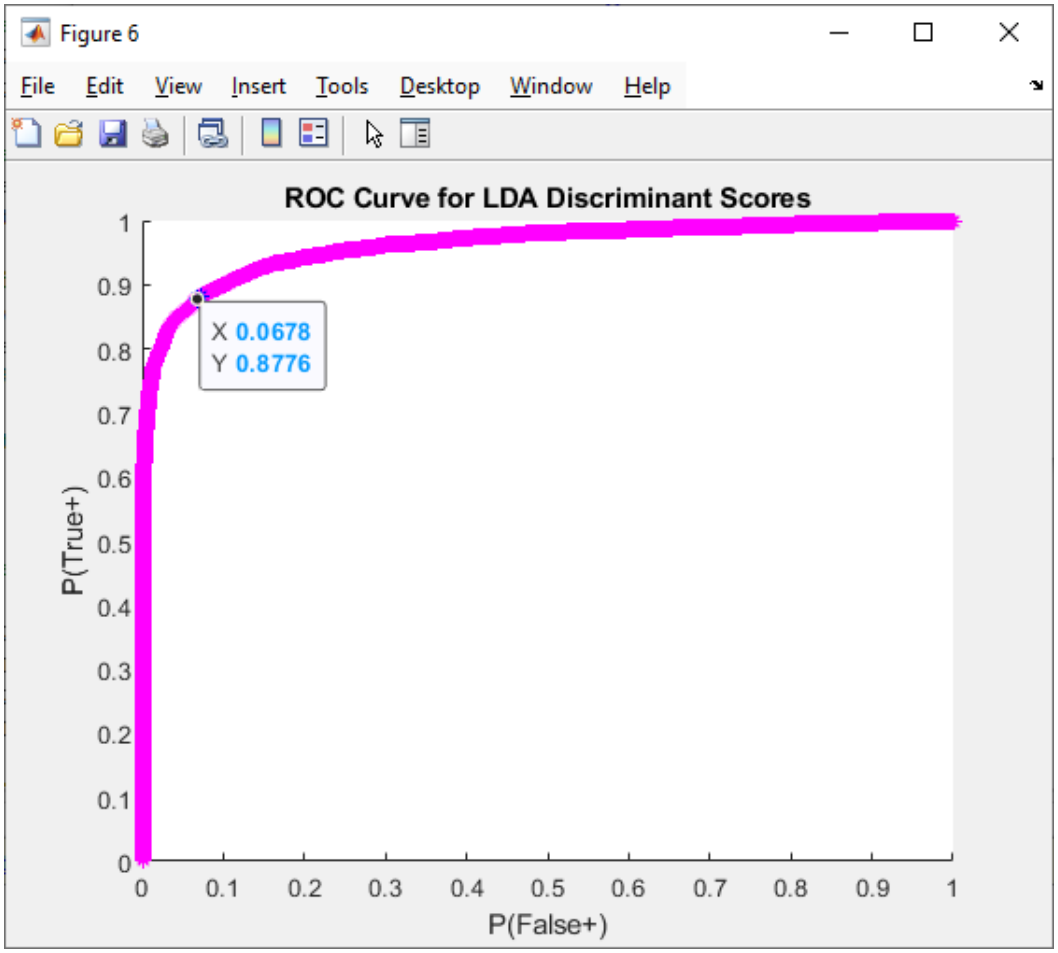

Figure 6 here shows the ROC curve of the Fisher LDA classifier

2 Question 2

2.1 Part A

1. The samples are generated in samples_2a.csv
2. In a minimum probability of error, a minimum expected risk classifier can be used. In this scenario, $P(D = i|L = j)$ can be defined using the confusion matrix given.

$$P(D = i|L = j) = \begin{bmatrix} 0.9509 & 0.0212 & 0.0517 \\ 0.0267 & 0.9560 & 0 \\ 0.0224 & 0.0228 & 0.9483 \end{bmatrix}$$

The rows of the matrix represent $D = i$ while the columns of the matrix represent $L = j$

3. Below in figure 8 is a visualization of the data.

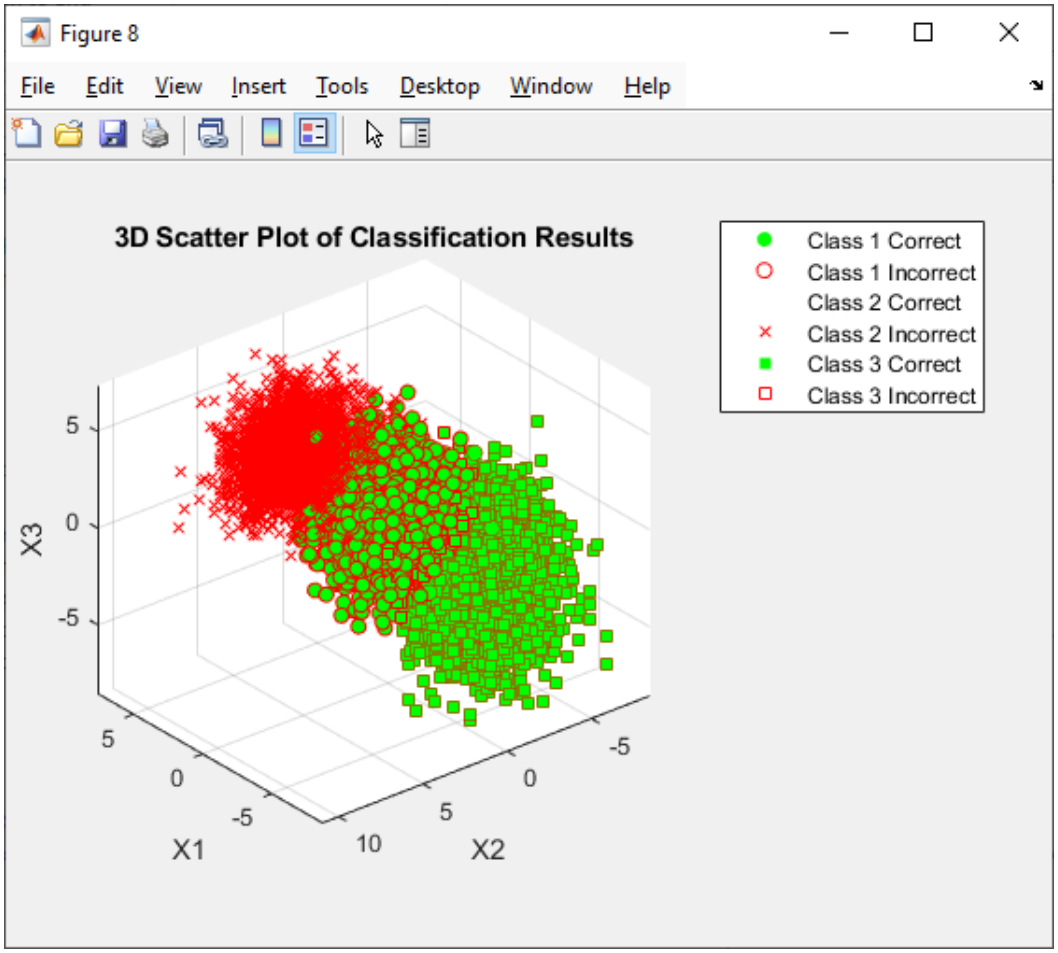

Figure 8 here shows the 3D plot representing the ERM samples. There are no Class 2 correct samples, which are also shown in the confusion matrix.

2.2 Part B

1. Repeating with a Λ_{10} loss matrix, we can see in the confusion matrix that there is a heavier emphasis on $L = 3$ values. Also looking below at figure 9, we can kind of make out less incorrect as we see more green. Especially at $D = 3 \& L = 3$ from the confusion matrix, we can see it getting closer to 1.

$$P(D = i|L = j) = \begin{bmatrix} 0.7846 & 0.0033 & 0.0141 \\ 0.0807 & 0.9948 & 0 \\ 0.1347 & 0.0020 & 0.9859 \end{bmatrix}$$

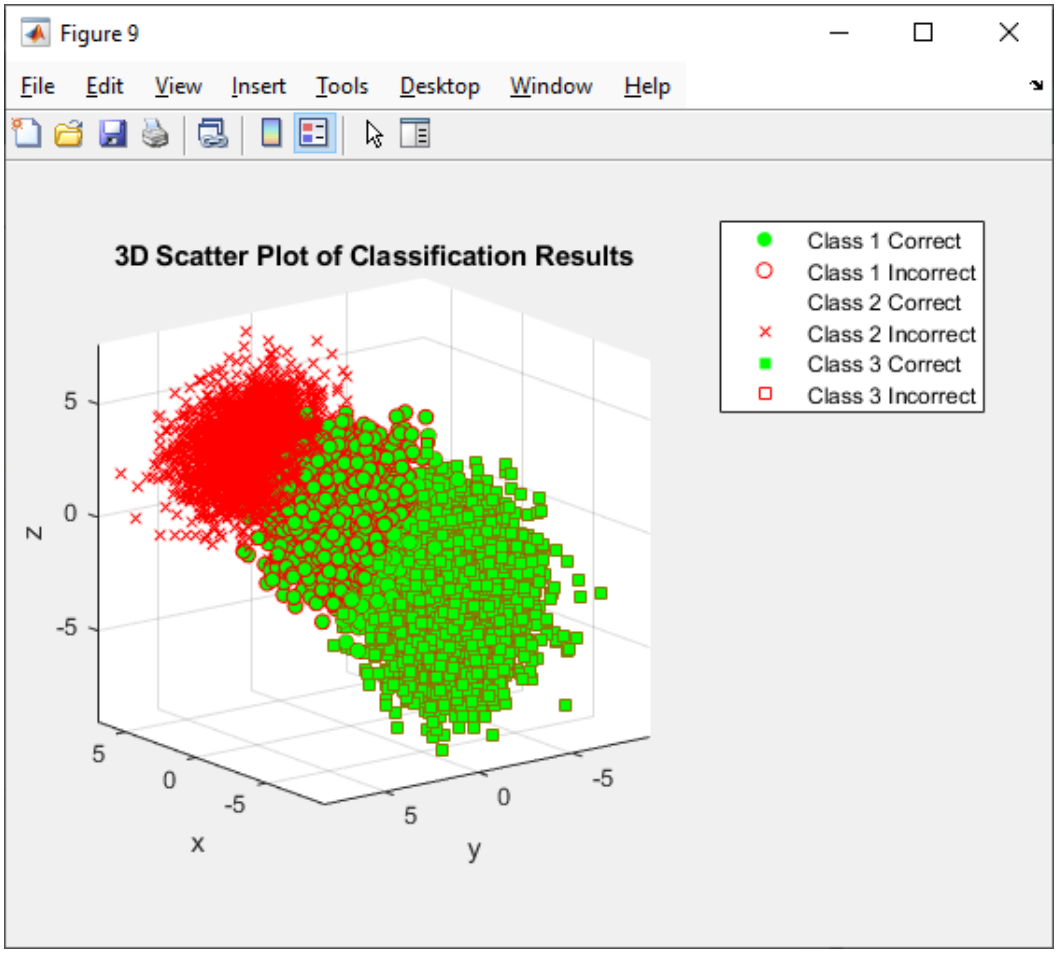

Figure 9 here shows the 3D plot representing the ERM samples with a Λ_{10} loss matrix. There are no Class 2 correct samples, which are also shown in the confusion matrix.

2. Repeating with Λ_{100} loss matrix, we see a similar result with Λ_{10} but at a more extreme degree. In the confusion matrix, it seems to classify $D = 3 \& L = 3$ really close to 1. Even looking below at figure 10, we can make out that there are less incorrect in

general.

$$P(D = i|L = j) = \begin{bmatrix} 0.4227 & 0.0007 & 0.0012 \\ 0.1112 & 0.9947 & 0 \\ 0.4662 & 0.0046 & 0.9988 \end{bmatrix}$$

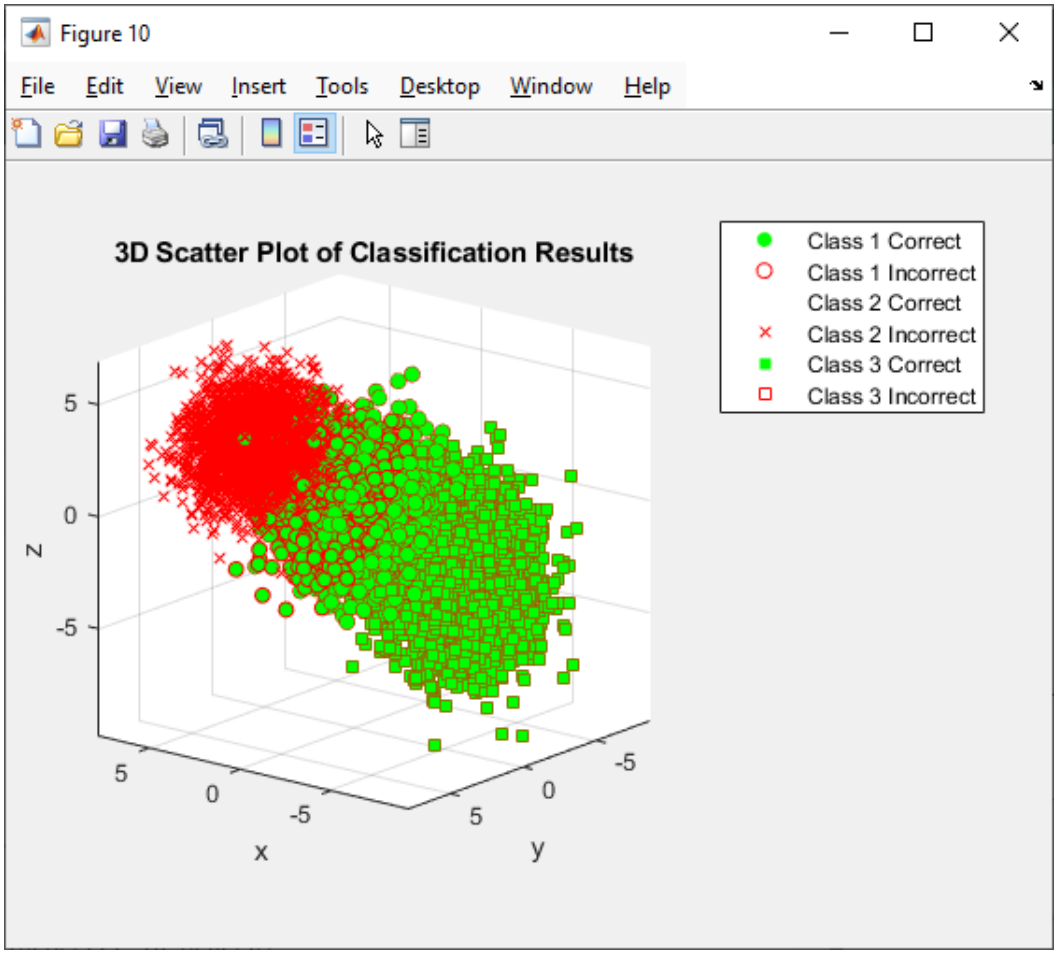

Figure 10 here shows the 3D plot representing the ERM samples with a Λ_{100} loss matrix. There are no Class 2 correct samples, which are also shown in the confusion matrix.

3 Question 3

3.1 Part A: Wine

Starting off this classifier, to get the mean vectors, I took the average of each dimension. For the priors, I found the percentage of each label for each wine. Note that there was no wine in the given sample that had $L = 0, 1, 2, 9, 10$. For the covariance matrix, I took the data and used $cov(x)$ to get the $C_{SampleAverage}$ of the data. To regularized the data, I used $\lambda = \frac{\alpha trace(C_{SampleAverage})}{rank(C_{SampleAverage})}$. By using an $\alpha = 0.75$ and plugging into:

$$C_{Regularized} = C_{SampleAverage} + \lambda \mathbf{I}$$

This gave me a regularized matrix where I used it for every dimension.

For the loss matrix, I used multiples of 5 spread out. This was just a decision choice to weigh the extremes more heavily and wasn't a calculated choice.

$$loss = \begin{bmatrix} 0 & 5 & 10 & 15 & 20 & 25 & 30 & 35 & 40 & 45 & 50 \\ 5 & 0 & 5 & 10 & 15 & 20 & 25 & 30 & 35 & 40 & 45 \\ 10 & 5 & 0 & 5 & 10 & 15 & 20 & 25 & 30 & 35 & 40 \\ 15 & 10 & 5 & 0 & 5 & 10 & 15 & 20 & 25 & 30 & 35 \\ 20 & 15 & 10 & 5 & 0 & 5 & 10 & 15 & 20 & 25 & 30 \\ 25 & 20 & 15 & 10 & 5 & 0 & 5 & 10 & 15 & 20 & 25 \\ 30 & 25 & 20 & 15 & 10 & 5 & 0 & 5 & 10 & 15 & 20 \\ 35 & 30 & 25 & 20 & 15 & 10 & 5 & 0 & 5 & 10 & 15 \\ 40 & 35 & 30 & 25 & 20 & 15 & 10 & 5 & 0 & 5 & 10 \\ 45 & 40 & 35 & 30 & 25 & 20 & 15 & 10 & 5 & 0 & 5 \\ 50 & 45 & 40 & 35 & 30 & 25 & 20 & 15 & 10 & 5 & 0 \end{bmatrix}$$

To take a look at the visualization, I took a few different classifications where the features were correlated and made plots to visualize them. Looking figure 11, 12, 13, we can see that there was a low accuracy when it comes to plotting the x dimensions. Most of the accuracy comes from But the classifications of the labels were a lot more accurate at about a 92.6% accuracy rate. To classify the labels, I took the amount that were the correct label and divided that number by the total sample.

in general, while this classifier may not be good at classifying the features, it does a decent job at classifying the labels for the wine quality.

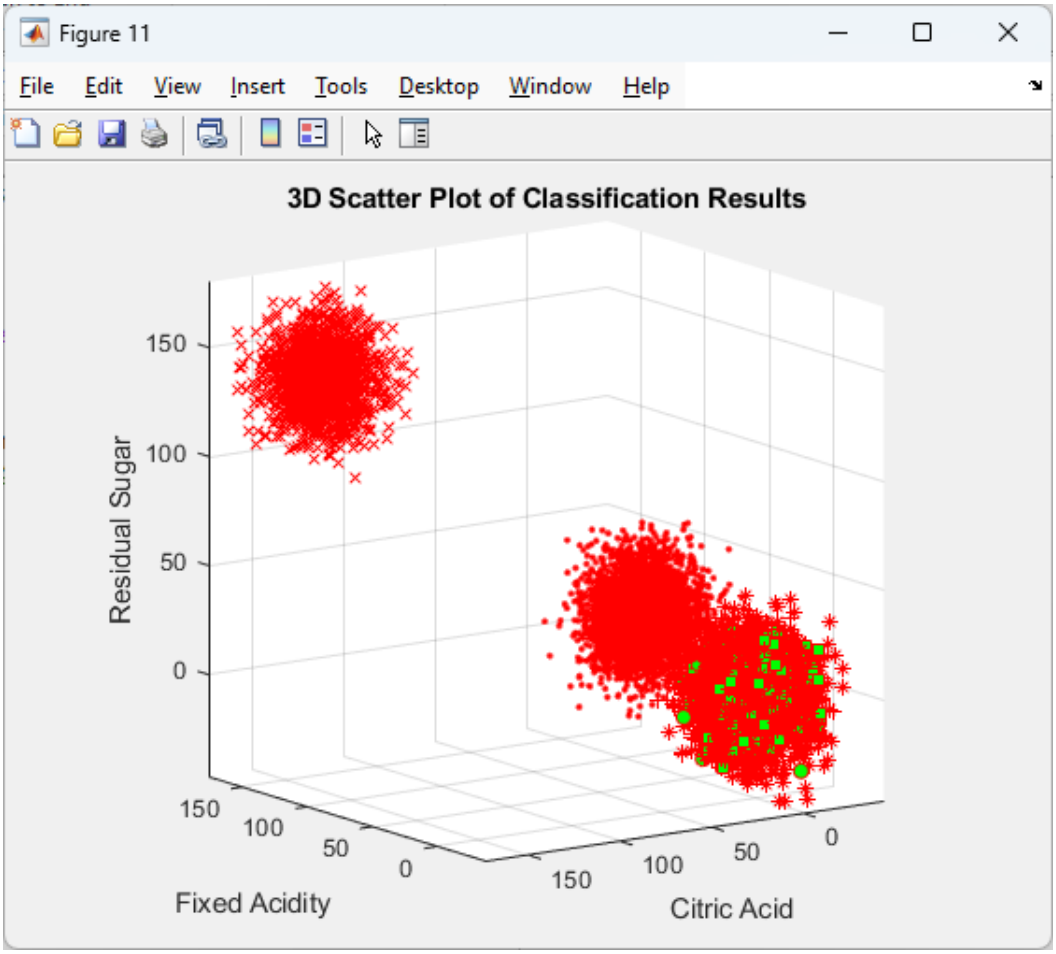

Figure 11 here shows the classifications of the Fixed Acidity, Residual Sugar, and Citric Acid of the wine. The green represents the correct classification while the red reprsents the incorrect classifications.

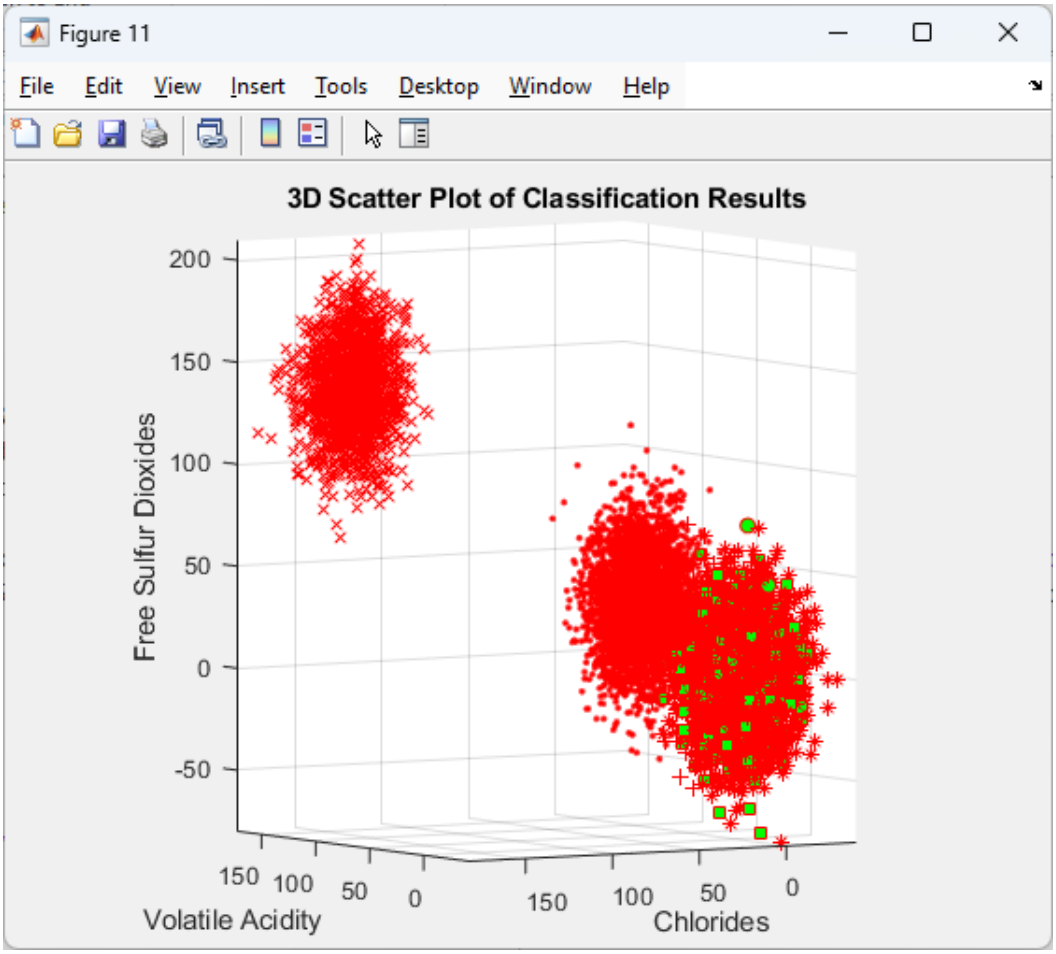

Figure 12 here shows the classifications of the Volatile Acidity, Chlorides, and Free Sulfur Dioxides of the wine. The green represents the correct classification while the red reprsents the incorrect classifications.

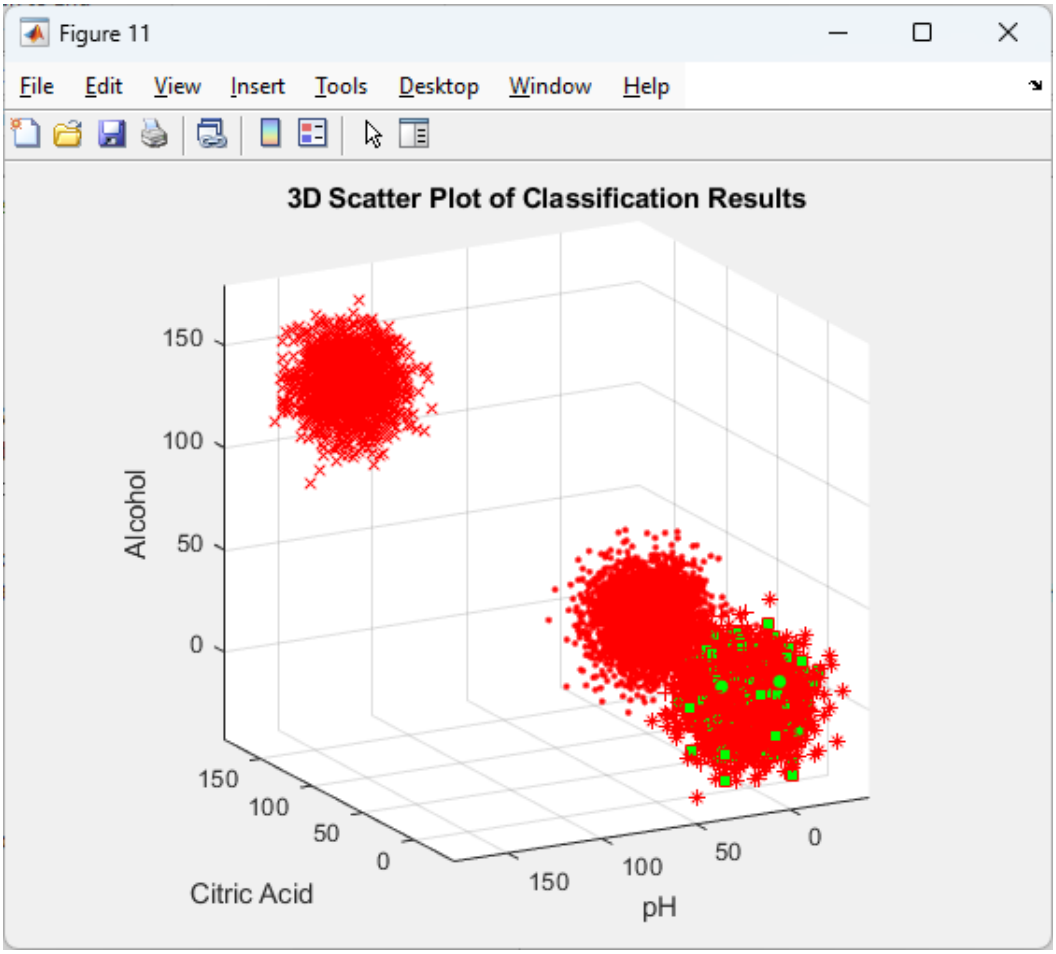

Figure 13 here shows the classifications of the Citric Acid, pH, and Alcohol of the wine. The green represents the correct classification while the red reprsents the incorrect classifications.

3.2 Part B: Human Activity

For the class priors and mean vectors, I used the same process similar to part a.

To implement a minimum-probability-of-error classifier, I used the same ERM model I have been using. Given the amount of features, there were dimension issues with the problem. If we were to do a $562 \times 562x \times 562$ model, it would take forever to classify. Given this, I adopted a $6 \times 6 \times 6$ due to runtime. Because I had to make these restrictions, I broke down my features into 6 even parts. I took each even part and I took an average of them to give me 6 dimensions. While this loses out on accuracy, this saves on runtime as my computer would not be able to efficiently handle a $562 \times 562x \times 562$ dimension.

Below is the confusion matrix for human activity data.

$$Confusion = \begin{bmatrix} 0.0060 & 0.0047 & 0 & 0 & 0 & \\ 0.8743 & 0.8627 & 0.6569 & 0.0039 & 0.0034 & 0.0204 \\ 0.0770 & 0.0861 & 0.1751 & 0.0178 & 0.0120 & 0.0453 \\ 0.0297 & 0.0337 & 0.0967 & 0.0450 & 0.0353 & 0.0862 \\ 0.0130 & 0.0128 & 0.0706 & 0.9333 & 0.9492 & 0.8480 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Looking at this data, we can see a lot of uncertainty with the data. But given the dataset, it would have been too complicated to run a super high dimension data where now it is restricted.

Of the 50000 samples generated, only 16875 of them were labeled correctly. This gives a 33.75% accuracy rate which is about twice as good as random guessing at 16.6%.

In general, this model would not be good for classifying both the features and the labels. Especially for the features, taking an average is not good for generating an accurate dataset. For the classifications of the labels, it does a better job as it is able to somewhat do better than random chance. Even though it does better than random, it is not a good classification as it is only accurate about 34% of the time.

4 Appendix

4.1 Code for Problem 1A

```
%% P1 Part A

N = 10000; % number of iid samples
n = 4;
mu(:,1) = [-1;-1;-1;-1];
mu(:,2) = [1;1;1;1];
sigma(:,:,1) =
   [2,-0.5,0.3,0;-0.5,1,-0.5,0;0.3,-0.5,1,0;0,0,0,2];

sigma(:,:,2) = [1,0.3,-0.2,0;0.3,2,0.3,0;-0.2,0.3,1,0;0,0,0,3];
p = [0.35,0.65]; % class priors for labels 0 and 1 respectively
label = rand(1,N) >= p(1);
Nc = [length(find(label==0)),length(find(label==1))]; % number
   of samples from each class
x = zeros(n,N); % save up space
% Draw samples from each class pdf
```

```
for l = 0:1
    x(:,label==l) = mvnrnd(mu(:,l+1),sigma(:,:,l+1),Nc(l+1))';
end
% Plotting Data
figure(1),
plot(x(1,label==0),x(2,label==0),'o'), hold on,
plot(x(1,label==1),x(2,label==1),'+'), axis equal,
legend('Class 0','Class 1'),
title('Data and their true labels'),
xlabel('x_1'), ylabel('x_2'),
hold off;

% Taken from LDAwithROCcurve with minor tweaks
function [Pfp,Ptp,Perror,thresholdList] = ROCcurve(
   discriminantScores,labels)
[sortedScores,~] = sort(discriminantScores,'ascend');
thresholdList = [min(sortedScores)-eps,(sortedScores(1:end-1)+
   sortedScores(2:end))/2, max(sortedScores)+eps];
for i = 1:length(thresholdList)
    tau = thresholdList(i);
    decisions = (discriminantScores >= tau);
    Ptn(i) = length(find(decisions==0 & labels==0))/length(find
       (labels==0));
    Pfp(i) = length(find(decisions==1 & labels==0))/length(find
       (labels==0));
    Ptp(i) = length(find(decisions==1 & labels==1))/length(find
       (labels==1));
    Perror(i) = sum(decisions~=labels)/length(labels);
end
end

% ROC Curve
g1 = evalGaussian(x,mu(:,1),sigma(:,:,1)); g2 = evalGaussian(x,
   mu(:,2),sigma(:,:,2));
discriminantScoresERM = log(g2./g1);
figure(2),
[PfpERM,PtpERM,PerrorERM,thresholdListERM] = ROCcurve(
   discriminantScoresERM,label);

% Getting Gamma Experimental and Theoretical
% Typing to Console
gamma_theoretical = 0.35/0.65; % Theoretical P(L=0) / P(L=1)
```

```
gamma_theoretical,

gamma_experimental = Nc(1) / Nc(2);
gamma_experimental,

% Getting Labels
dis_0 = discriminantScoresERM(label == 0);
dis_1 = discriminantScoresERM(label == 1);

% Calculate lambdas
thy_lambda_0 = sum(dis_0 >= gamma_theoretical) / length(dis_0);
    % False Positive Rate
thy_lambda_1 = sum(dis_1 >= gamma_theoretical) / length(dis_1);
    % True Positive Rate
thy_lambdas = [thy_lambda_0, thy_lambda_1];

% Calculate probability of error
thy_p_err = thy_lambdas(1) * 0.7 + (1 - thy_lambdas(2)) * 0.3;

% Typing to Console
Perror_theoretical = thy_p_err;
Perror_theoretical,

Perror_experimental = min(PerrorERM);
Perror_experimental,

perror_index = find(min(PerrorERM));
% Plotting ROC Curve
hold on, plot(PfpERM,PtpERM,'*m'),
hold on, plot(thy_lambda_0,thy_lambda_1,"*b"),
xlabel('P(False+)'),ylabel('P(True+)'), title('ROC Curve'),

writematrix(x,'samples_1a.csv');
writematrix(label,'samples_1a.csv','WriteMode','append');
```

4.2 Code for Problem 1B

```
%% P1 Part B

N = 10000; % number of iid samples
n = 4;
```

```
mu(:,1) = [-1;-1;-1;-1];
mu(:,2) = [1;1;1;1];
sigma(:,:,1) = [2,0,0,0;
                0,1,0,0;
                0,0,1,0;
                0,0,0,2];

sigma(:,:,2) = [1,0,0,0;
                0,2,0,0;
                0,0,1,0;
                0,0,0,3];
p = [0.35,0.65]; % class priors for labels 0 and 1 respectively
label = rand(1,N) >= p(1);
Nc = [length(find(label==0)),length(find(label==1))]; % number
   of samples from each class
x = zeros(n,N); % save up space
% Draw samples from each class pdf
for l = 0:1
    x(:,label==l) = mvnrnd(mu(:,l+1),sigma(:,:,l+1),Nc(l+1))';
end
% Plotting Data
figure(3),
plot(x(1,label==0),x(2,label==0),'o'), hold on,
plot(x(1,label==1),x(2,label==1),'+'), axis equal,
legend('Class 0','Class 1'),
title('Data and their true labels'),
xlabel('x_1'), ylabel('x_2'),
hold off;

% Taken from LDAwithROCcurve
function [Pfp,Ptp,Perror,thresholdList] = ROCcurve(
   discriminantScores,labels)
[sortedScores,~] = sort(discriminantScores,'ascend');
thresholdList = [min(sortedScores)-eps,(sortedScores(1:end-1)+
   sortedScores(2:end))/2, max(sortedScores)+eps];
for i = 1:length(thresholdList)
    tau = thresholdList(i);
    decisions = (discriminantScores >= tau);
    Ptn(i) = length(find(decisions==0 & labels==0))/length(find
       (labels==0));
    Pfp(i) = length(find(decisions==1 & labels==0))/length(find
       (labels==0));
```

```
    Ptp(i) = length(find(decisions==1 & labels==1))/length(find
       (labels==1));
    Perror(i) = sum(decisions~=labels)/length(labels);
end
end

% ROC Curve
g1 = evalGaussian(x,mu(:,1),sigma(:,:,1)); g2 = evalGaussian(x,
   mu(:,2),sigma(:,:,2));
discriminantScoresERM = log(g2./g1);
figure(4),
[PfpERM,PtpERM,PerrorERM,thresholdListERM] = ROCcurve(
   discriminantScoresERM,label);

% Getting Gamma Experimental and Theoretical
% Typing to Console
gamma_theoretical_naive = 0.35/0.65; % Theoretical P(L=0) / P(L
   =1)
gamma_theoretical_naive,

gamma_experimental_naive = Nc(1) / Nc(2);
gamma_experimental_naive,

% Getting Labels
dis_0 = discriminantScoresERM(label == 0);
dis_1 = discriminantScoresERM(label == 1);

% Calculate lambdas
thy_lambda_0 = sum(dis_0 >= gamma_theoretical_naive) / length(
   dis_0); % False Positive Rate
thy_lambda_1 = sum(dis_1 >= gamma_theoretical_naive) / length(
   dis_1); % True Positive Rate
thy_lambdas = [thy_lambda_0, thy_lambda_1];

% Calculate probability of error
thy_p_err = thy_lambdas(1) * 0.7 + (1 - thy_lambdas(2)) * 0.3;

% Typing to Console
Perror_theoretical_naive = thy_p_err;
Perror_theoretical_naive,

Perror_experimental_naive = min(PerrorERM);
```

```
Perror_experimental_naive,

% Plotting ROC Curve
hold on, plot(PfpERM,PtpERM,'*m'),
hold on, plot(thy_lambda_0,thy_lambda_1,"*b"),
xlabel('P(False+)'),ylabel('P(True+)'), title('ROC Curve'),

writematrix(x,'samples_1b.csv');
writematrix(label,'samples_1b.csv','WriteMode','append');
```

4.3 Code for Problem 1C

```
                clear all, close all,

% Generate n-dimensional data vectors from 2 Gaussian pdfs
n = 4;
N1 = 5000; mu1 = [-1;-1;-1;-1]; A1 =
   [2,-0.5,0.3,0;-0.5,1,-0.5,0;0.3,-0.5,1,0;0,0,0,2];
N2 = 5000; mu2 = [1;1;1;1]; A2 =
   [1,0.3,-0.2,0;0.3,2,0.3,0;-0.2,0.3,1,0;0,0,0,3];
x1 = A1*randn(n,N1)+mu1*ones(1,N1);
x2 = A2*randn(n,N2)+mu2*ones(1,N2);
labels = [zeros(1,N1),ones(1,N2)];

% Estimate mean vectors and covariance matrices from samples
mu1hat = mean(x1,2); S1hat = cov(x1');
mu2hat = mean(x2,2); S2hat = cov(x2');

% Calculate the between/within-class scatter matrices
Sb = (mu1hat-mu2hat)*(mu1hat-mu2hat)';
Sw = S1hat + S2hat;

% Solve for the Fisher LDA projection vector (in w)
[V,D] = eig(inv(Sw)*Sb);
[~,ind] = sort(diag(D),'descend');
w = V(:,ind(1)); % Fisher LDA projection vector

% Linearly project the data from both categories on to w
y1 = w'*x1;
y2 = w'*x2;

% Plot the data before and after linear projection
```

```
figure(5),
hold on, subplot(2,1,1), plot(x1(1,:),x1(2,:),'r*');
hold on, plot(x2(1,:),x2(2,:),'bo'); axis equal,
hold on, subplot(2,1,2), plot(y1(1,:),zeros(1,N1),'r*');
hold on, plot(y2(1,:),zeros(1,N2),'bo'); axis equal,

% ROC curve for Fisher LDA
discriminantScoresLDA = [y1,y2];
[PfpLDA,PtpLDA,PerrorLDA,thresholdListLDA] = ROCcurve(
   discriminantScoresLDA,labels);
Perror_lda = min(PerrorLDA);
i = find(PerrorLDA == min(PerrorLDA));
figure(6),
hold on, plot(PfpLDA,PtpLDA,'*m'),
hold on, plot(PfpLDA(i),PtpLDA(i),'*b'),
xlabel('P(False+)'),ylabel('P(True+)'), title('ROC Curve for
   LDA Discriminant Scores'),
figure(7),
hold on, plot(thresholdListLDA,PerrorLDA,'og'),
xlabel('Thresholds'), ylabel('P(error) for LDA Discriminant
   Scores'),


function [Pfp,Ptp,Perror,thresholdList] = ROCcurve(
   discriminantScores,labels)
[sortedScores,ind] = sort(discriminantScores,'ascend');
thresholdList = [min(sortedScores)-eps,(sortedScores(1:end-1)+
   sortedScores(2:end))/2, max(sortedScores)+eps];
for i = 1:length(thresholdList)
    tau = thresholdList(i);
    decisions = (discriminantScores >= tau);
    Ptn(i) = length(find(decisions==0 & labels==0))/length(find
       (labels==0));
    Pfp(i) = length(find(decisions==1 & labels==0))/length(find
       (labels==0));
    Ptp(i) = length(find(decisions==1 & labels==1))/length(find
       (labels==1));
    Perror(i) = sum(decisions~=labels)/length(labels);
end
end
```

4.4 Code for Problem 2A

```
                % Expected risk minimization with C classes
clear all, close all,

C = 3;
N = 10000; % Number of samples
n = 3; % Data dimensionality (must be 2 for plots to work)

% Specify mean vectors (manually chosen for desired separations
   )
gmmParameters.priors = [0.3, 0.3, 0.4];
gmmParameters.meanVectors = [0,0,0;3,3,3;-3,-3,-3;6,6,6]';

% Define covariance matrices for each Gaussian component
cov_base = 2 * eye(n);
gmmParameters.covMatrices(:,:,1) = [2.5 ,-0.2, 0.1;
                                    -0.2,1.8,-0.3;
                                    0.1 ,-0.3,2.1 ];

gmmParameters.covMatrices(:,:,2) = [1.7 ,-0.1 ,0.3 ;
                                    -0.1,2.3  ,-0.5;
                                    0.3 ,-0.5 ,1.5 ];
gmmParameters.covMatrices(:,:,3) = [3.0 ,-0.5,0.4 ;
                                    -0.5,2.8,-0.2;
                                    0.4 ,-0.2,3.2 ];
gmmParameters.covMatrices(:,:,4) = [2.9 ,-0.3,0.1 ;
                                    -0.3,3.1 ,-0.4;
                                    0.1 ,-0.4,2.7 ];

gmmParameters.covMatrices(:,:,1),
gmmParameters.covMatrices(:,:,2),
gmmParameters.covMatrices(:,:,3),
gmmParameters.covMatrices(:,:,4),

% Generate data from specified pdf
[x,labels] = generateDataFromGMM(N,gmmParameters); % Generate
   data
for l = 1:C
    Nclass(l,1) = length(find(labels==l));
end

% Evaluate class-conditional PDFs for each Gaussian component
```

```
pxgivenl = zeros(C, N); % Initialize
for l = 1:C
    if l == 3
        % Condition for L = 3 where it is a combination
        pxgivenl(l, :) = 0.5 * evalGaussianPDF(x, gmmParameters
           .meanVectors(:,3), gmmParameters.covMatrices(:,:,3))
            + ...
                         0.5 * evalGaussianPDF(x, gmmParameters
                            .meanVectors(:,4), gmmParameters.
                            covMatrices(:,:,4));
    else
        % Class 1 and Class 2 each have one Gaussian
        pxgivenl(l, :) = evalGaussianPDF(x, gmmParameters.
           meanVectors(:,l), gmmParameters.covMatrices(:,:,l));
    end
end

% Calculate total probability
px = gmmParameters.priors*pxgivenl; % Total probability theorem
classPosteriors = pxgivenl.*repmat(gmmParameters.priors',1,N)./
   repmat(px,C,1); % P(L=l|x)

lossMatrix = ones(C,C)-eye(C); % For min-Perror design, use 0-1
    loss
lossMatrix,
expectedRisks = lossMatrix*classPosteriors; % Expected Risk for
    each label (rows) for each sample (columns)
[~,decisions] = min(expectedRisks,[],1); % Minimum expected
   risk decision with 0-1 loss is the same as MAP

% The matrix computed here is P(D=d|L=l,x)
% It is called the confusion matrix if decisions are class
   labels.
figure;
hold on;
colors = {'g', 'r'}; % Green for correct, red for incorrect
shapes = {'o', 'x', 's'}; % Circle, X, square for each class
figure(8), clf,
for d = 1:C % each decision option
    for l = 1:C % each class label
        i_correct = find(decisions == d & labels == l);
        i_incorrect = find(decisions ~= d & labels == l);
```

```
        ConfusionMatrix(d,l) = length(i_correct)/length(find(
            labels==l));
        scatter3(x(1, i_correct), x(2, i_correct), x(3,
            i_correct), 36, colors{l}, shapes{l}, 'filled');
            hold on, axis equal, grid on,
        scatter3(x(1, i_incorrect), x(2, i_incorrect), x(3,
            i_incorrect), 36, colors{2}, shapes{l}); hold on,
            axis equal, grid on,
    end
end


title('3D Scatter Plot of Classification Results');
xlabel('X1'); ylabel('X2'); zlabel('X3');
legend({'Class 1 Correct', 'Class 1 Incorrect', 'Class 2
    Correct', 'Class 2 Incorrect', 'Class 3 Correct', 'Class 3
    Incorrect'});
grid on;
hold off;
ConfusionMatrix,

writematrix(x,'samples_2a.csv');
writematrix(labels,'samples_2a.csv','WriteMode','append');
```

4.5 Code for Problem 2B for Loss=10 Matrix

```
                % Expected risk minimization with C classes
clear all, close all,

C = 3;
N = 10000; % Number of samples
n = 3; % Data dimensionality (must be 2 for plots to work)

% Specify mean vectors (manually chosen for desired separations
    )
gmmParameters.priors = [0.3, 0.3, 0.4];
gmmParameters.meanVectors = [0,0,0;3,3,3;-3,-3,-3;6,-6,0]';

% Define covariance matrices for each Gaussian component
cov_base = 2 * eye(n);
gmmParameters.covMatrices(:,:,1) = [2.5 ,-0.2, 0.1;
                                    -0.2,1.8,-0.3;
```

```
                                        0.1  ,-0.3,2.1 ];

gmmParameters.covMatrices(:,:,2) = [1.7  ,-0.1  ,0.3 ;
                                    -0.1,2.3   ,-0.5;
                                    0.3  ,-0.5  ,1.5 ];
gmmParameters.covMatrices(:,:,3) = [3.0  ,-0.5,0.4 ;
                                    -0.5,2.8,-0.2;
                                    0.4  ,-0.2,3.2 ];
gmmParameters.covMatrices(:,:,4) = [2.9  ,-0.3,0.1 ;
                                    -0.3,3.1  ,-0.4;
                                    0.1  ,-0.4,2.7 ];

gmmParameters.covMatrices(:,:,1),
gmmParameters.covMatrices(:,:,2),
gmmParameters.covMatrices(:,:,3),
gmmParameters.covMatrices(:,:,4),

% Generate data from specified pdf
[x,labels] = generateDataFromGMM(N,gmmParameters); % Generate
   data
for l = 1:C
    Nclass(l,1) = length(find(labels==l));
end

% Evaluate class-conditional PDFs for each Gaussian component
pxgivenl = zeros(C, N); % Initialize
for l = 1:C
    if l == 3
        % Condition for L = 3 where it is a combination
        pxgivenl(l, :) = 0.5 * evalGaussianPDF(x, gmmParameters
           .meanVectors(:,3), gmmParameters.covMatrices(:,:,3))
            + ...
                         0.5 * evalGaussianPDF(x, gmmParameters
                            .meanVectors(:,4), gmmParameters.
                            covMatrices(:,:,4));
    else
        % Class 1 and Class 2 each have one Gaussian
        pxgivenl(l, :) = evalGaussianPDF(x, gmmParameters.
           meanVectors(:,l), gmmParameters.covMatrices(:,:,l));
    end
end
```

```
% Calculate total probability
px = gmmParameters.priors*pxgivenl; % Total probability theorem
classPosteriors = pxgivenl.*repmat(gmmParameters.priors',1,N)./
   repmat(px,C,1); % P(L=l|x)

lossMatrix10 = [0,10,10;
                1,0 ,10;
                1,1 ,0];

lossMatrix = lossMatrix10;
expectedRisks = lossMatrix*classPosteriors; % Expected Risk for
    each label (rows) for each sample (columns)
[~,decisions] = min(expectedRisks,[],1); % Minimum expected
   risk decision with 0-1 loss is the same as MAP

% The matrix computed here is P(D=d|L=l,x)
% It is called the confusion matrix if decisions are class
   labels.
hold on;
colors = {'g', 'r'}; % Green for correct, red for incorrect
shapes = {'o', 'x', 's'}; % Circle, X, square for each class
figure(9), clf,
for d = 1:C % each decision option
    for l = 1:C % each class label
        i_correct = find(decisions == d & labels == l);
        i_incorrect = find(decisions ~= d & labels == l);
        ConfusionMatrix(d,l) = length(i_correct)/length(find(
           labels==l));
        scatter3(x(1, i_correct), x(2, i_correct), x(3,
           i_correct), 36, colors{1}, shapes{l}, 'filled');
           hold on, axis equal, grid on,
        scatter3(x(1, i_incorrect), x(2, i_incorrect), x(3,
           i_incorrect), 36, colors{2}, shapes{l}); hold on,
           axis equal, grid on,
    end
end

title('3D Scatter Plot of Classification Results');
xlabel('x'); ylabel('y'); zlabel('z');
legend({'Class 1 Correct', 'Class 1 Incorrect', 'Class 2
   Correct', 'Class 2 Incorrect', 'Class 3 Correct', 'Class 3
   Incorrect'});
```

```
grid on;
hold off;
ConfusionMatrix,

writematrix(x,'samples_2b_10.csv');
writematrix(labels,'samples_2b_10.csv','WriteMode','append');
```

4.6 Code for 2B for Loss=100

```
                % Expected risk minimization with C classes
clear all, close all,

C = 3;
N = 10000; % Number of samples
n = 3; % Data dimensionality (must be 2 for plots to work)

% Specify mean vectors (manually chosen for desired separations
   )
gmmParameters.priors = [0.3, 0.3, 0.4];
gmmParameters.meanVectors = [0,0,0;3,3,3;-3,-3,-3;6,-6,0]';

% Define covariance matrices for each Gaussian component
cov_base = 2 * eye(n);
gmmParameters.covMatrices(:,:,1) = [2.5 ,-0.2, 0.1;
                                    -0.2,1.8,-0.3;
                                    0.1 ,-0.3,2.1 ];

gmmParameters.covMatrices(:,:,2) = [1.7 ,-0.1 ,0.3 ;
                                    -0.1,2.3  ,-0.5;
                                    0.3 ,-0.5 ,1.5 ];
gmmParameters.covMatrices(:,:,3) = [3.0 ,-0.5,0.4 ;
                                    -0.5,2.8,-0.2;
                                    0.4 ,-0.2,3.2 ];
gmmParameters.covMatrices(:,:,4) = [2.9 ,-0.3,0.1 ;
                                    -0.3,3.1 ,-0.4;
                                    0.1 ,-0.4,2.7 ];

gmmParameters.covMatrices(:,:,1),
gmmParameters.covMatrices(:,:,2),
gmmParameters.covMatrices(:,:,3),
gmmParameters.covMatrices(:,:,4),
```

```
% Generate data from specified pdf
[x,labels] = generateDataFromGMM(N,gmmParameters); % Generate
   data
for l = 1:C
    Nclass(l,1) = length(find(labels==l));
end

% Evaluate class-conditional PDFs for each Gaussian component
pxgivenl = zeros(C, N); % Initialize
for l = 1:C
    if l == 3
        % Condition for L = 3 where it is a combination
        pxgivenl(l, :) = 0.5 * evalGaussianPDF(x, gmmParameters
           .meanVectors(:,3), gmmParameters.covMatrices(:,:,3))
            + ...
                         0.5 * evalGaussianPDF(x, gmmParameters
                            .meanVectors(:,4), gmmParameters.
                            covMatrices(:,:,4));
    else
        % Class 1 and Class 2 each have one Gaussian
        pxgivenl(l, :) = evalGaussianPDF(x, gmmParameters.
           meanVectors(:,l), gmmParameters.covMatrices(:,:,l));
    end
end

% Calculate total probability
px = gmmParameters.priors*pxgivenl; % Total probability theorem
classPosteriors = pxgivenl.*repmat(gmmParameters.priors',1,N)./
   repmat(px,C,1); % P(L=l|x)

lossMatrix100 = [0,100,100;
                 1 ,0   ,100;
                 1 ,1   ,0 ];

lossMatrix = lossMatrix100;
expectedRisks = lossMatrix*classPosteriors; % Expected Risk for
    each label (rows) for each sample (columns)
[~,decisions] = min(expectedRisks,[],1); % Minimum expected
   risk decision with 0-1 loss is the same as MAP

% The matrix computed here is P(D=d|L=l,x)
% It is called the confusion matrix if decisions are class
```

```
   labels.
hold on;
colors = {'g', 'r'}; % Green for correct, red for incorrect
shapes = {'o', 'x', 's'}; % Circle, X, square for each class
figure(10), clf,
for d = 1:C % each decision option
    for l = 1:C % each class label
        i_correct = find(decisions == d & labels == l);
        i_incorrect = find(decisions ~= d & labels == l);
        ConfusionMatrix(d,l) = length(i_correct)/length(find(
           labels==l));
        scatter3(x(1, i_correct), x(2, i_correct), x(3,
           i_correct), 36, colors{1}, shapes{l}, 'filled');
           hold on, axis equal, grid on,
        scatter3(x(1, i_incorrect), x(2, i_incorrect), x(3,
           i_incorrect), 36, colors{2}, shapes{l}); hold on,
           axis equal, grid on,
    end
end

title('3D Scatter Plot of Classification Results');
xlabel('x'); ylabel('y'); zlabel('z');
legend({'Class 1 Correct', 'Class 1 Incorrect', 'Class 2
   Correct', 'Class 2 Incorrect', 'Class 3 Correct', 'Class 3
   Incorrect'});
grid on;
hold off;
ConfusionMatrix,

writematrix(x,'samples_2b_100.csv');
writematrix(labels,'samples_2b_100.csv','WriteMode','append');
```

4.7 Code for Problem 3 Wine

```
                % Expected risk minimization with C classes
clear all, close all,

% Loading Dataset
data_wine = readtable('winedata.csv', 'Delimiter', ';');

% Intializing Dataset
```

```
init_data = table2array(data_wine(:, 1:end-1)); % Takes the
   features of the wine
labels = table2array(data_wine(:,end)); % Takes the last column

C = 11;
N = 10000; % Number of samples
n = 11; % Features for the wine

% Initialize priors vector
gmmParameters.priors = zeros(1, C);

% Generates priors based on mean
for l = 1:C
    gmmParameters.priors(l) = sum(labels == l) / 4898; % Prior
       for each class
end

gmmParameters.meanVectors = zeros(1, C);
for l = 1:C
    for m = 1:C
        gmmParameters.meanVectors(m,l) = sum(init_data(:,l)) /
           4898;
    end
end

% Initialize covariance matrix
gmmParameters.covMatrices = zeros(n, n, C);

lambda = 0.75*trace(cov(init_data))/rank(cov(init_data));
C_regularized = cov(init_data) + lambda * eye(C,C);

for l = 1:C
    gmmParameters.covMatrices(:,:,l) = C_regularized;
end


% Generate data from specified pdf
[x,labels] = generateDataFromGMM(N,gmmParameters); % Generate
   data
for l = 1:C
    Nclass(l,1) = length(find(labels==l));
end
```

```
% Shared computation for both parts
for l = 1:C
    pxgivenl(l,:) = evalGaussianPDF(x,gmmParameters.meanVectors
       (:,l),gmmParameters.covMatrices(:,:,l)); % Evaluate p(x|
       L=l)
end

px = gmmParameters.priors*pxgivenl; % Total probability theorem
classPosteriors = pxgivenl.*repmat(gmmParameters.priors',1,N)./
   repmat(px,C,1); % P(L=l|x)


lossMatrix = [0,5,10,15,20,25,30,35,40,45,50;
              5,0,5,10,15,20,25,30,35,40,45;
              10,5,0,5,10,15,20,25,30,35,40;
              15,10,5,0,5,10,15,20,25,30,35;
              20,15,10,5,0,5,10,15,20,25,30;
              25,20,15,10,5,0,5,10,15,20,25;
              30,25,20,15,10,5,0,5,10,15,20;
              35,30,25,20,15,10,5,0,5,10,15;
              40,35,30,25,20,15,10,5,0,5,10;
              45,40,35,30,25,20,15,10,5,0,5;
              50,45,40,35,30,25,20,15,10,5,0;];
expectedRisks = lossMatrix*classPosteriors; % Expected Risk for
    each label (rows) for each sample (columns)
[~,decisions] = min(expectedRisks,[],1); % Minimum expected
   risk decision with 0-1 loss is the same as MAP

% It is called the confusion matrix if decisions are class
   labels.
colors = {'g', 'r'}; % Green for correct label, red for
   incorrect label
shapes = {'', '', 'o', '+', '*', '.', 'x', 'square', 'hexagram'
   , '', ''}; % For each class that has prior > 0
figure(11), clf,
for d = 1:C % each decision option
    for l = 1:C % each class label
        i_correct = find(decisions == d & labels == l);
        i_incorrect = find(decisions ~= d & labels == l);
        ConfusionMatrix(d,l) = length(i_correct)/length(find(
           labels==l));
```

```
        if (d > 1) && (d < 9)
            scatter3(x(3, i_incorrect), x(9, i_incorrect), x
               (11, i_incorrect), 36, colors{2}, shapes{1});
               hold on, axis equal, grid on,
            scatter3(x(3, i_correct), x(9, i_correct), x(11,
               i_correct), 36, colors{1}, shapes{1}, 'filled');
                hold on, axis equal, grid on,
        end
    end
end
ConfusionMatrix,

title('3D Scatter Plot of Classification Results');
xlabel('Citric Acid'); ylabel('pH'); zlabel('Alcohol');

count = 0;
for i = 1:length(decisions)
    if decisions(i) == labels(i)
        count = count + 1;
    end
end
count,

writematrix(x,'samples_3_wine.csv');
writematrix(labels,'samples_3_wine.csv','WriteMode','append');
```

4.8 Code for Problem 3 Human Activity

```
                % Expected risk minimization with C classes
clear all, close all,

% Loading Dataset
X1 = readmatrix('UCI HAR Dataset\train\X_train.txt');
X2 = readmatrix('UCI HAR Dataset\test\X_test.txt');

Y1 = readmatrix('UCI HAR Dataset\train\y_train.txt');
Y2 = readmatrix('UCI HAR Dataset\test\y_test.txt');


list_data = [X1;X2];
list_label = [Y1;Y2];
```

```
l1 = mean(list_data(:,1:90),2);
l2 = mean(list_data(:,91:180),2);
l3 = mean(list_data(:,181:270),2);
l4 = mean(list_data(:,271:360),2);
l5 = mean(list_data(:,361:450),2);
l6 = mean(list_data(:,451:end),2);

init_data = [l1,l2,l3,l4,l5,l6];

labels = list_label;

% Intializing Dataset

C = 6;
N = 50000; % Number of samples
n = 6; % Features for human activity

% Initialize priors vector
gmmParameters.priors = zeros(1, C);

% Generates priors based on mean
for l = 1:C
    gmmParameters.priors(l) = sum(labels == l) / 10299; % Prior
        for each class
end

gmmParameters.meanVectors = zeros(1, C);
for l = 1:C
    for m = 1:C
        gmmParameters.meanVectors(m,l) = sum(init_data(:,l)) /
           10299;
    end
end

% Initialize covariance matrix
gmmParameters.covMatrices = zeros(n, n, C);

lambda = 0.75*trace(cov(init_data))/rank(cov(init_data));
C_regularized = cov(init_data) + lambda * eye(n,n);

for l = 1:C
    gmmParameters.covMatrices(:,:,l) = C_regularized;
```

```
end


% Generate data from specified pdf
[x,labels] = generateDataFromGMM(N,gmmParameters); % Generate
   data
for l = 1:C
    Nclass(l,1) = length(find(labels==l));
end

% Shared computation for both parts
for l = 1:C
    pxgivenl(l,:) = evalGaussianPDF(x,gmmParameters.meanVectors
       (:,l),gmmParameters.covMatrices(:,:,l)); % Evaluate p(x|
       L=l)
end

px = gmmParameters.priors*pxgivenl; % Total probability theorem
classPosteriors = pxgivenl.*repmat(gmmParameters.priors',1,N)./
   repmat(px,C,1); % P(L=l|x)


lossMatrix = [0,10,30,80,120,170;
              10,0,10,30,80,120;
              30,10,0,10,30,80;
              80,30,10,0,10,30;
              120,80,30,10,0,10;
              170,120,80,30,10,0];
lossMatrix = lossMatrix * 2;
expectedRisks = lossMatrix*classPosteriors; % Expected Risk for
    each label (rows) for each sample (columns)
[~,decisions] = min(expectedRisks,[],1); % Minimum expected
   risk decision with 0-1 loss is the same as MAP

% It is called the confusion matrix if decisions are class
   labels.
colors = {'g', 'r'}; % Green for correct label, red for
   incorrect label
shapes = {'', '', 'o', '+', '*', '.', 'x', 'square', 'hexagram'
   , '', ''}; % For each class that has prior > 0
figure(11), clf,
for d = 1:C % each decision option
```

```
    for l = 1:C % each class label
        i_correct = find(decisions == d & labels == l);
        i_incorrect = find(decisions ~= d & labels == l);
        ConfusionMatrix(d,l) = length(i_correct)/length(find(
           labels==l));
    end
end
ConfusionMatrix,

title('3D Scatter Plot of Classification Results');
xlabel('Citric Acid'); ylabel('pH'); zlabel('Alcohol');

count = 0;
for i = 1:length(decisions)
    if decisions(i) == labels(i)
        count = count + 1;
    end
end
count,

writematrix(x,'samples_3_human.csv');
writematrix(labels,'samples_3_human.csv','WriteMode','append');
```