EECE5644 Fall 2021
Homework #1

Runzhi Wang

Oct 6 2021

Problem 1

Part A

For this problem a 10000 sample, 2-dimensional real valued random vector X was generated with the following characteristics: $p(x) = P(L=0)p(x|L=0) + P(L=1)p(x|L=1)$ where L L is the true class label of the pdf which generated the data. The class conditional PDFs are defined as $p(x|L=0) = w_1 g(x|m_{01}, C_{01}) + w_2 g(x|m_{02}, C_{02})$ and $p(x|L=1) = g(x|m_1, C_1)$) where m is the mean vector and C is the covariance matrix.

To generate data, for $p(x|L=0)$

$$P(L=0)p(x|L=0) = w_1 P(L=0) g(x|\mu_1 \Sigma_1) + w_2 P(L=0) g(x|\mu_2 \Sigma_2)$$
$$P(L=1)p(x|L=1) = P(L=1) g(x|\mu_3 \Sigma_3)$$
$$p(x) = w_1 P(L=0) g(x|\mu_1 \Sigma_1) + w_2 P(L=0) g(x|\mu_2 \Sigma_2) + P(L=1) g(x|\mu_3 \Sigma_3)$$

$$E[x|L=0] = w_1 \mu_1 + w_2 \mu_2$$
$$Cov[x|L=0] = w_1 \Sigma_1 + w_2 \Sigma_2$$

The parameters and priors for these distributions are shown below. A plot of the vector **X** generated using these parametersis shown in Figure 1.

$$m_{01} = \begin{bmatrix} 3 \\ 0 \end{bmatrix} \; C_{01} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$
$$m_{02} = \begin{bmatrix} 0 \\ 3 \end{bmatrix} \; C_{02} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \quad P(L=0) = 0.65$$

$$m_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \; C_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad P(L=1) = 0.35$$

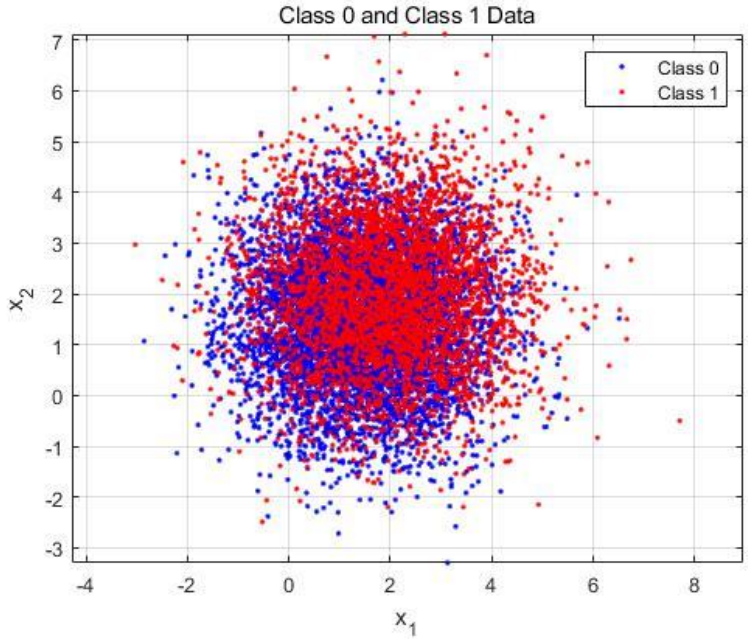

*Figure 1: **X** with True Labels for Problem 1*

1. Minimum expected risk classification rule in the form of a likelihood-ratio test

$$(D=1) \quad \frac{P(x|L1)}{P(x|L0)} \gtrless \frac{\lambda_{10}-\lambda_{00}}{\lambda_{01}-\lambda_{11}} * \frac{P(L0)}{P(L1)} = \gamma \quad (D=0)$$

   To minimize probability of misclassifications the cost for incorrect classification should be 1 and the cost for correct classifications should be 0 which results in the gamma shown below.

$$(D=1) \quad \frac{P(x|L1)}{P(x|L0)} \gtrless \frac{1-0}{1-0} * \frac{0.65}{0.35} = 1.86 = \gamma \quad (D=0)$$

2. The classifier was implemented for multiple values of gamma and the ROC curve is shown in Figure 2 below. The locations of the theoretical minimum error as well as the minimum error determined by a parametric sweep of gamma are marked on the plot.

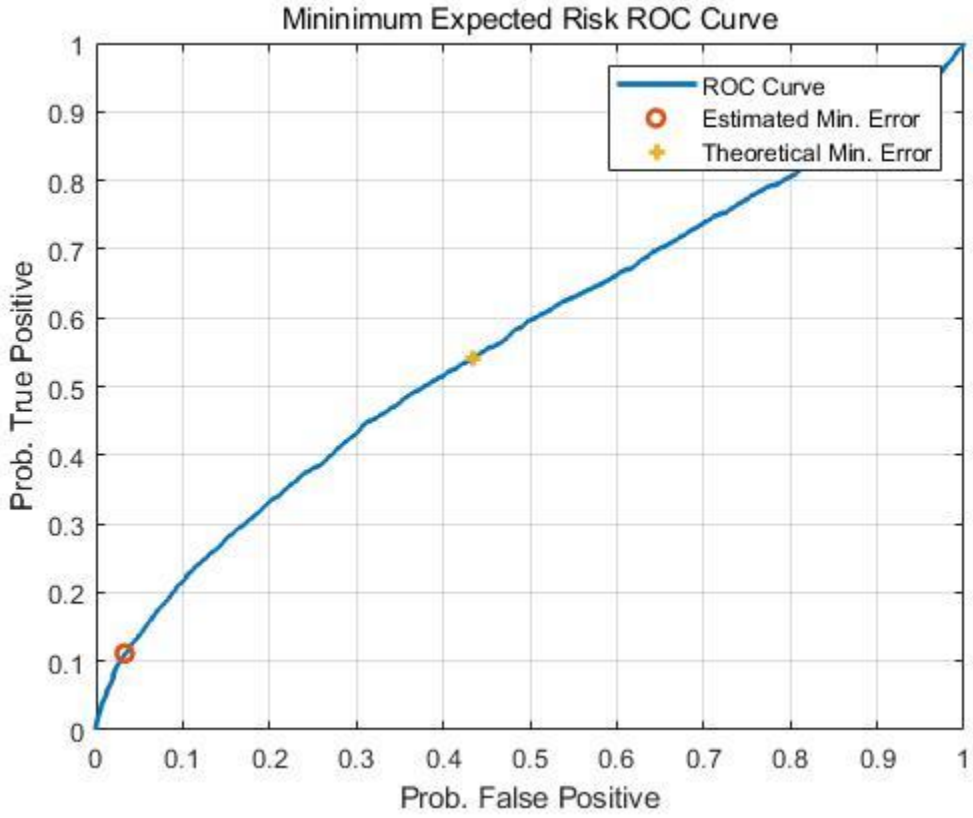

Figure 2: ROC Curve for ERM Classification with Known Data Distributions

3. Table 1 contains the theoretical and estimated gamma values that result in the minimum probability of error. As can be seen the two values closely align providing confidence in the estimated value.

Table 1: Comparison of Gammas to Produce Minimum Errors

	γ	$Min.P_{error}$
Theoretical	1.86	0.4046
Estimated From Data	9.10	0.3411

Figure 3 shows a plot of the probability of errors versus the gamma parameters. The location of the minimum error is marked. In addition, as the gamma parameter approached its limits at 0 and $+\infty$ the probability of error asymptotes to the priors for the two distributions. That is when the gamma is set to its minimum value all of the data points will be classified as class 1 so the overall error will be the proportion of data in class 0 which is equivalents to its prior. Similarly, when gamma is at $+\infty$ all of the data points will be classified as class 0 and so all of the class 1 data will be misclassified and the probability of error is equivalent to the prior for class 1.

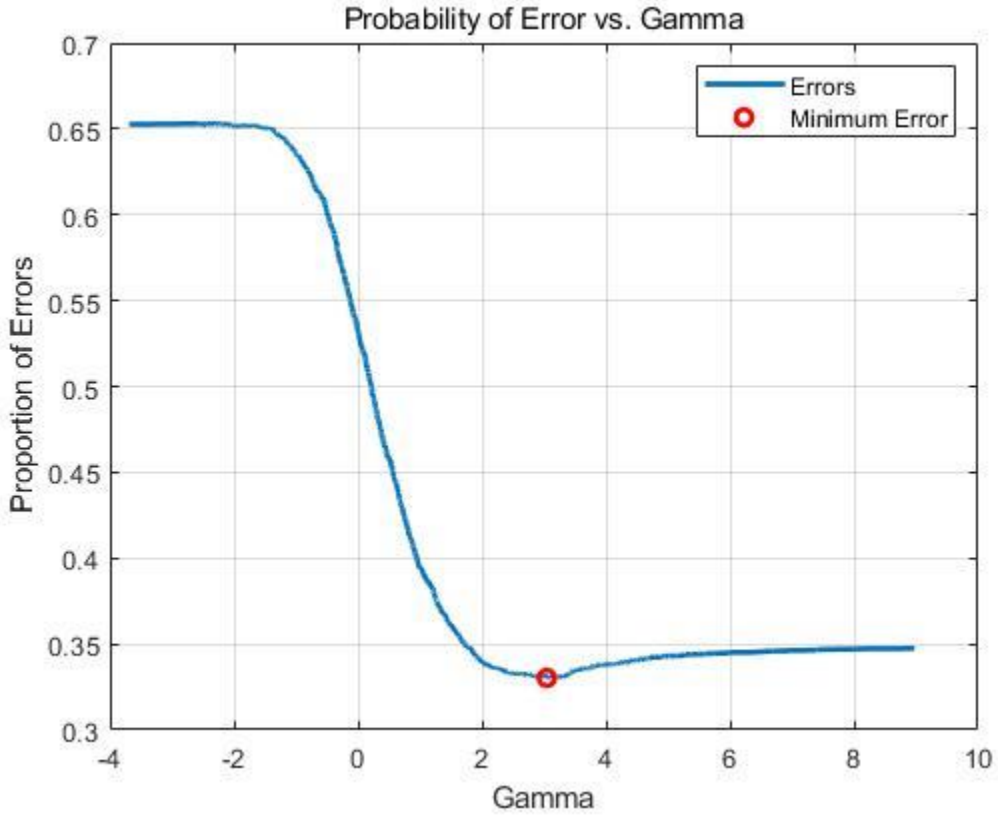

Figure 3: Probability of Error vs. ln (Gamma) for ERM Classification

Part B: Fisher Linear Discriminant Analysis (LDA)

In the part B classification was performed by using a Fisher LDA approach. In this approach the two datasets being classified are projected into a single dimension. The projection maximizes the ratio of the difference in their means to their variances. This minimizes the overlap between the two datasets aiding the classifications process. In this implementation the analysis was performed using the sample mean and covariances of the two sets of data.

1. Fisher LDA Classification rule is shown below.

$$(D=1) \quad W_{LDA}X \gtrless \tau \quad (D=0)$$

Where W_{LDA} is the generalized eigenvector of the scatter matrices SB and SW with the largesteigenvalue as described by the following equations

$$S_W^{-1}S_BW = \lambda W$$
$$S_W = \Sigma_0 + \Sigma_1$$
$$S_B = (\mu_0 - \mu_1)(\mu_0 - \mu_1)^T$$

2. Data from the projection is shown in Figure 4 below. The tau that minimizes the probability of error is also marked. Figure 5 shows the ROC curve. The location of the minimum error determined by a parametric sweep of tau are marked on the plot.

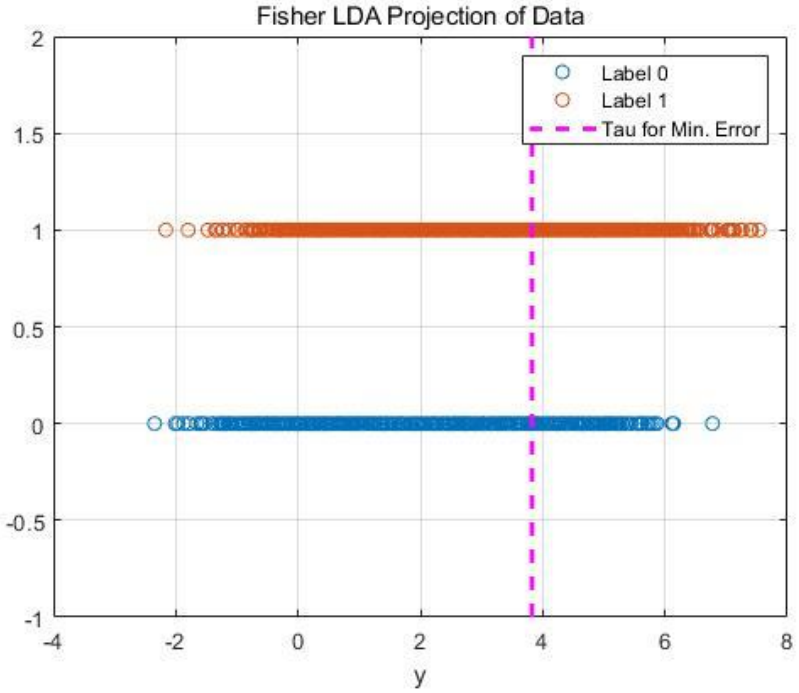

Figure 4: Fisher LDA Projection

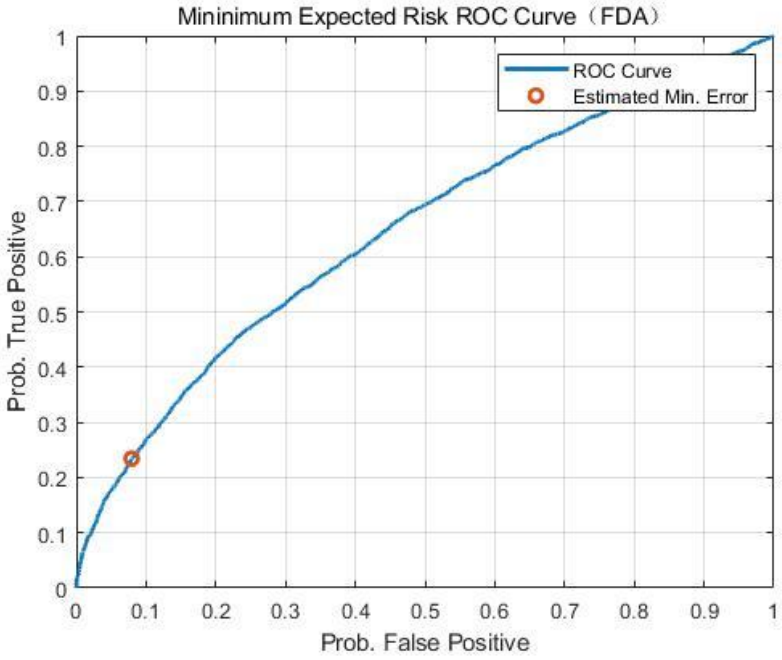

Figure 5: ROC Curve for Fisher LDA

3. A comparison of the probability of errors for the two methods of classification that were performed is shown in Table 2. As can be seen in this table the minimum probability of error was slightly smaller than both the theoretical and analytical estimates of probability of error

generated using the ERM with known covariances method.

Due to the lack of real statistics of the classified data, the accuracy of the classification is not slightly affected. In other cases where the data means are closer to each other, this assumption may lead to worse classification performance compared to the previous classifiers.

Table 2: Probability of Error for All Classification Methods

Method	$Min.P_{error}$
ERM Theoretical	0.4046
ERM Known Covariance	0.3411
Fisher LDA	0.3492

Figure 6 shows a plot of probability of error versus the tau parameter. The shape of this curveis similar to the curves for the ERM based approaches.

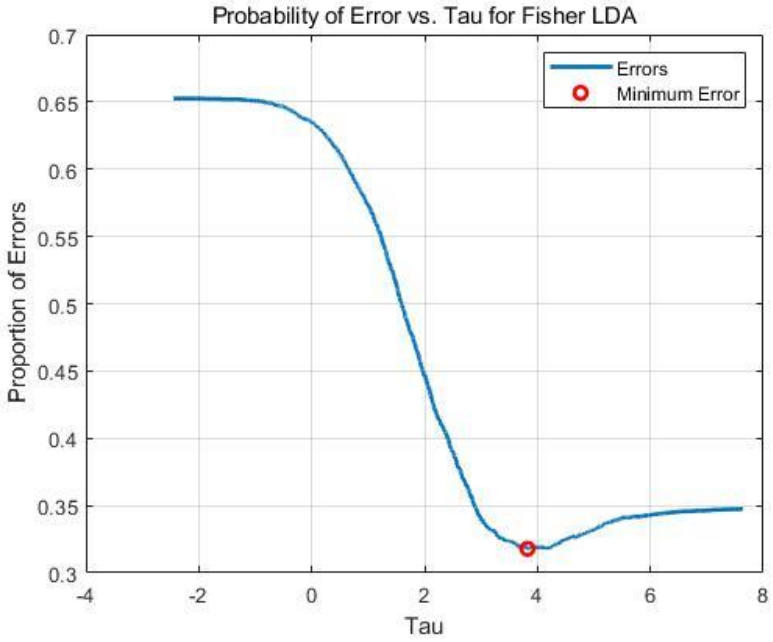

Figure 6: Fisher LDA Probability of Error vs. Tau

Problem 2

A 3-dimensional random Vector X was generated from a mixture of four Gaussian distributions. Data from each distribution was labelled with one of three labels L ϵ {1,2,3}. Class 3 data originates from a mixture of the remaining 2 Gaussian components with equal weights. The class priors were set to 0.3, 0.3, and 0.4 for 3 classes respectively as was specified in the assignment. The means for each of the distributions were set in a line along the x-axis and the mean of each distribution was equidistant from the adjacent means. The covariances for the 4 distributions were set to be scaled versions of the identity matrix. The exact parameters used for each of the 4 sets of data are shown below.

$$\mu_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \Sigma_1 = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix}, \quad p(L=1) = 0.3$$

$$\mu_2 = \begin{bmatrix} 4 \\ 4 \\ 4 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 6 \end{bmatrix}, \quad p(L=2) = 0.3$$

$$\mu_{31} = \begin{bmatrix} 8 \\ 8 \\ 8 \end{bmatrix}, \Sigma_{31} = \begin{bmatrix} 9 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 9 \end{bmatrix}$$

$$\mu_{32} = \begin{bmatrix} 12 \\ 12 \\ 12 \end{bmatrix}, \Sigma_{32} = \begin{bmatrix} 12 & 0 & 0 \\ 0 & 12 & 0 \\ 0 & 0 & 12 \end{bmatrix} \quad , \; p(L=3) = 0.4$$

Part A: Minimum Probability of Error Classification

1. For the first section, 10000 data points were generated, and the labels were tracked. Figure 7 below shows the generated data with data points from each label identified.

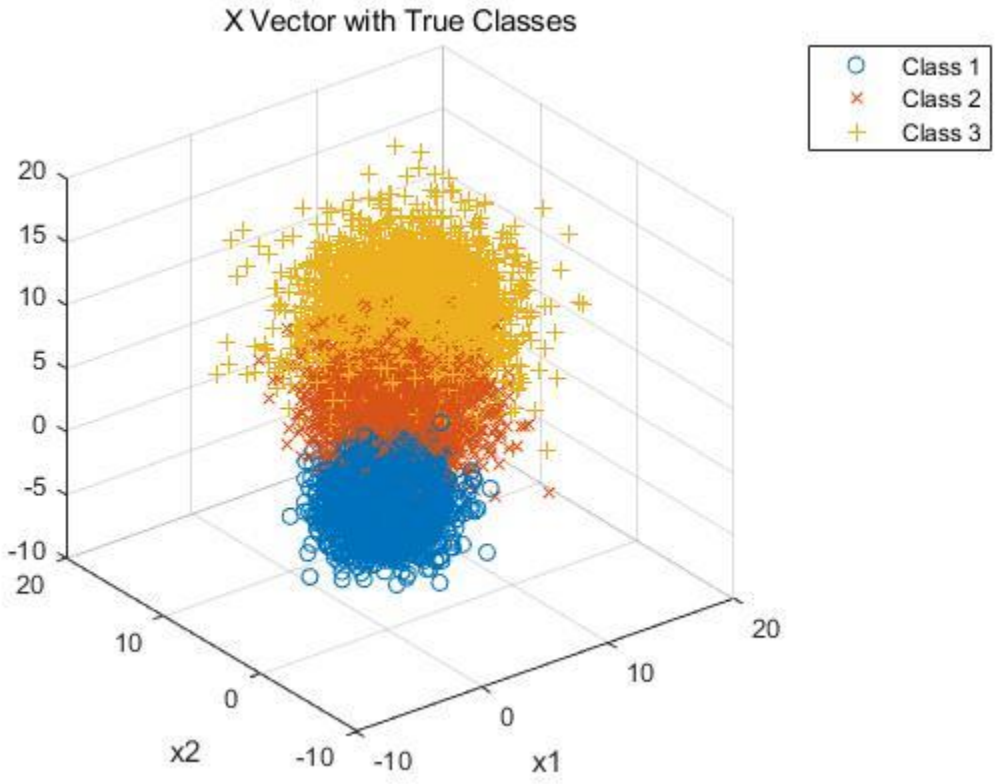

Figure 7: Four Gaussian Distribution with Labels

2. The samples were then analyzed using the following ERM decision rule

$$D(x) = \underset{D \in :1,2,3,4\}}{arg\min} \sum_{l=1}^{C} \lambda_{Dl}\, p(L=l|x)$$

$where\ C\ is\ the\ number\ of\ classes$

$\lambda_{Dl}\ is\ the\ loss\ for\ classifying\ a\ point\ from\ label\ l\ in\ class\ D$

$p(L=l|x)\ is\ the\ class\ posteriors\ which\ are\ calculated\ as\ shown\ below$

$$p(L=l|x) = \frac{p(x|L=l)\,p(L=l)}{p(x)}$$

$where\ p(x|L=l)\ is\ the\ class\ conditional$

$p(L=l)\ is\ the\ class\ prior$

$$And,\ p(x) = \sum_{j=1}^{c} p(x|L=j)\,p(L|j)$$

The cost function used in the classification was chosen to minimize the probability of error and the loss matrix consisting of the λ_{ij} is shown below.

$$\Lambda = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

The data and the resulting classification decisions are shown in Figure 8. The resulting confusion

matrix which shows the probability of classification for each class is shown in Table 3

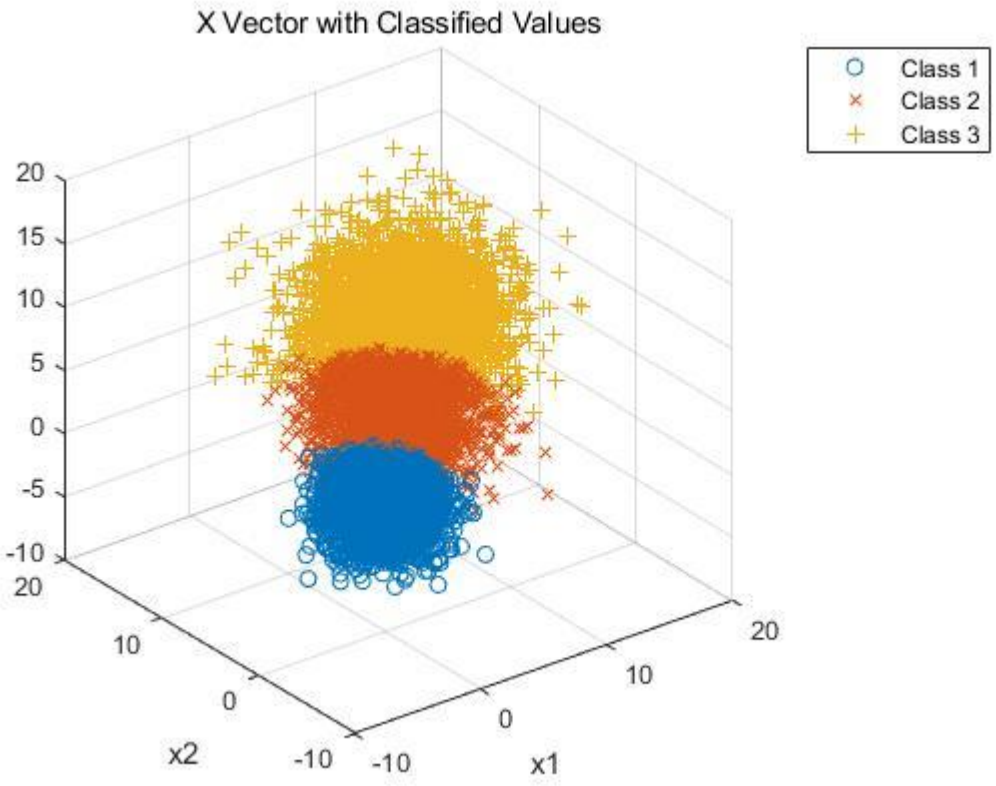

Figure 8: X Vector with Data Classification Decisions for Min. Error Case

Table 3: Confusion Matrix for Minimum Error Loss

		Truth		
		1	2	3
Decision	1	0.9616	0.0541	0
	2	0.0384	0.8345	0.1001
	3	0	0.1114	0.8999

The expected risk for this dataset and cost function was also calculated using the formula shown below. The expected risk was calculated to be 0.31.

$$E\{Loss\} = \frac{1}{N}\sum_{j=1}^{k}\sum_{i=1}^{n}\lambda_{ij}Num\_Decision_{ij}$$

Where N is the number of data points

λ_{ij} is the cost associated with a particular i Decision for a j Label

$Num\_Decision_{ij}$ is the number of decisions for a particular combination of classification and label

3. Figure 9 and Figure 10 are visualizations of the classifications which indicate the 3 classes and which points are correctly and incorrectly categorized.

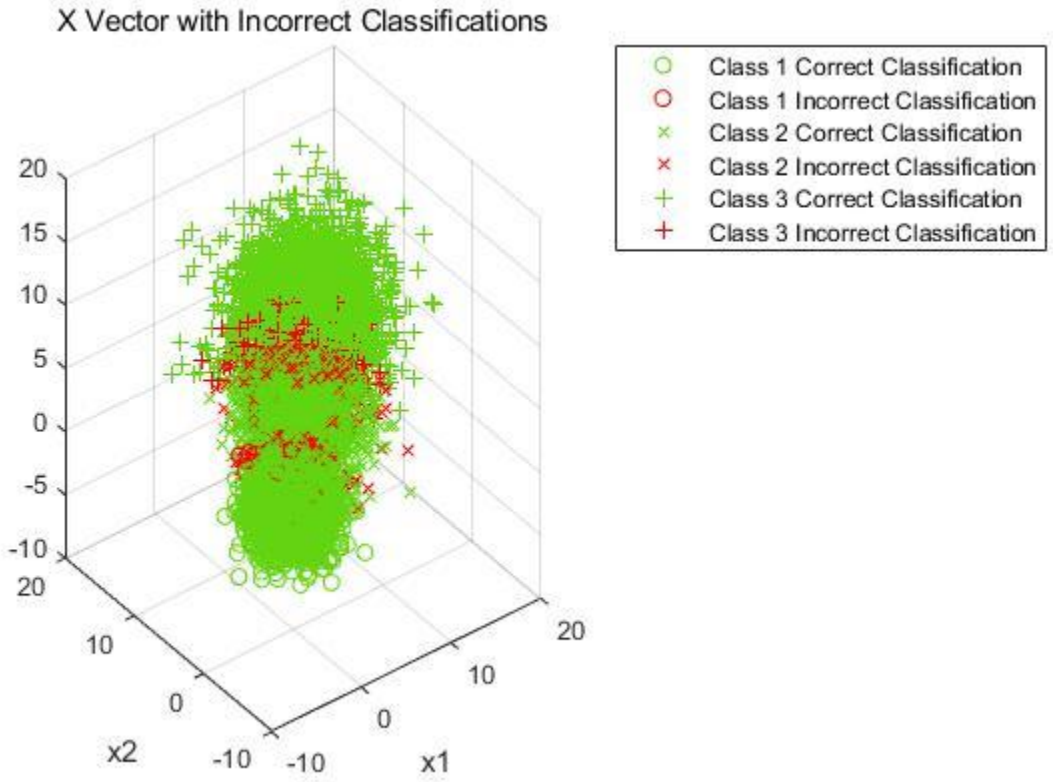

Figure 9: X Vector with Classifications for Min. Error Case

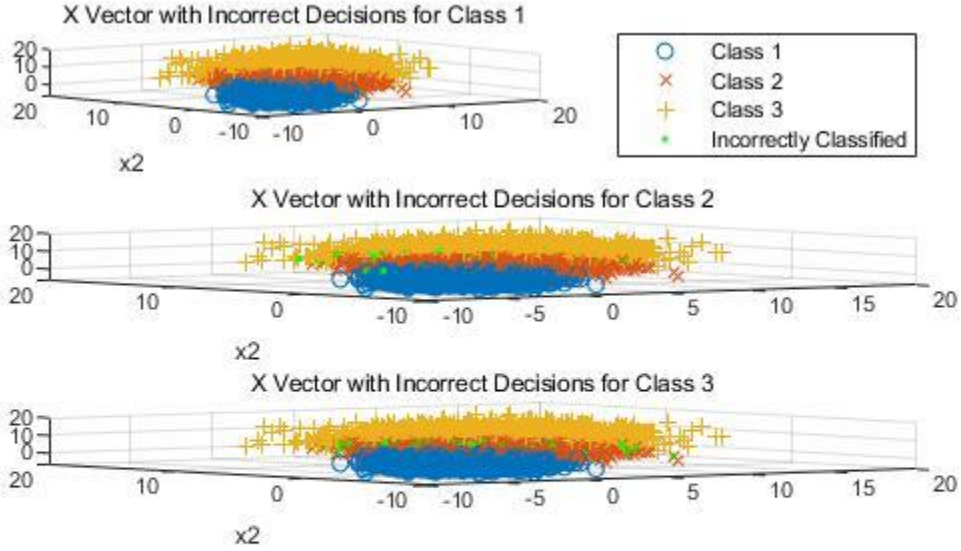

Figure 10: Alternative Visualization of X Vector for Min. Error Case

Part B: Higher Cost for Difference Between Means Classification

In this part the same data use in part A was analyzed using a cost function which increased the cost for misclassification between Gaussian pairs whose means have higher separation. The ERM classification rule with the following loss matrices which respectively care 10 times or 100 times more about not making mistakes when L = 3 is shown below.

$$\Lambda_{10} = \begin{bmatrix} 0 & 1 & 10 \\ 1 & 0 & 10 \\ 1 & 1 & 0 \end{bmatrix} \; and \; \Lambda_{100} = \begin{bmatrix} 0 & 1 & 100 \\ 1 & 0 & 100 \\ 1 & 1 & 0 \end{bmatrix}$$

Figure 11 and figure 12 show the data as classified in care 10 times or 100 times. Figure 13 and Figure 15 show different visualization of the resulting data classifications for the different classes indicating which points were correctly and incorrectly classified when care 10 times. In addition, figure 14 and Figure 16 show different visualization of the resulting data classifications for the different classes indicating which points were correctly and incorrectly classified when care 100 times.

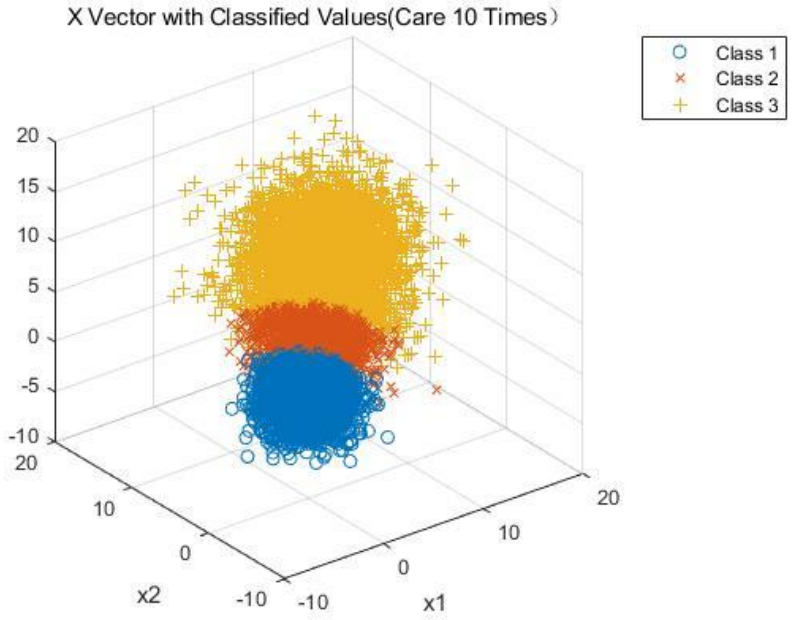

Figure 11: X Vector with Data Classification Decisions for Min. Mean Distance Case (Care 10 Times)

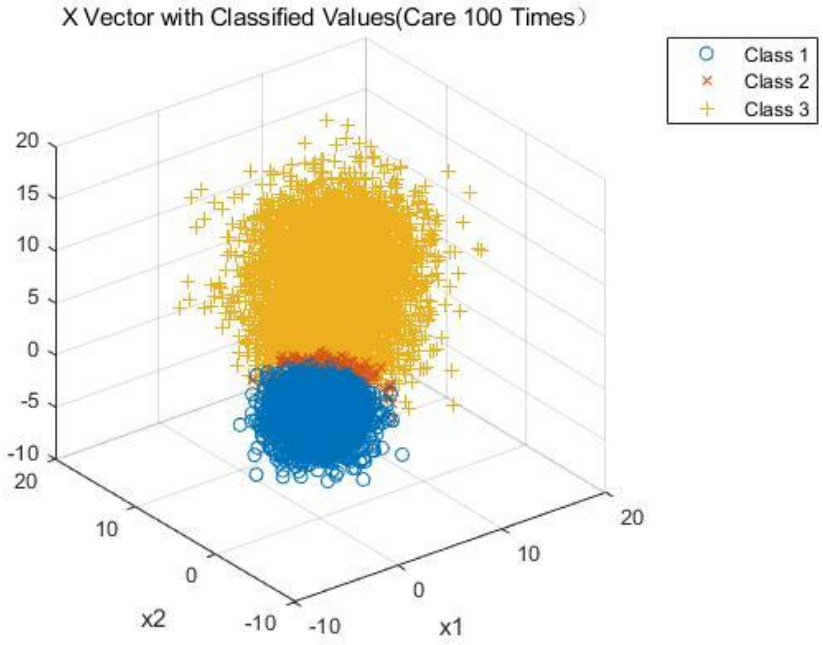

Figure 12: X Vector with Data Classification Decisions for Min. Mean Distance Case (Care 100 Times)

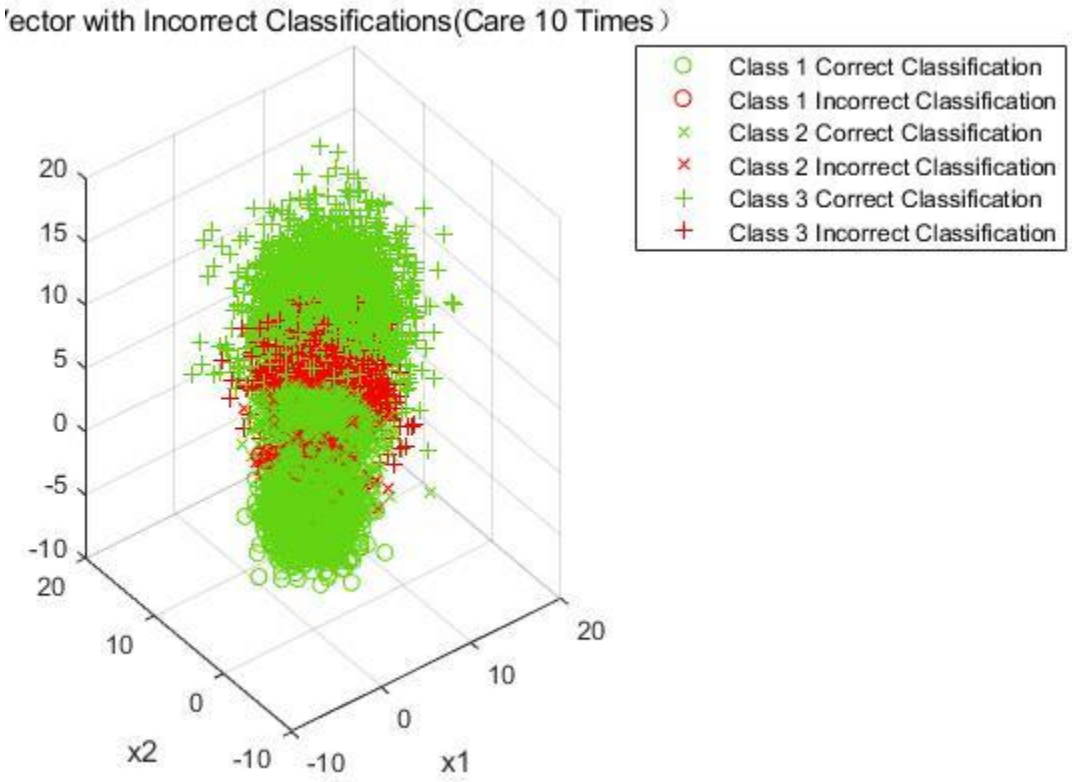

Figure 13: X Vector with Classifications for Min. Mean Distance Case (Care 10 Times)

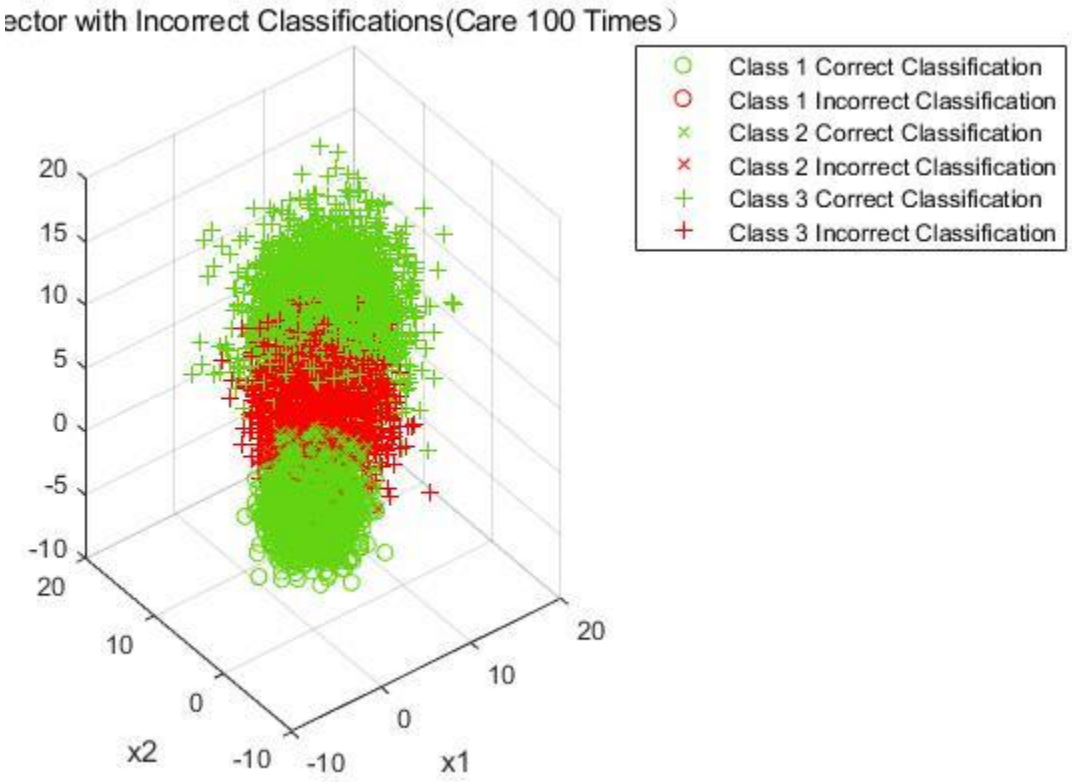

Figure 14: X Vector with Classifications for Min. Mean Distance Case (Care 100 Times)

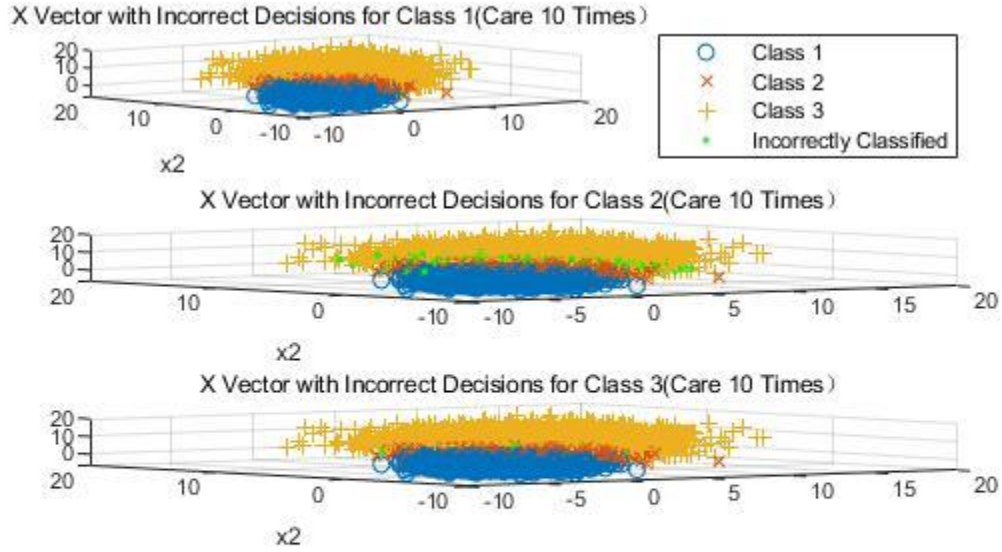

Figure 15: Alternate Visualization of Classifications for Min. Mean Distance Case (Care 10 Times))

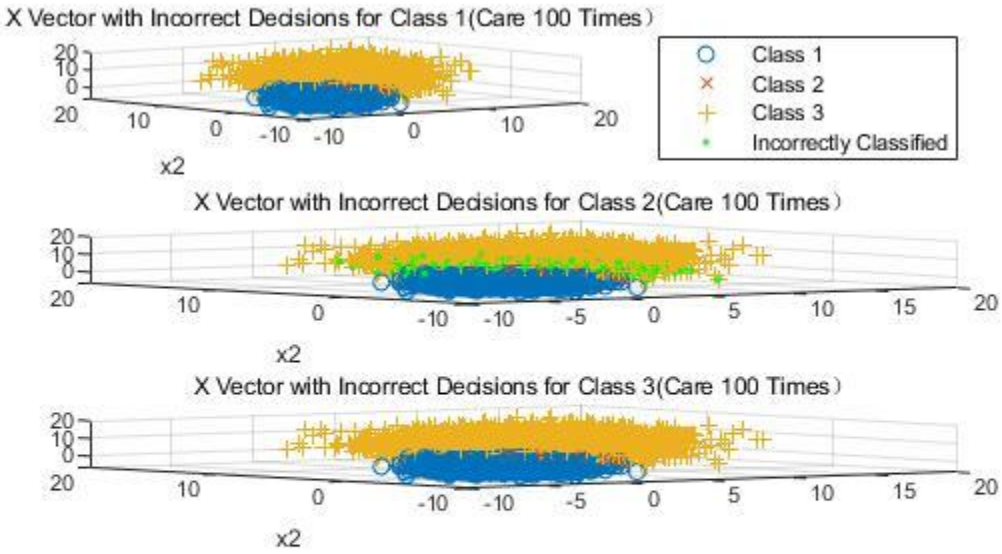

Figure 16: Alternate Visualization of Classifications for Min. Mean Distance Case (Care 100 Times)

The resulting confusion matrix which shows the probability of classification for each class is shown in Table 4 and Table 5. The expected risk was also calculated using the formula in part A and are 1.73(Care 10 Times) and 3.78(Care 100 Times).

Table 4: Confusion Matrix for Min. Distance Loss Matrix (Care 10 Times)

		Truth		
		1	2	3
Decision	1	0.9616	0.0541	0
	2	0.0384	0.5199	0.0166
	3	0	0.4261	0.9834

Table 5: Confusion Matrix for Min. Distance Loss Matrix (Care 100 Times)

		Truth		
		1	2	3
Decision	1	0.9616	0.0541	0
	2	0.0323	0.0771	0
	3	0.0061	0.8689	0.9998

To analyze the effects of changing the loss matrix the confusion matrices generated for the two cases were compared. This was done by subtracting the two matrices to generate a delta confusion matrix where each value indicates the relative change in classification between the two cases. By comparing the values in the rows, it is possible to see which classifications became more or less common due to the change in the loss matrix. The resulting matrix

$$DeltaConfMatrix(care\ 10\ times) = ConfMatrix_{MinDistance(care\ 10\ times)} - ConfMatrix_{MinError}$$

$$DeltaConfMatrix(care\ 10\ times) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -0.3146 & -0.0835 \\ 0 & 0.3146 & 0.0835 \end{bmatrix}$$

$$DeltaConfMatrix(care\ 100\ times) = ConfMatrix_{MinDistance(care\ 100\ times)} - ConfMatrix_{MinError}$$

$$DeltaConfMatrix(care\ 100\ times) = \begin{bmatrix} 0 & 0 & 0 \\ -0.0061 & -0.7574 & -0.0999 \\ 0.0061 & 0.7574 & 0.0999 \end{bmatrix}$$

Delta Confusion Matrix between min. error and min. mean distance classification cases

As can be seen in this delta confusion matrix, when L=3, the cost matrix that uses the cost of increasing the distance between the averages has the effect of increasing the proportion of data points in the class 3. This may be because the loss matrices of care 10 times and care 100 times both act on the class 3. This change is somewhat unintuitive, and this asymmetry in the classification conversion seems to be the result of a priori and covariance asymmetry between labels 1 and 2 and label 3.

To further examine the effects of the change in cost matrix a special test case was performed where the covariances and class priors of the 3 classes were all set to be equivalent to each other. The delta confusion matrix for this case is shown in the table below and as can be seen in this test case with confounders removed the effect of classifications moving from class 2 into class 3 clearly evident and while the shifts still are not entirely symmetric it is possible (likely) that this is due to the probabilistic

nature of the assessment.

$$DeltaConfMatrix(care\ 10\ times\ TEST) = \begin{bmatrix} 0 & 0 & 0 \\ -0.0046 & -0.4954 & -0.1876 \\ -0.0046 & 0.4954 & 0.1876 \end{bmatrix}$$

$$DeltaConfMatrix(care\ 100\ times\ TEST) = \begin{bmatrix} -0.0913 & -0.0530 & -0.0024 \\ -0.0689 & -0.7257 & -0.2031 \\ 0.1602 & 0.7786 & 0.2055 \end{bmatrix}$$

Delta Confusion Matrix between min. error and min. mean distance classification cases

Appendix A: Question 1 Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%EECE5644 Fall 2021
%Homework #1
%Problem #1
%Significant parts of this code were derived from the following sources
%g/Code/ExpectedRiskMinimization.m
%g/Code/fisher_LDA.m
%g/Code/evalGaussian.m
%/g/Practive/EECE5644_2020Spring/EECE5644_2020Spring_TakeHome1Solutions_v20200307.pdf
%/g/Practive/EECE5644_2020Summer2/EECE5644_2020summer2_TakeHome1Solution_Matlab.pdf
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all;
%Initialize Parameters and Generate Data
N = 10000; %Number of data points
n=2; %Dimensions of data
p0 = 0.65; %Prior for label 0
p1 = 0.35; %Prior for label 1
u = rand(1,N)>=p0; %Determine posteriors
%Create appropriate number of data points from each distribution
N0 = length(find(u==0));
N1 = length(find(u==1));
N=N0+N1;
label=[zeros(1,N0) ones(1,N1)];
%Parameters for two classes
mu01 = [3;0];
Sigma01 = [2 ,0 ;
0,1];
mu02 = [0;3];
Sigma02 = [1 ,0 ;
0,2];
mu1 = [2;2];
Sigma1 = [1,0;
0,1];
%Generate data as prescribed in assignment descriptio
r01 = mvnrnd(mu01, Sigma01, N0);
r02 = mvnrnd(mu02, Sigma02, N0);
r0=0.5*r01+0.5*r02;
r1 = mvnrnd(mu1, Sigma1, N1);
%Plot data showing two classes
figure;
plot(r0(:,1),r0(:,2),'.b','DisplayName','Class 0');
axis equal;
hold on;
plot(r1(:,1),r1(:,2),'.r','DisplayName','Class 1');
axis equal;
hold on;
xlabel('x_1');ylabel('x_2');zlabel('x_3');
grid on;
title('Class 0 and Class 1 Data');
legend 'show';
%Combine data from each distribution into a single dataset
x=zeros(N,n);
x(label==0,:)=r0;
x(label==1,:)=r1;
%Part 1: ERM Classification with True Knowledge
```

```
discScore=log(evalGaussian(x' ,mu1,Sigma1)./(0.5.*(evalGaussian(x' ,mu01,Sigma01)+evalGaussian(
x' ,mu02,Sigma02))));
sortDS=sort(discScore);
%Generate vector of gammas for parametric sweep
logGamma=[min(discScore)-eps sort(discScore)+eps];
for ind=1:length(logGamma)
decision=discScore>logGamma(ind);
Num_pos(ind)=sum(decision);
pFP(ind)=sum(decision==1 & label==0)/N0;
pTP(ind)=sum(decision==1 & label==1)/N1;
pFN(ind)=sum(decision==0 & label==1)/N1;
pTN(ind)=sum(decision==0 & label==0)/N0;
%Two ways to make sure I did it right
pFE(ind)=(sum(decision==0 & label==1) + sum(decision==1 & label==0))/N;
pFE2(ind)=(pFP(ind)*N0 + pFN(ind)*N1)/N;
end
%Calculate Theoretical Minimum Error
logGamma_ideal=log(p0/p1);
decision_ideal=discScore>logGamma_ideal;
pFP_ideal=sum(decision_ideal==1 & label==0)/N0;
pTP_ideal=sum(decision_ideal==1 & label==1)/N1;
pFE_ideal=(pFP_ideal*N0+(1-pTP_ideal)*N1)/(N0+N1);
%Estimate Minimum Error
%If multiple minimums are found choose the one closest to the theoretical
%minimum
[min_pFE, min_pFE_ind]=min(pFE);
if length(min_pFE_ind)>1
[~,minDistTheory_ind]=min(abs(logGamma(min_pFE_ind)-logGamma_ideal));
min_pFE_ind=min_pFE_ind(minDistTheory_ind);
end
%Find minimum gamma and corresponding false and true positive rates
minGAMMA=exp(logGamma(min_pFE_ind));
min_FP=pFP(min_pFE_ind);
min_TP=pTP(min_pFE_ind);
%Plot
figure;
plot(pFP,pTP,'DisplayName','ROC Curve','LineWidth',2);
hold all;
plot(min_FP,min_TP,'o','DisplayName','Estimated Min. Error','LineWidth',2);
plot(pFP_ideal,pTP_ideal,'+','DisplayName',...
'Theoretical Min. Error','LineWidth',2);
xlabel('Prob. False Positive');
ylabel('Prob. True Positive');
title('Mininimum Expected Risk ROC Curve');
legend 'show';
grid on; box on;
fprintf('Theoretical: Gamma=%1.2f, Error=%1.2f%%\n',...
    exp(logGamma_ideal),100*pFE_ideal);
fprintf('Estimated: Gamma=%1.2f, Error=%1.2f%%\n',minGAMMA,100*min_pFE);
figure;
plot(logGamma,pFE,'DisplayName','Errors','LineWidth',2);
hold on;
plot(logGamma(min_pFE_ind),pFE(min_pFE_ind),...
'ro','DisplayName','Minimum Error','LineWidth',2);
xlabel('Gamma');
ylabel('Proportion of Errors');
title('Probability of Error vs. Gamma')
grid on;
```

```
legend 'show';
%Part 2: Fisher LDA
%Compute Sample Mean and covariances
mu0_hat=mean(r0)';
mu1_hat=mean(r1)';
Sigma0_hat=cov(r0);
Sigma1_hat=cov(r1);
%Compute scatter matrices
Sb=(mu0_hat-mu1_hat)*(mu0_hat-mu1_hat)';
Sw=Sigma0_hat+Sigma1_hat;
%Eigen decompostion to generate WLDA
[V,D]=eig(inv(Sw)*Sb);
[~,ind]=max(diag(D));
w=V(:,ind);
y=w'*x';
w=sign(mean(y(find(label==1))-mean(y(find(label==0)))))*w;
y=sign(mean(y(find(label==1))-mean(y(find(label==0)))))*y;
%Evaluate for different taus
tau=[min(y)-0.1 sort(y)+0.1];
for ind=1:length(tau)
decision=y>tau(ind);
Num_pos_LDA(ind)=sum(decision);
pFP_LDA(ind)=sum(decision==1 & label==0)/N0;
pTP_LDA(ind)=sum(decision==1 & label==1)/N1;
pFN_LDA(ind)=sum(decision==0 & label==1)/N1;
pTN_LDA(ind)=sum(decision==0 & label==0)/N0;
pFE_LDA(ind)=(sum(decision==0 & label==1)...
+ sum(decision==1 & label==0))/(N0+N1);
end
%Estimated Minimum Error
[min_pFE_LDA, min_pFE_ind_LDA]=min(pFE_LDA);
minTAU_LDA=tau(min_pFE_ind_LDA);
min_FP_LDA=pFP_LDA(min_pFE_ind_LDA);
min_TP_LDA=pTP_LDA(min_pFE_ind_LDA);
%Plot results
figure;
plot(y(label==0),zeros(1,N0),'o','DisplayName','Label 0');
hold all;
plot(y(label==1),ones(1,N1),'o','DisplayName','Label 1');
ylim([-1 2]);
plot(repmat(tau(min_pFE_ind_LDA),1,2),ylim,'m--',...
'DisplayName','Tau for Min. Error','LineWidth',2);
grid on;
xlabel('y');
title('Fisher LDA Projection of Data');
legend 'show';
figure;
plot(pFP_LDA,pTP_LDA,'DisplayName','ROC Curve','LineWidth',2);
hold all;
plot(min_FP_LDA,min_TP_LDA,'o','DisplayName',...
'Estimated Min. Error','LineWidth',2);
xlabel('Prob. False Positive');
ylabel('Prob. True Positive');
title('Mininimum Expected Risk ROC Curve£¨FDA£©');
legend 'show';
grid on; box on;
figure;
plot(tau,pFE_LDA,'DisplayName','Errors','LineWidth',2);
```

```
hold on;
plot(tau(min_pFE_ind_LDA),pFE_LDA(min_pFE_ind_LDA),'ro',...
'DisplayName','Minimum Error','LineWidth',2);
xlabel('Tau');
ylabel('Proportion of Errors');
title('Probability of Error vs. Tau for Fisher LDA')
grid on;
legend 'show';
fprintf('Estimated for LDA: Tau=%1.2f, Error=%1.2f%%\n',...
minTAU_LDA,100*min_pFE_LDA);
%% ====================== Question 1: Functions ====================== %%
%reference g/Code/evalGaussian.m
function g = evalGaussian(x ,mu,Sigma)
%Evaluates the Gaussian pdf N(mu, Sigma ) at each column of X
[n,N] = size(x);
C = ((2*pi)^n * det(Sigma))^(-1/2); %coefficient
E = -0.5*sum((x-repmat(mu,1,N)).*(inv(Sigma)*(x-repmat(mu,1,N))),1);%exponent
g = C*exp(E); %finalgaussianevaluation
end
```

Appendix A: Question 2

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%EECE5644 Fall 2021
%Homework #1
%Problem #2
%Significant parts of this code were derived from the following sources
%g/Code/ERMwithClabels.m
%g/Code/evalGaussian.m
%g/Code/generateDataFromGMM.m
%/g/Practive/EECE5644_2020Summer2/EECE5644_2020summer2_TakeHome1Solution_Matlab.pdf
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all, close all,
C = 4;
N = 10000;
n = 3;
symbols='ox+*';
%Parameters for different classes
params.mean_scaling=4*(0:C-1);
params.Cov_scaling=3*(1:C) ;
gmmParameters.priors = [0.3 0.3 0.4];
gmmParameters.meanVectors(1,:) = params.mean_scaling;
gmmParameters.meanVectors(2,:) = params.mean_scaling;
gmmParameters.meanVectors(3,:) = params.mean_scaling;
%Define loss matrices
lossMatrix={'minErr' 'deltaU10' 'deltaU100'};
lossMatrixA.minErr = ones(C-1,C-1)-eye(C-1);
lossMatrixA.deltaU10= [0 1 10;
1 0 10;
1 1 0];
lossMatrixA.deltaU100= [0 1 100;
1 0 100;
1 1 0];
for ind = 1:C
gmmParameters.covMatrices(:,:,ind) =params.Cov_scaling(ind)*eye(n);
%A = params.Cov_scaling(ind)*eye(n);
%gmmParameters.covMatrices(:,:,ind) = A'*A; % arbitrary covariance matrices
end

% Generate data from specified pdf
[x,labels] = generateDataFromGMM(N,gmmParameters);
for ind = 1:3
Nclass(ind,1) = length(find(labels==ind));
end
C=C-1;
% Shared computation for both parts
for ind = 1:3
pxgivenl(ind,:) =...
evalGaussianPDF(x,gmmParameters.meanVectors(:,ind),...
gmmParameters.covMatrices(:,:,ind));
end
px = gmmParameters.priors*pxgivenl;
classPosteriors = pxgivenl.*repmat(gmmParameters.priors',1,N)./repmat(px,C,1);
%Plot Data with True Labels
figure;
for ind=1:C
plot3(x(1,labels==ind),x(2,labels==ind),x(3,labels==ind),symbols(ind),'DisplayName',...
```

```
['Class ' num2str(ind)]);
hold on;
end
xlabel('x1');
ylabel('x2');
grid on;
title('X Vector with True Classes');
legend 'show';
%Classify Data based on loss values
for ind3=1:length(lossMatrix)
expectedRisksA.(lossMatrix{ind3}) =...
lossMatrixA.(lossMatrix{ind3})*classPosteriors; % Expected Risk for each label (rows) for each
sample (columns)
[~,decisionsA.(lossMatrix{ind3})] =...
min(expectedRisksA.(lossMatrix{ind3}),[],1); % Minimum expected risk decision with 0-1 loss is
the same as MAP
fDecision_ind.(lossMatrix{ind3})=...
(decisionsA.(lossMatrix{ind3})~=labels);%Incorrect classificiation vector
%Confusion Matrix
confMatrix.(lossMatrix{ind3})=zeros(C,C);
for ind=1:C
for ind2=1:C
confMatrix.(lossMatrix{ind3})(ind,ind2)...
=sum(decisionsA.(lossMatrix{ind3})==ind...
& labels==ind2)/Nclass(ind2);
end
end
%Expected Risk
ExpRisk.(lossMatrix{ind3})=...
gmmParameters.priors*diag(expectedRisksA.(lossMatrix{ind3})'...
*confMatrix.(lossMatrix{ind3}));
fprintf('Expected Risk for %s=%1.2f\n',...
lossMatrix{ind3},ExpRisk.(lossMatrix{ind3}));
%Plot Decisions
figure;
for ind=1:C
class_ind=decisionsA.(lossMatrix{ind3})==ind;
plot3(x(1,class_ind),x(2,class_ind),x(3,class_ind),symbols(ind),...
'DisplayName',['Class ' num2str(ind)]);
hold on;
end
xlabel('x1');
ylabel('x2');
grid on;
title('X Vector with Classified Values');
legend 'show';
%Plot Decisions with Incorrect Results as specified in assignment
figure;
for ind=1:C
class_ind=decisionsA.(lossMatrix{ind3})==ind;
plot3(x(1,class_ind & ~fDecision_ind.(lossMatrix{ind3})),...
x(2,class_ind & ~fDecision_ind.(lossMatrix{ind3})),...
x(3,class_ind & ~fDecision_ind.(lossMatrix{ind3})),...
symbols(ind),'Color',[0.39 0.83 0.07],'DisplayName',...
['Class ' num2str(ind) ' Correct Classification']);
hold on;
plot3(x(1,class_ind & fDecision_ind.(lossMatrix{ind3})),...
x(2,class_ind & fDecision_ind.(lossMatrix{ind3})),...
```

```
x(3,class_ind & fDecision_ind.(lossMatrix{ind3})),...
['r' symbols(ind)],'DisplayName',...
['Class ' num2str(ind) ' Incorrect Classification']);
hold on;
end
xlabel('x1');
ylabel('x2');
grid on;
title('X Vector with Incorrect Classifications');
legend 'show';
%Plot Decisions with Incorrect Decisions
figure;
for ind2=1:C
subplot(4,1,ind2);
for ind=1:C
class_ind=decisionsA.(lossMatrix{ind3})==ind;
plot3(x(1,class_ind),x(2,class_ind),x(3,class_ind),symbols(ind),'DisplayName',...
['Class ' num2str(ind)]);
hold on;
end
plot3(x(1,fDecision_ind.(lossMatrix{ind3}) & labels==ind2),...
x(2,fDecision_ind.(lossMatrix{ind3}) & labels==ind2),...
x(3,fDecision_ind.(lossMatrix{ind3}) & labels==ind2),...
'g.','DisplayName','Incorrectly Classified');
ylabel('x2');
grid on;
title(['X Vector with Incorrect Decisions for Class ' num2str(ind2)]);
if ind2==1
legend 'show';
elseif ind2==4
xlabel('x1');
end
end
end
%% ======================= Question 2: Functions ====================== %%
%From g/Code/evalGaussian.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function g = evalGaussian(x ,mu,Sigma)
%Evaluates the Gaussian pdf N(mu, Sigma ) at each column of X
[n,N] = size(x);
C = ((2*pi)^n * det(Sigma))^(-1/2); %coefficient
E = -0.5*sum((x-repmat(mu,1,N)).*(inv(Sigma)*(x-repmat(mu,1,N))),1);%exponent
g = C*exp(E); %finalgaussianevaluation
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%From g/Code/generateDataFromGMM.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x,labels] = generateDataFromGMM(N,gmmParameters)
% Generates N vector samples from the specified mixture of Gaussians
% Returns samples and their component labels
% Data dimensionality is determined by the size of mu/Sigma parameters
priors = gmmParameters.priors; % priors should be a row vector
meanVectors = gmmParameters.meanVectors;
covMatrices = gmmParameters.covMatrices;
n = size(gmmParameters.meanVectors,1); % Data dimensionality
C = length(priors); % Number of components
x = zeros(n,N); labels = zeros(1,N);
% Decide randomly which samples will come from each component
```

```
u = rand(1,N); thresholds = [cumsum(priors),1];
for l = 1:C
indl = find(u <= thresholds(l)); Nl = length(indl);
labels(1,indl) = l*ones(1,Nl);
u(1,indl) = 1.1*ones(1,Nl); % these samples should not be used again
x(:,indl) = mvnrnd(meanVectors(:,l),covMatrices(:,:,l),Nl)';
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%From g/Code/evalGaussianPDF.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function px = evalGaussianPDF(x,mu,Sigma)
% x should have n-dimensional N vectors in columns
n = size(x,1); % data vectors have n-dimensions
N = size(x,2); % there are N vector-valued samples
C = (2*pi)^(-n/2)*det(Sigma)^(-1/2); % normalization constant
a = x-repmat(mu,1,N); b = inv(Sigma)*a;
% a,b are preparatory random variables, in an attempt to avoid a for loop
px = C*exp(-0.5*sum(a.*b,1)); % px is a row vector that contains p(x_i) values
end
```