Assignment 1

DANDAN LIN

NUID:001093902

Question 1:

Part A: Expected Risk Minimization based classification

We can know that:

$$m_{01} = \begin{bmatrix} 3 \\ 0 \end{bmatrix} \quad C_{01} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \quad m_{02} = \begin{bmatrix} 0 \\ 3 \end{bmatrix} \quad C_{02} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \quad m_{1} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \quad C_{1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$P(L=0) = 0.65 \quad P(L=1) = 0.35$$

$$w_1 = w_2 = \frac{1}{2}$$

1.The minimum expected risk classification rule:

$$(D=1)[\frac{g(x|L=1)}{g(x|L=0)}] \underset{<}{>} [\frac{\lambda_{10}-\lambda_{00}}{\lambda_{01}-\lambda_{11}} \bullet \frac{P(L=0)}{P(L=1)}] = \gamma(D=0)$$

Where λ_{00} means the risk of true negative classification, λ_{01} means the risk of false negative classification, λ_{10} means the risk of false positive classification, λ_{11} means the risk of true positive classification. To minimize the probability of wrong classification in 0-1 loss function, the loss of false classification should be 1 and the loss of true classification should be 0, so the rule can be simplified into:

$$(D=1)[\frac{g(x|L=1)}{g(x|L=0)}] \underset{<}{>} [\frac{1-0}{1-0}] \bullet \frac{P(L=0)}{P(L=1)} = \gamma(D=0)$$

$$(D=1)[\frac{g(x|L=1)}{g(x|L=0)}] \underset{<}{>} [\frac{0.65}{0.35}] = 1.86 = \gamma(D=0)$$

2. According to this question, we can use the mean and covariance matrix values to generate 10000 samples, and plot samples in each class as shown below:

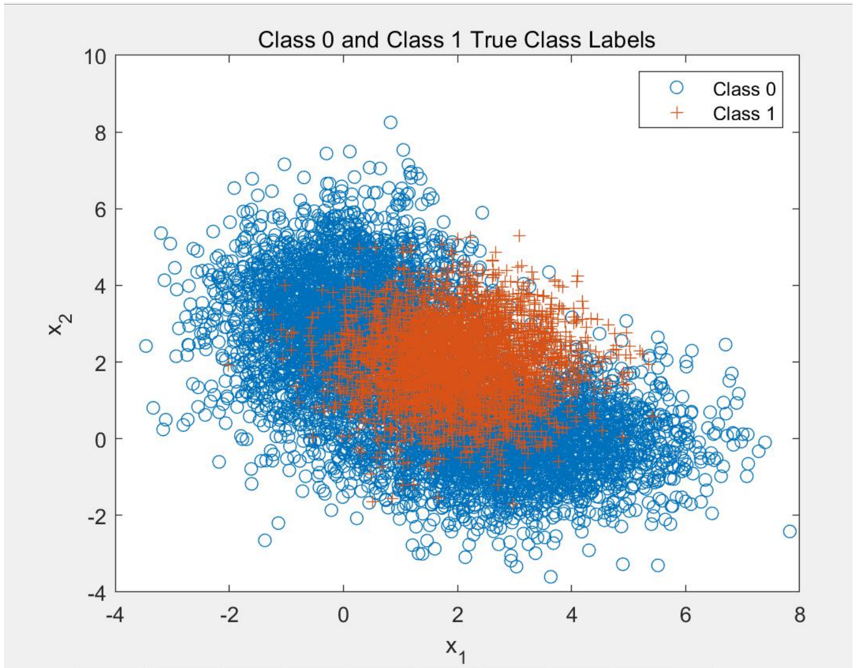

Figure 1. samples in each class

For class 0,we can use gmm function to generate the data by knowing that

$$p(x|L=0) = w_1 g(x|m_{01}, C_{01}) + w_2 g(x|m_{02}, C_{02})$$

The following figure shows the ROC curve where the minimum expected risk classifier applied with γ varies from 0 to ∞ :

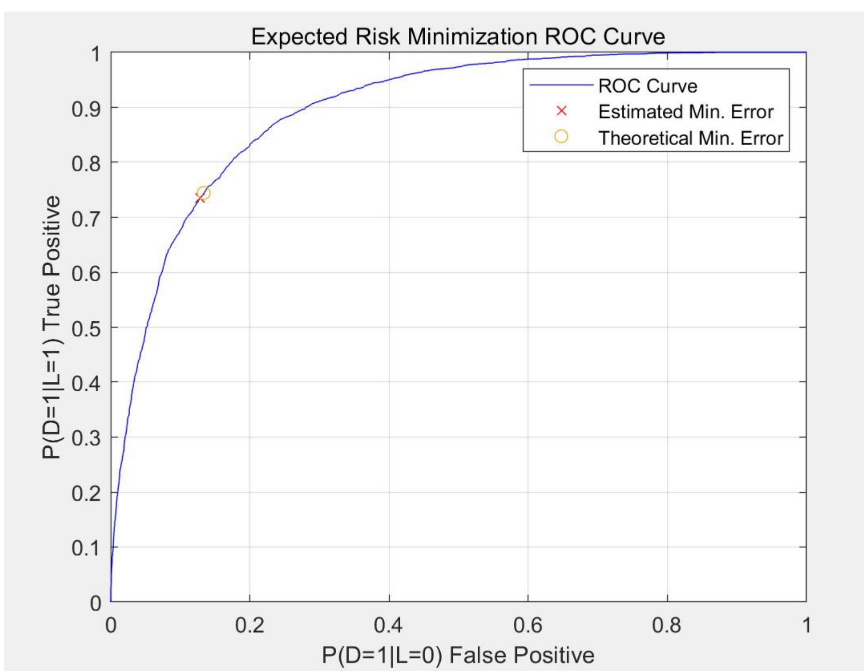

Figure 2. ROC Curve for ERM using true knowledge of class conditional pdfs

3. Using the generated samples, the probability of error was tracked for each threshold γ ,by varying the γ from 0 to ∞ ,and the minimum was found to be 17.57%. at a threshold γ=1.96. The point is marked in the Figure 2 by red.

Since $P_e = 1 - P(D=0|L=0)P(L=0) - P(D=1|L=1)P(L=1)$

We can calculate that the theoretical minimum probability of error is 17.63% at a threshold γ=1.86 which is pointed by yellow and Theoretically, the optimal threshold we calculated above is also 1.86, which corresponds to the minimum error probability 17.63%.

	γ	min p_e
Theoretical	1.86	0.1763
Calculated	1.96	0.1757

Theoretical Results:

Minimum probability of error: 17.63%

Threshold Value : 1.86

Calculated Results:

Minimum probability of error:17.57%

Threshold Value: 1.96

Meanwhile,from the plot below,we can more easily fine the minimum error and its threshold Value.The plot is more visible.

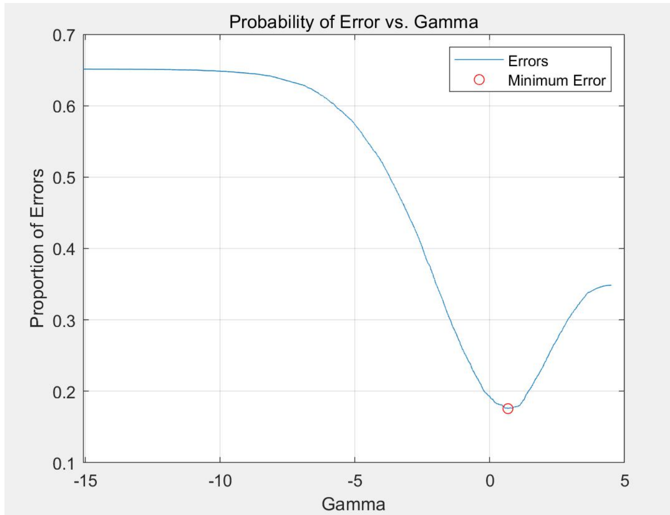

Figure 3. minimum error

Part B:

In the part B of question1, Fisher Linear Discriminant Analysis(LDA) was used to create a classifier and plot ROC curve.

The LDA classification rule:

$$(D=1)\, w_{LDA}{}^{T} x \underset{<}{\overset{>}{}} \tau (D=0)$$

Figure 4 below shows the resulting projection of data:

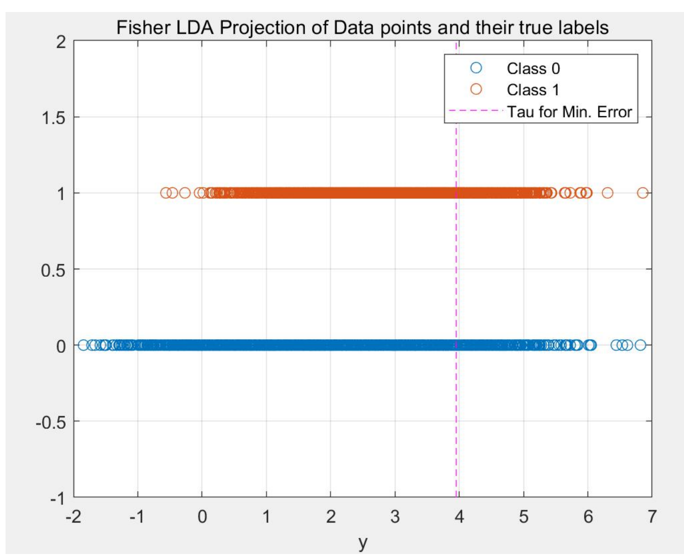

Figure 4. LDA projection

Based on the 10,000 samples generated, the minimum probability of error was calculated to be 0.3417 at a threshold of 3.95.This error and threshold value were calculated by finding the minimum error as the value of τ was changed from $-\infty$ to ∞. The orange circle in Figure 5 below marks this point.

This result makes sense since in the best case scenario, all data points in the smaller of the two classes (in this case, class 1), would be classified wrong, causing a probability of error of 0.35 since class 1 has a 0.35 class prior. It can also be calculated as:

$$P_e = 1 - P(D=0|L=0)P(L=0) - P(D=1|L=1)P(L=1) = 1 - 1 \bullet 0.65 - 0 \bullet 0.35 = 0.35$$

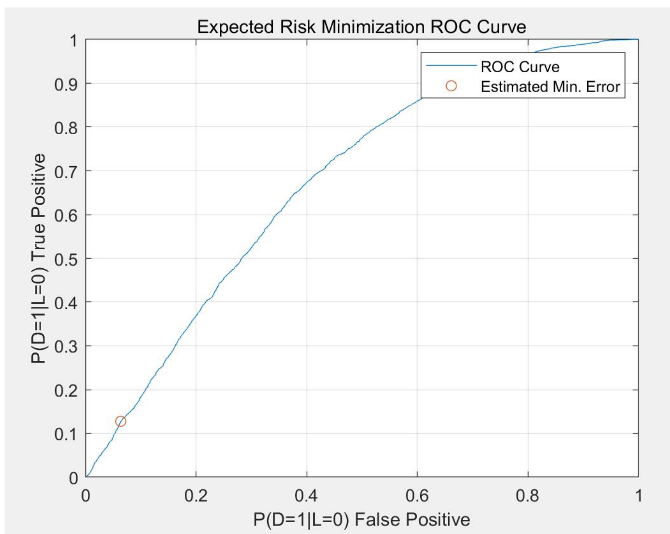

Figure 5. LDA ROC curve

	τ	min p_e
Theoretical	1.86	0.35
Calculated	3.95	0.3417

Calculated Results , LDA

Minimum probability of error =34.17%

Threshold Value (tau)=3.95

So the LDA classification rules result in significantly worse minimum probability of error as compared to the classifier model from Part 1 and the figure 6 below helps the number more visible.

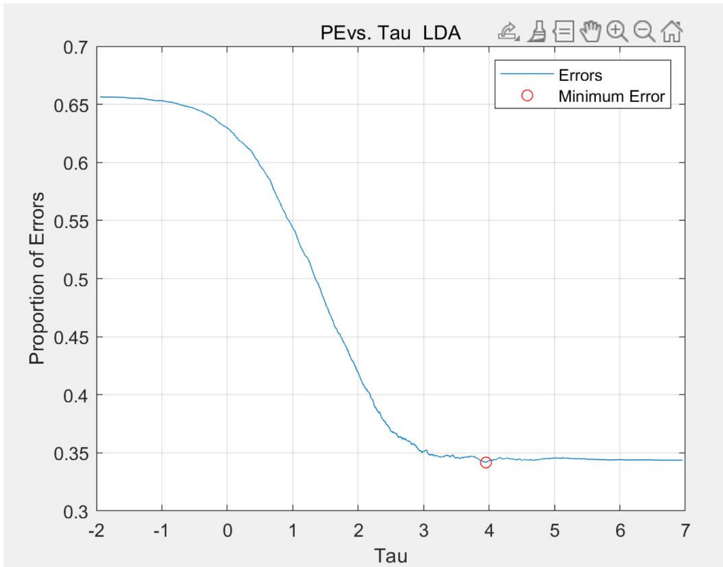

Figure 6. minimum error for fisher LDA

Question 2:

To ensure significant overlap between the class conditional pdfs, the means were set with a distance of twice the average standard deviation of all the Gaussians.

So the condition PDF I could set below:

$$m_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad m_2 = \begin{bmatrix} 5 \\ 5 \\ 5 \end{bmatrix} \quad m_3 = \begin{bmatrix} 9 \\ 9 \\ 9 \end{bmatrix} \quad m_4 = \begin{bmatrix} 13 \\ 13 \\ 13 \end{bmatrix}$$

The covariance matrices were set to be diagonal. Each of the diagonal entries was randomly selected between 0 and 8. The data generated fit these covariance matrices:

$$C_1 = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 6 \end{bmatrix}$$

$$C_2 = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 6 \end{bmatrix}$$

$$C_3 = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 6 \end{bmatrix}$$

$$C_4 = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 6 \end{bmatrix}$$

The class priors were set as:

$P(L=1)=0.3$ $P（L=2)=0.3$ $P(L=3)=0.4$

Part A: Minimum probability of error classification

1.From the data above,the generating the sample data was plotted to show the true data distribution for each class, shown in Figure7.

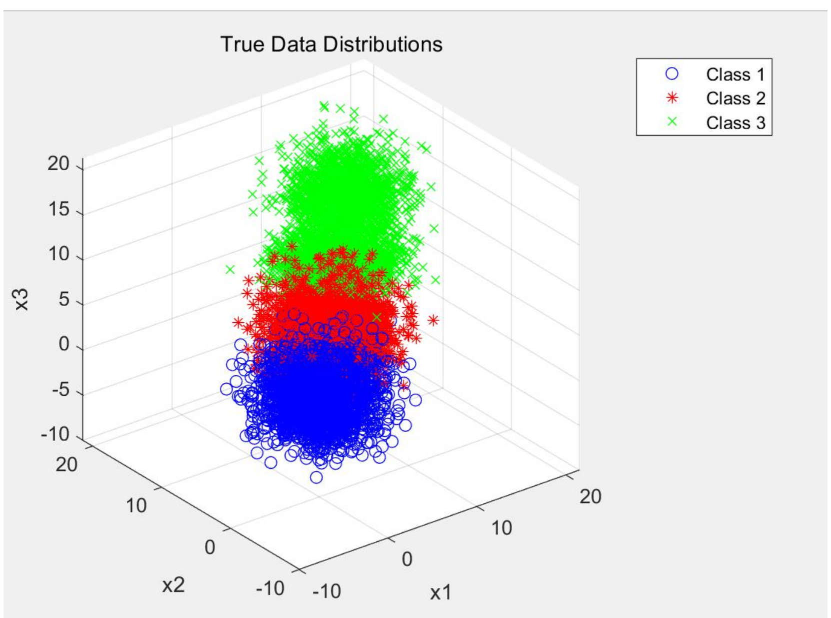

Figure 7. True data distributions for generated samples

2.The ERM decision rule is shown as below:

$$D(x) = \arg\min \sum_{l=1}^{c} \lambda_{dl} p(L = l | x)$$

Where c is the number of classes, λ_{dl} is the loss for classifying a sample in class l as label i. To calculate $p(L = l | x)$, we know that it is the class posterior,which defined by

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

And

$$p(x) = \sum_{l=1}^{c} p(x | L = l) p(L = l)$$

The loss matrix we used to do the classification is 0-1 loss matrix, which is

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

After applying the decision rule, we can get the confusion matrix, which is shown below:

$$\begin{array}{cc} & \text{truth} \\ \text{Decision} & \begin{bmatrix} 0.922 & 0.07 & 0 \\ 0.078 & 0.905 & 0.01 \\ 0 & 0.025 & 0.99 \end{bmatrix} \end{array}$$

3.we get the plot of the correct and incorrect classification using 0-1 loss matrix:

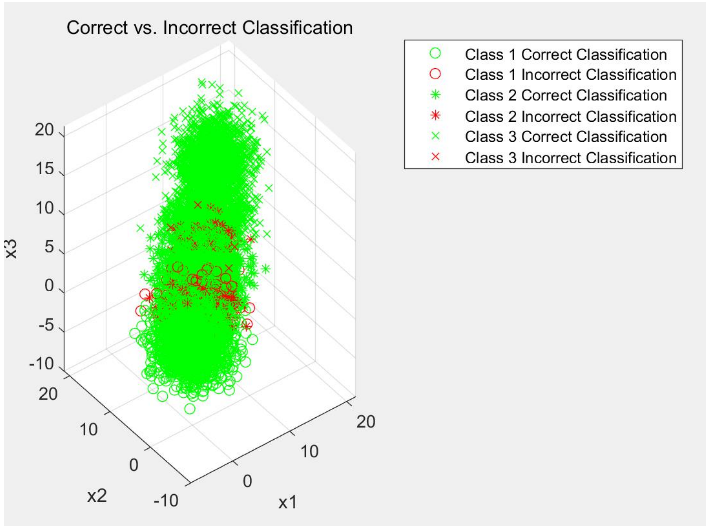

Figure 8. Correct vs incorrect classifications

The plot in Figure 8 could visualize the data for each sample.As we can see, the incorrect classification mostly happened on the adjacent places of each class.

Part B: Using Different Loss Matrix

1. Using $\Lambda_{10} = \begin{bmatrix} 0 & 1 & 10 \\ 1 & 0 & 10 \\ 1 & 1 & 0 \end{bmatrix}$

We can get the confusion matrix:

$$\begin{array}{cc} & \text{truth} \\ \textit{decision} & \begin{bmatrix} 0.921 & 0.0775 & 0 \\ 0.0790 & 0.8595 & 0.0005 \\ 0 & 0.0630 & 0.9995 \end{bmatrix} \end{array}$$

The plot is below in figure 9:

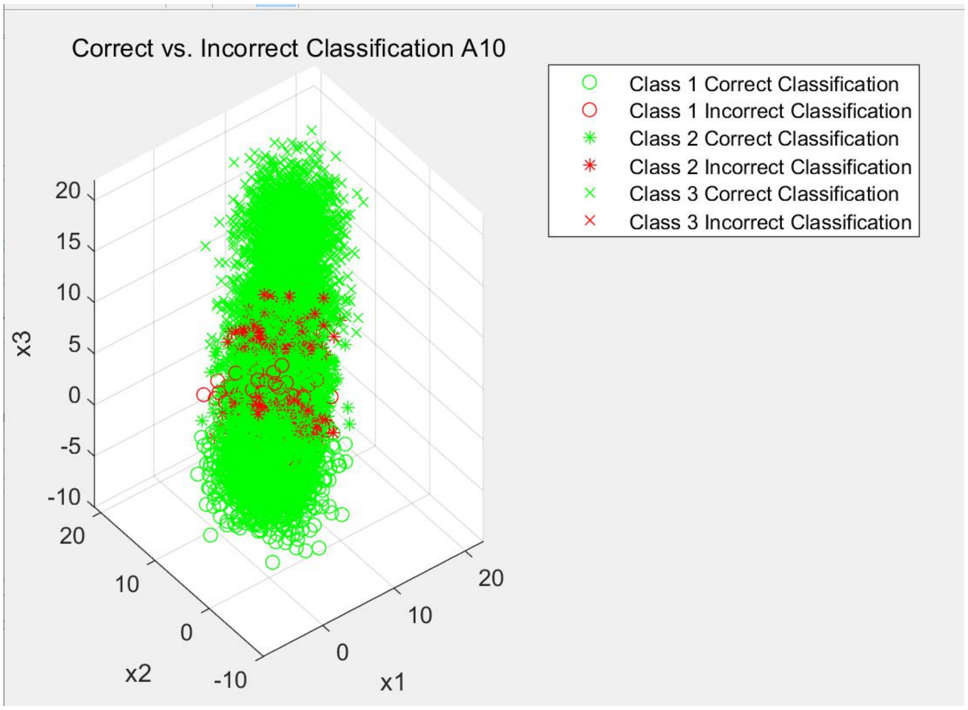

Figure 9. Correct vs incorrect classifications, 10 times

2. Using $\Lambda_{100} = \begin{bmatrix} 0 & 1 & 100 \\ 1 & 0 & 100 \\ 1 & 1 & 0 \end{bmatrix}$

We can get the confusion matrix:

$$\begin{array}{cc} & \text{truth} \\ \textit{decision} & \begin{bmatrix} 0.9253 & 0.0794 & 0 \\ 0.0747 & 0.7145 & 0.0003 \\ 0 & 0.2061 & 0.9997 \end{bmatrix} \end{array}$$

The plot is below in figure 10:

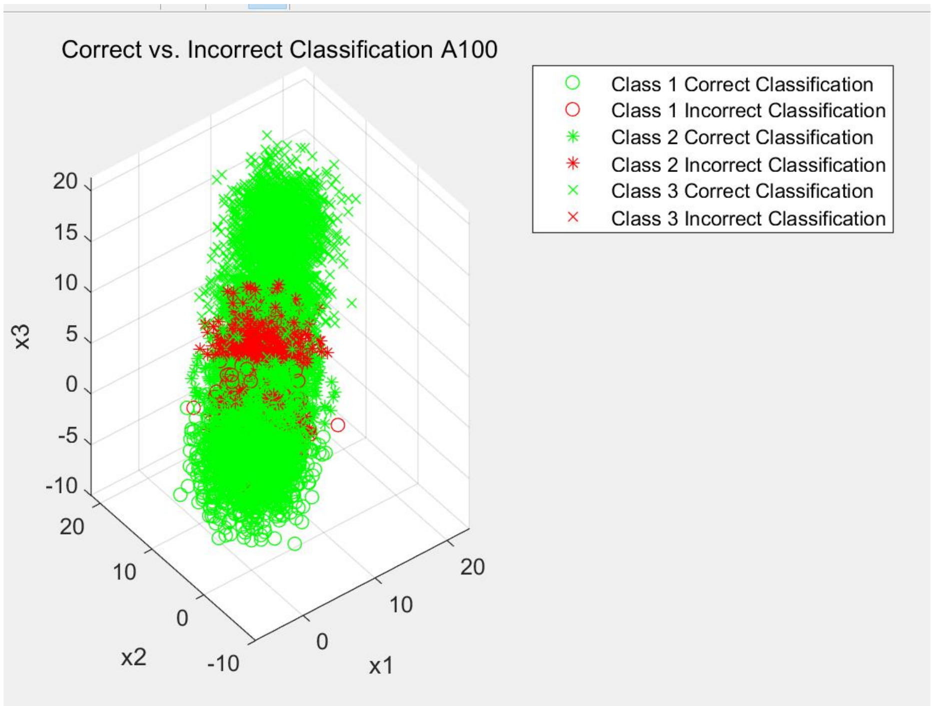

Figure 10. Correct vs incorrect classifications, 100 times

It can be seen from above,With 10 times sensitivity to incorrect classification of Class 3, a majority of the points are classified as Class 3. Over 99.95% of Class 3 points are correctly classified, at the cost of misclassifying much of Classes 1 and 2 as Class 3. With increasing the sensitivity to 100,this effect becomes more clear and over 99.97% of Class 3 points are correctly classified.The decision of the classifier is almost always to choose Class 3. This results almost every Class 3 point being classified correctly, but also most of Class 1 and Class 2 could not be classified correctly.

Also, we can see in the figure 9 and 10, There is more misclassification of Classes 1 and 2, causing the misclassed area to become wider.

Appendix:

Q1:

```
%%=========================Question 1=========================%%
% Dandan lin/001093902
% Code help and example from Prof.Deniz

clear all; close all;clc;
%Initialize Parameters and Generate Data
N = 10000;
n = 2;
p=[0.65,0.35];
%Determine posteriors
label=rand(1, N) >= p(1);

%Label 0
mu0(:,1) = [3;0];
mu0(:,2) = [0;3];
Sigma0(:,:,1)=[2 0;0 1];
Sigma0(:,:,2)=[1 0;0 2];
alpha0=[0.5 0.5];
```

```
%Label 1 SingleGaussianStats
mu1=[2 2]';
Sigma1=[1 0;0 1];
alpha1=1;

%Create appropriate number of data points from each distribution
Nc=[sum(label==0),sum(label==1)];

%Generate data as prescribed in assignment description
x=zeros(n,N);
x(:,label==0)=randGMM(Nc(1),alpha0,mu0,Sigma0);
x(:,label==1)=randGMM(Nc(2),alpha1,mu1,Sigma1);

% Plot true class labels
figure(1);
plot(x(1,label==0),x(2,label==0),'o',x(1,label==1),x(2,label==1),'+');
title('Class 0 and Class 1 True Class Labels')
xlabel('x_1'),ylabel('x_2')
legend('Class 0','Class 1')
%% Part A - ERM with True Knowledge
px0=evalGMM(x,alpha0,mu0,Sigma0);
px1=evalGaussian(x,mu1,Sigma1);
discrimiantScore=log(px1./px0);
sortDS=sort(discrimiantScore);
%Generate vector of gammas for parametric sweep
logGamma=[min(discrimiantScore)-eps sort(discrimiantScore)+eps];
for ind=1:length(logGamma)
    decision=discrimiantScore>logGamma(ind);
```

```
    Num_pos(ind)=sum(decision);
    pFP(ind)=sum(decision==1 & label==0)/Nc(1);
    pTP(ind)=sum(decision==1 & label==1)/Nc(2);
    pFN(ind)=sum(decision==0 & label==1)/Nc(1);
    pTN(ind)=sum(decision==0 & label==0)/Nc(2);
    %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)*Nc(1) + pFN(ind)*Nc(2))/N;
end
%Calculate Theoretical Minimum Error
logGamma_ideal=log(p(1)/p(2));
decision_ideal=discrimiantScore>logGamma_ideal;
pFP_ideal=sum(decision_ideal==1 & label==0)/Nc(1);
pTP_ideal=sum(decision_ideal==1 & label==1)/Nc(2);
pFE_ideal=(pFP_ideal*Nc(1)+(1-pTP_ideal)*Nc(2))/(Nc(1)+Nc(2));
%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, 'b-','DisplayName','ROC Curve');
hold all;
plot(min_FP,min_TP, 'rx','DisplayName','Estimated Min. Error');
plot(pFP_ideal,pTP_ideal,'o','DisplayName',...
    'Theoretical Min. Error');
xlabel('P(D=1|L=0) False Positive');
ylabel('P(D=1|L=1) True Positive');
title('Expected Risk Minimization 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');
hold on;
plot(logGamma(min_pFE_ind),pFE(min_pFE_ind),...
    'ro','DisplayName','Minimum Error');
xlabel('Gamma');
ylabel('Proportion of Errors');
title('Probability of Error vs. Gamma')
grid on;
legend 'show';

%Part B: Fisher LDA
%Compute scatter matrices
```

```
x0=x(:,label==0)';
x1=x(:,label==1)';
mu0_hat=mean(x0);
mu1_hat=mean(x1);
Sigma0_hat=cov(x0);
Sigma1_hat=cov(x1);
%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)/Nc(1);
   pTP_LDA(ind)=sum(decision==1 & label==1)/Nc(2);
   pFN_LDA(ind)=sum(decision==0 & label==1)/Nc(2);
   pTN_LDA(ind)=sum(decision==0 & label==0)/Nc(1);
   pFE_LDA(ind)=(sum(decision==0 & label==1)...
      + sum(decision==1 & label==0))/(Nc(1)+Nc(2));
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,Nc(1)),'o','DisplayName','Class 0');
hold all;
plot(y(label==1),ones(1,Nc(2)),'+','DisplayName','Class 1');
ylim([-1 2]);
plot(repmat(tau(min_pFE_ind_LDA),1,2),ylim,'m--',...
    'DisplayName','Tau for Min. Error');
grid on;
xlabel('y');
title('Fisher LDA Projection of Data');
legend 'show';
figure;
plot(pFP_LDA,pTP_LDA,'DisplayName','ROC Curve');
hold all;
plot(min_FP_LDA,min_TP_LDA,'o','DisplayName',...
    'Estimated Min. Error');
xlabel('P(D=1|L=0) False Positive');
ylabel('P(D=1|L=1) True Positive');
title('Expected Risk Minimization ROC Curve');
legend 'show';
grid on; box on;
figure;
```

```
plot(tau,pFE_LDA,'DisplayName','Errors');
hold on;
plot(tau(min_pFE_ind_LDA),pFE_LDA(min_pFE_ind_LDA),'ro',...
    'DisplayName','Minimum Error');
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);
% Plot Fisher LDA Projection
figure(4);
plot(y(label==0),zeros(1,Nc(1)),'o','DisplayName','Class 0');
hold all;
plot(y(label==1),ones(1,Nc(2)),'o','DisplayName','Class 1');
ylim([-1 2]);
plot(repmat(tau(min_pFE_ind_LDA),1,2),ylim,'m--',...
    'DisplayName','Tau for Min. Error');
grid on;
xlabel('y');
title('Fisher LDA Projection of Data points and their true labels');
legend 'show';

% Plot ROC
figure(5);
plot(pFP_LDA,pTP_LDA,'DisplayName','ROC Curve');
hold all;
```

```
plot(min_FP_LDA,min_TP_LDA,'o','DisplayName',...
    'Estimated Min. Error');
xlabel('P(D=1|L=0) False Positive');
ylabel('P(D=1|L=0) True Positive');
title('Expected Risk Minimization ROC Curve');
legend 'show';
grid on; box on;
% Plot tau
figure(6);
plot(tau,pFE_LDA,'DisplayName','Errors');
hold on;
plot(tau(min_pFE_ind_LDA),pFE_LDA(min_pFE_ind_LDA),'ro',...
    'DisplayName','Minimum Error');
xlabel('Tau');
ylabel('Proportion of Errors');
title('PEvs. Tau  LDA')
grid on;
legend 'show';

%% ======================= Question 1 : Functions ====================== %%
function g = evalGaussian(x,mu,sigma)
% Evaluate the Gaussian pdf N(mu,Sigma) at each column of x
[n,N] = size(x);
C = ((2*pi)^n*det(sigma))^(-1/2); % normalization constant
E = -0.5*sum((x-repmat(mu,1,N)).*(inv(sigma)*(x-repmat(mu,1,N))),1); % exponent
g = C*exp(E);% Gaussian PDF values in a 1xN row vector
function [x,labels] = randGMM(N,alpha,mu,Sigma)
d = size(mu,1); % nality of samples
```

```
cum_alpha = [0,cumsum(alpha)];
u = rand(1,N); x = zeros(d,N); labels = zeros(1,N);
for m = 1:length(alpha)
    ind = find(cum_alpha(m)<u & u<=cum_alpha(m+1));
    x(:,ind) = randGaussian(length(ind),mu(:,m),Sigma(:,:,m));
    labels(ind)=m-1;
end
End
function x = randGaussian(N,mu,Sigma)
% Generates N samples from a Gaussian pdf with mean mu covariance Sigma
n = length(mu);
z = randn(n,N);
A = Sigma^(1/2);
x = A*z + repmat(mu,1,N);
end
function gmm = evalGMM(x,alpha,mu,Sigma)
gmm = zeros(1,size(x,2));
for m = 1:length(alpha) % evaluate the GMM on the grid
    gmm = gmm + alpha(m)*evalGaussian(x,mu(:,m),Sigma(:,:,m));
end
end
```

Q2:

```
%% Setup and Sample Generation
clear all;
close all;
N = 10000; %number of samples
```

```
n = 3; %number of dimensions
C = 3; %number of classes
% Class priors and class conditional distributions
p = [0.3, 0.3, 0.4]; %class priors
sigma(:,:,1) = [6 0 0
 0 6 0
 0 0 6];
sigma(:,:,2) = [6 0 0
 0 6 0
 0 0 6];
sigma(:,:,3) = [6 0 0
 0 6 0
 0 0 rand];
sigma(:,:,4) = [6 0 0
 0 6 0
 0 0 6];
averageStdDev = trace(sum(sqrt(sigma),3))/16; %offset means by 2 std devs
mu(:,1) = [1; 1; 1];
mu(:,2) = [5; 5; 5];
mu(:,3) = [9; 9; 9];
mu(:,4) = [13; 13; 13];
% Data generation and labelling
label = rand(1,N);
for i = 1:length(label)
 if label(i) < p(1)
 label(i) = 1;
 elseif label(i) < (p(2)+p(1))
 label(i) = 2;
```

```
 elseif label(i) < ((p(3)/2)+p(2)+p(1)) %two subclasses for the last class, will be combined later
 label(i) = 3;
 else
 label(i) = 4;
 end
end
NumClass = [sum(label==1),sum(label==2),sum(label==3),sum(label==4)];
x = zeros(n,N);
x(:, label==1) = mvnrnd(mu(:,1), sigma(:,:,1), NumClass(1))';
x(:, label==2) = mvnrnd(mu(:,2), sigma(:,:,2), NumClass(2))';
x(:, label==3) = mvnrnd(mu(:,3), sigma(:,:,3), NumClass(3))';
x(:, label==4) = mvnrnd(mu(:,4), sigma(:,:,4), NumClass(4))';
% Combine labels 2 and 3 into one class under label 2
for i = 1:length(label)
 if label(i) == 4
label(i) = 3;
 end
end
NumClass = [sum(label==1),sum(label==2),sum(label==3)];
% Evaluate class conditional pdfs
pxgivenl(1,:) = mvnpdf(x', mu(:,1)', sigma(:,:,1))';
pxgivenl(2,:) = mvnpdf(x', mu(:,2)', sigma(:,:,2))';
pxgivenl(3,:) = .5*mvnpdf(x', mu(:,3)', sigma(:,:,3))' + .5*mvnpdf(x', mu(:,4)',
sigma(:,:,4))'; %two distributions for class 3
% Find class posteriors
px = p*pxgivenl; %total probability
plgivenx = pxgivenl.*repmat(p',1,N)./repmat(px,C,1); %class posterior functions
% Loss matrix, 0-1 loss provides minimum probability of error
lossMatrix = ones(3,3)-eye(3);
```

```
expectedRisks = lossMatrix*plgivenx;

[~,decisions] = min(expectedRisks,[],1);

% Make confusion matrix and plot data

figure

shapes = ['o','*','x'];

for i = 1:C %each decision

 for j = 1:C %each class label

 confusionMatrix(i,j) = sum(decisions==i & label==j)/NumClass(j);

 if i == j

 scatter(i,j) = scatter3(x(1,decisions==i & label==j),x(2,decisions==i &
label==j),x(3,decisions==i & label==j),'g',shapes(j),'DisplayName', ['Class 'num2str(j) ' Correct
Classification']);

 hold on

 else

 scatter(i,j) = scatter3(x(1,decisions==i & label==j),x(2,decisions==i &
label==j),x(3,decisions==i & label==j),'r',shapes(j),'DisplayName', ['Class 'num2str(j) ' Incorrect
Classification']);

 hold on

 end

 end

end

title('Correct vs. Incorrect Classification')

legend([scatter(1,1) scatter(2,1) scatter(2,2) scatter(3,2) scatter(3,3) scatter(1,3)])

xlabel('x1')

ylabel('x2')

zlabel('x3')

hold off
```