EECE 5644: Machine Learning / Pattern Recognition

Assignment 1

Jiayun Xin

NUID: 001563582

College of Engineering

Northeastern University Boston, Massachusetts

Spring, 2022

Q1

Code for this question can be found in Appendix A.

Part A

This problem produced a 10000 sample, 4-dimensional real valued random vector X with the following characteristics: p(x)=p(x|L=0)P(L=0)+p(x|L=1)p(L=1) where L is the true class label that indicates which class-label-conditioned pdf generates the data. The class conditional PDFs are p(x|L=0)=g(x|mo,Co) and p(x|L=1)=g(x|m1,C1) where m is the mean vector and C is the covariance matrix.

The particular parameters and class priors are shown below. A plot of the vector X generated using these parameters is shown in Figure 1.

$$\mathbf{m}_0 = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \quad \mathbf{C}_0 = \begin{bmatrix} 2 & -0.5 & 0.3 & 0 \\ -0.5 & 1 & -0.5 & 0 \\ 0.3 & -0.5 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \quad \mathbf{m}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad \mathbf{C}_1 = \begin{bmatrix} 1 & 0.3 & -0.2 & 0 \\ 0.3 & 2 & 0.3 & 0 \\ -0.2 & 0.3 & 1 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

$P(L = 0) = 0.7$ and $P(L = 1) = 0.3$

1.

Part A

1.

$$(D=1)\ \frac{P(x|L_1)}{P(x|L_0)} \gtrless \frac{\lambda_{10}-\lambda_{00}}{\lambda_{01}-\lambda_{11}} * \frac{P(L_0)}{P(L_1)} = \gamma \quad (D=0)$$

$$(D=1)\ \frac{P(x|L_1)}{P(x|L_0)} \gtrless \frac{1-0}{1-0} * \frac{0.7}{0.3} = 2.33 \quad (D=0)$$

2.

Multiple gamma values were used and ROC is shown below.
The estimated and theoretical minimum error point are marked on the plot.

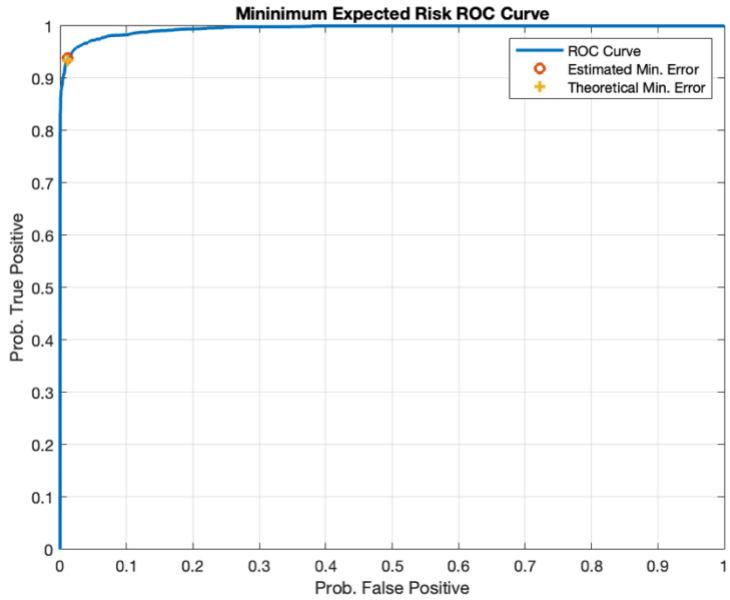

Figure 1: ROC Curve for ERM Classification

3.

	Gamma	Min.P_{error}
Theoretical	2.33	2.82%
Emprical	1.68	2.70%

Table 1

The table shows that empirically selected gamma value has a lower minimum P(error) than theoretically optimal threshold.

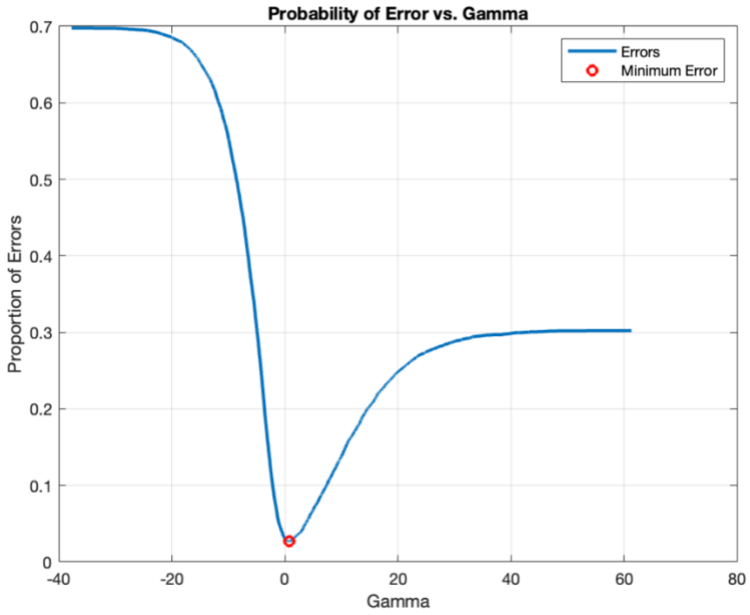

Figure 2

Figure 3 displays the values of error proportion as gamma grows and the point with the lowest error proportion is marked. When the gamma is set as the minimum value, input can be classified with the best result.

Part B

1.

The values remain the same with part A.

Part A

1.

$$(D=1)\ \frac{P(x|L_1)}{P(x|L_0)} \gtrless \frac{\lambda_{10}-\lambda_{00}}{\lambda_{01}-\lambda_{11}} * \frac{P(L_0)}{P(L_1)} = \gamma \quad (D=0)$$

$$(D=1)\ \frac{P(x|L_1)}{P(x|L_0)} \gtrless \frac{1-0}{1-0} * \frac{0.7}{0.3} = 2.33 \quad (D=0)$$

2.

Like part A, multiple gamma values were used and ROC is shown below.
The theoretical minimum error point is marked on the plot.

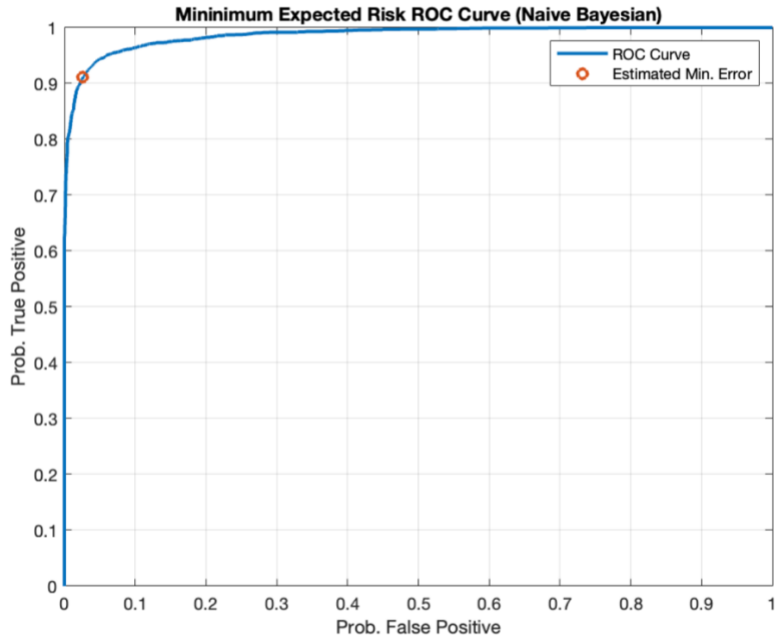

Figure 3

3.

The table 2 shows the comparison of threshold between the theoretical values and naïve Bayesian. As we can see, they produced similar gamma value, but the minimum P_{error} of naïve Bayesian is a little higher than the theoretical because of the lack of true statistics of data being classified.

Comparing the figure 2 and 4, they have almost the same shape.

	Gamma	Min.P_{error}
Theoretical	2.33	2.82%

Emprical	1.68	2.70%
Naïve Bayesian	1.78	4.48%

Table 2

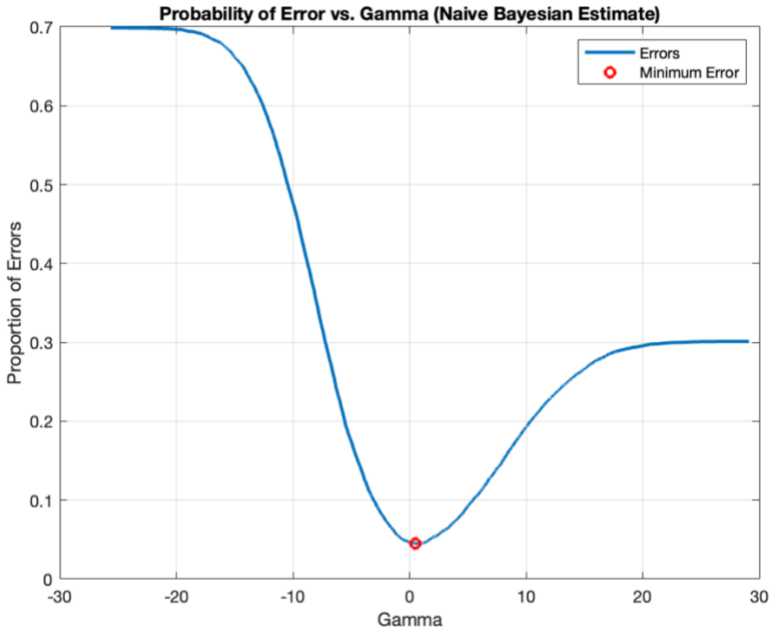

Figure 4

Part C

1.

Fisher LDA classification rule:

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

W_{LDA}: generalized eigendecom-position of within and between class scatter matrices

2.

The projection figure is shown below and the tau for minimum error is mark in figure.
The ROC curve is shown in figure 6.
The theoretical minimum error point is marked on the plot.

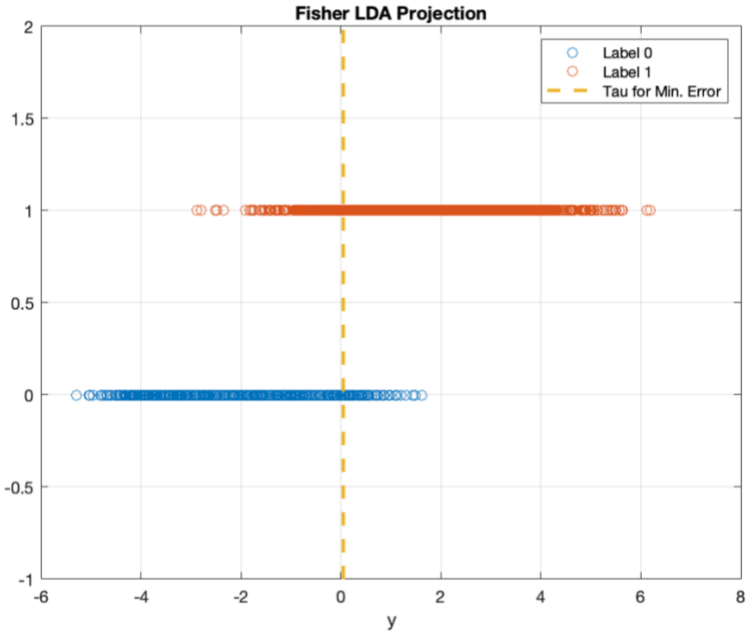

Figure 5

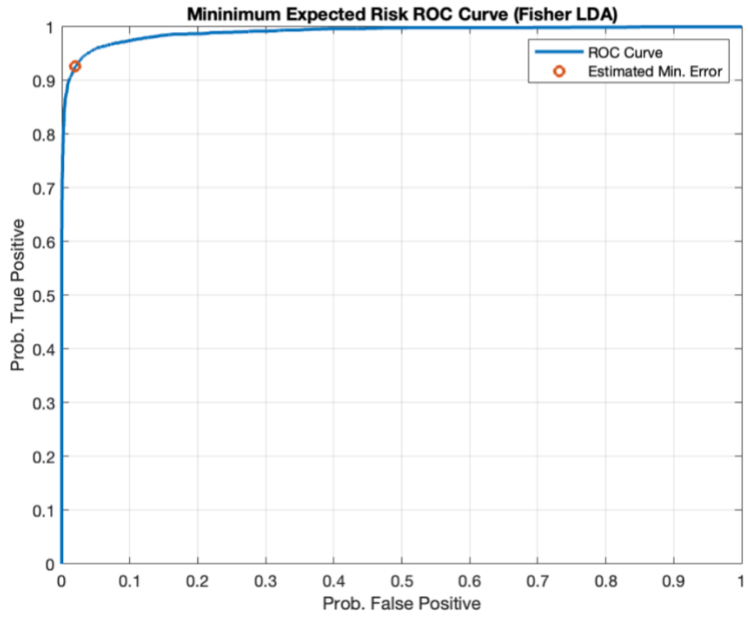

Figure 6

3.

The table 3 lists the minimum P_{error} results of all three methods. As we can see, Fisher LDA has a lower probability of error than Naïve Bayesian method and a little higher error probability than ERM method.

	$Min.P_{error}$
Theoretical	2.82%
Emprical	2.70%
Naïve Bayesian	4.48%
Fisher LDA	3.63%

Table 3

The shape of figure 7 has almost the same shape with figure 2 and 4.

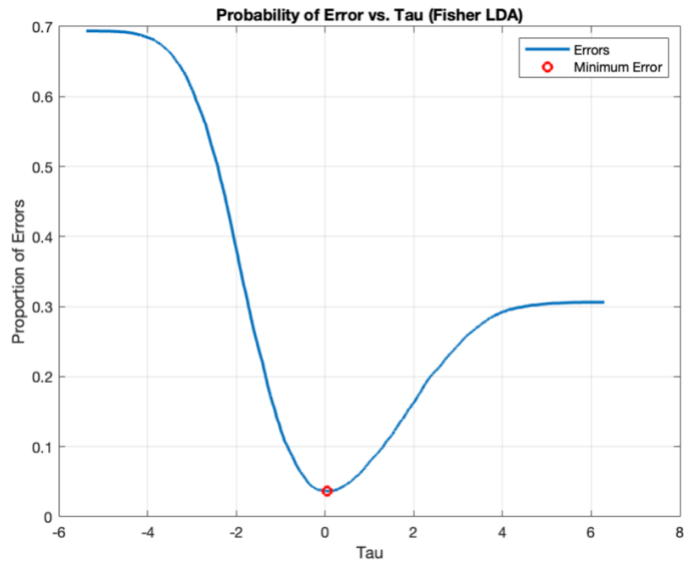
Probability of Error vs. Tau (Fisher LDA)
Errors
Minimum Error
Proportion of Errors
Tau
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-6
-4
-2
0
2
4
6
8

Figure 7

Q2

Code for this question can be found in Appendix B.

Part A

1.

mean vector m: To ensure overlap between class-conditional pdfs, the mean vectors were set by twice average standard deviation of all Gaussians.

covariance matrix c: it was set to be diagonal.

```
val(:,:,1) =

    0.8147         0         0
         0    0.9058         0
         0         0    0.1270


val(:,:,2) =

    0.9134         0         0
         0    0.6324         0
         0         0    0.0975


val(:,:,3) =

    0.2785         0         0
         0    0.5469         0
         0         0    0.9575


val(:,:,4) =

    0.9649         0         0
         0    0.1576         0
         0         0    0.9706
```

The figure 8 shows the true data distribution of the three classes.

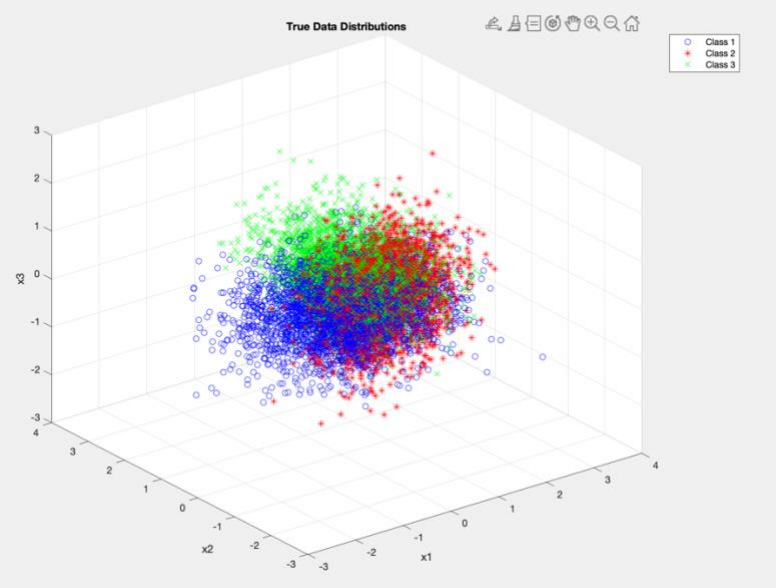

Figure 8

2.
The following decision rule was used to classify the data samples:

$$D(x) = \min_{i\epsilon\{1,2,3\}} \sum_{j=1}^{C} \lambda_{ij}\, p(L = j|x)$$

Where C is the number of classes, lambda is the loss and p mean a pdf. 0-1 loss is selected as follow:

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

The confusion table is as below:

	1	2	3
1	0.7152	0.1370	0.1473
2	0.0726	0.5661	0.1282
3	0.2122	0.2969	0.7245

Table 4

3.
The visualization of the data is shown as figure 9. As we can see, different marker shape marks different class. The green marker relates to correct classification and red marker is wrong classification.

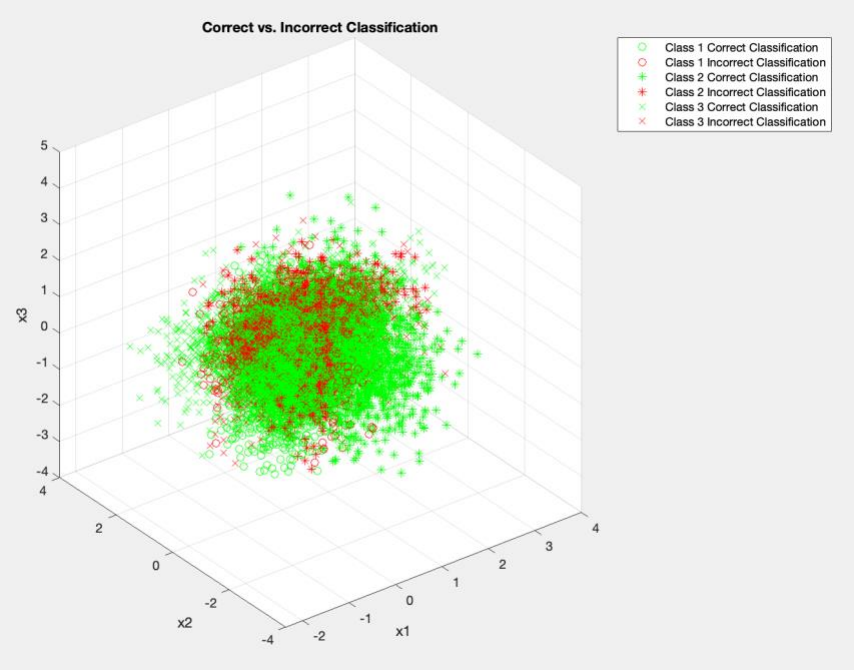

Figure 9

Part B

After repeating the ERM classification rule and resetting loss matrices of 10 and 100 times about not making mistakes when L = 3, the table 5, table 6, figure 10 and figure 11 show the results respectively. We can see that there are more misclassified points as loss matrices of 'L = 3' cares more and more. The overall probability of error increases from 27% to 43% to 54 for 1, 10 and 100 times sensitivity.

Depends on the table 5 and 6, we can see that more and more data was classified to class 3 and class 3 had a good classified-accuracy. But it causes more class 1 and class 2 data misclassified because the relevant confusion value become lower.

	1	2	3
1	0.4586	0.0313	0.0037
2	0.1058	0.7759	0.0042
3	0.4356	0.1927	0.9921

Table 5: confusion matrix when lambda is 10

	1	2	3
1	0.3525	0.0217	2.4740e-04
2	0.0882	0.6489	4.9480e-04
3	0.5593	0.3294	0.9993

Table 6: confusion matrix when lambda is 100

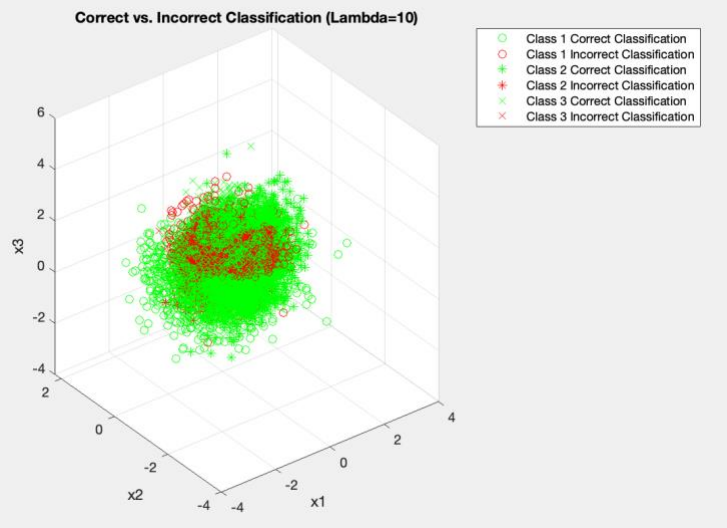

Figure 10

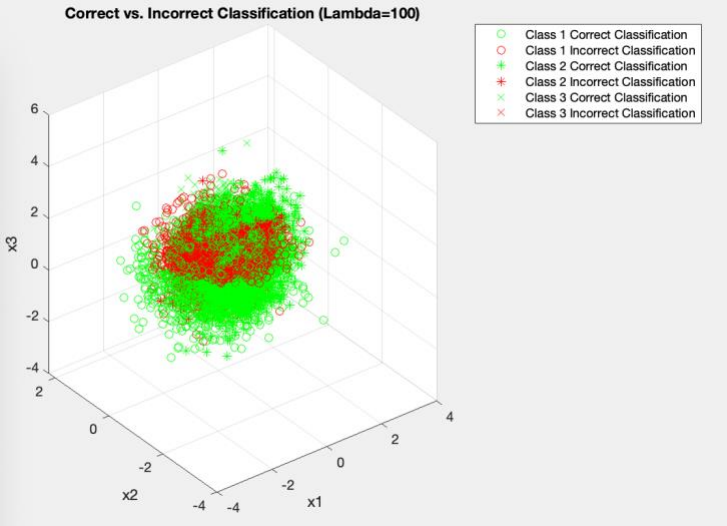
Correct vs. Incorrect Classification (Lambda=100)
Class 1 Correct Classification
Class 1 Incorrect Classification
Class 2 Correct Classification
Class 2 Incorrect Classification
Class 3 Correct Classification
Class 3 Incorrect Classification
x3
x2
x1

Figure 11

Q3

Code for this question can be found in Appendix C.

I split the whole question into two 2 parts and talk about wine quality dataset and human activity recognition dataset separately.

A regularization term is used to covariance to ensure that the regularized covariance matrix $C_{regularized}$ has all eigenvalues larger than the parameter. The regularization term formula is:

$$\mathbf{C}_{Regularized} = \mathbf{C}_{SampleAverage} + \boldsymbol{\lambda}.$$

Where lambda is set as 0.1

Part A

The probability of error is 54.49%. The count error is 2669. The confusion matrix is shown as table 7 in which the first column represents the decision class label, and the first row relates to the true class label. From the confusion matrix, there are many 0 values in the table. Class 0, 1, 2 and 10 do not show up. Class 4, 8 and 9 have mall class priors, so they are not considered. Class 7 with 0.1797 class prior has a large overlap with other classes. Class 5 and 6 are the two classes with highest class prior, so they are chosen to decide class label.

	True Class Label											
Decision		0	1	2	3	4	5	6	7	8	9	10
	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	0	0.2500	0.0123	0.0034	0.0036	0.0023	0.0229	0	0
	4	0	0	0	0	0	0	0	0	0	0	0
	5	0	0	0	0.2000	0.1963	0.1935	0.1128	0.0466	0.0343	0	0
	6	0	0	0	0.5500	0.7914	0.8030	0.8835	0.9511	0.9429	1	0
	7	0	0	0	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	0	0	0	0
	10	0	0	0	0	0	0	0	0	0	0	0

Table 7: confusion matrix for white wine quality dataset

The Principal Component Analysis was used, and the result is shown as figure 12. As we can see, the shape of it like a triangle but Gaussian distribution. Different classes are overlapped because the mean (7.65) of between two classes is smaller than the average standard deviation (10.34).

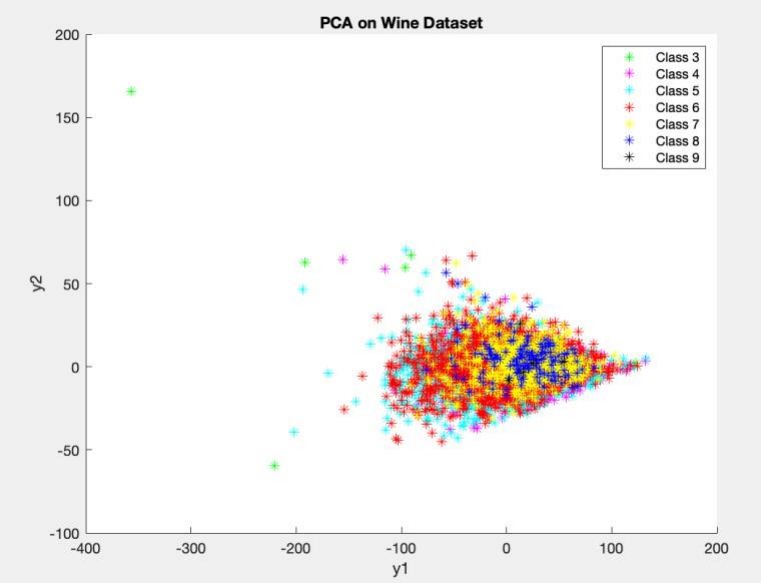
PCA on Wine Dataset
Class 3
Class 4
Class 5
Class 6
Class 7
Class 8
Class 9
y1
y2
200
150
100
50
0
-50
-100
-400
-300
-200
-100
0
100
200

Figure 12

Part B

The probability of error for this dataset is 1.73%. The count error is 178. The confusion matrices result is shown as table 8. As we can see, every value belongs to class 2 and 6 was classified correctly. All 6 classes have a good class distribution with little overlaps.

	1	2	3	4	5	6
1	0.9983	0	0	0	0	0
2	0.0012	1	0.0206	0	0	0
3	5.8072e-04	0	0.9794	0	0	0
4	0	0	0	0.9235	0.0053	0
5	0	0	0	0.0765	0.9947	0
6	0	0	0	0	0	1

Table 8: confusion matrix for human activity recognition dataset

The figure 13 shows the PCA results on human activity dataset. Compared with the PCA results of wine quality, human activity dataset has a clearer distribution. The average distance between means is 1.20 which is much higher than the average standard deviation 0.2. It explains the little overlap between each classes.

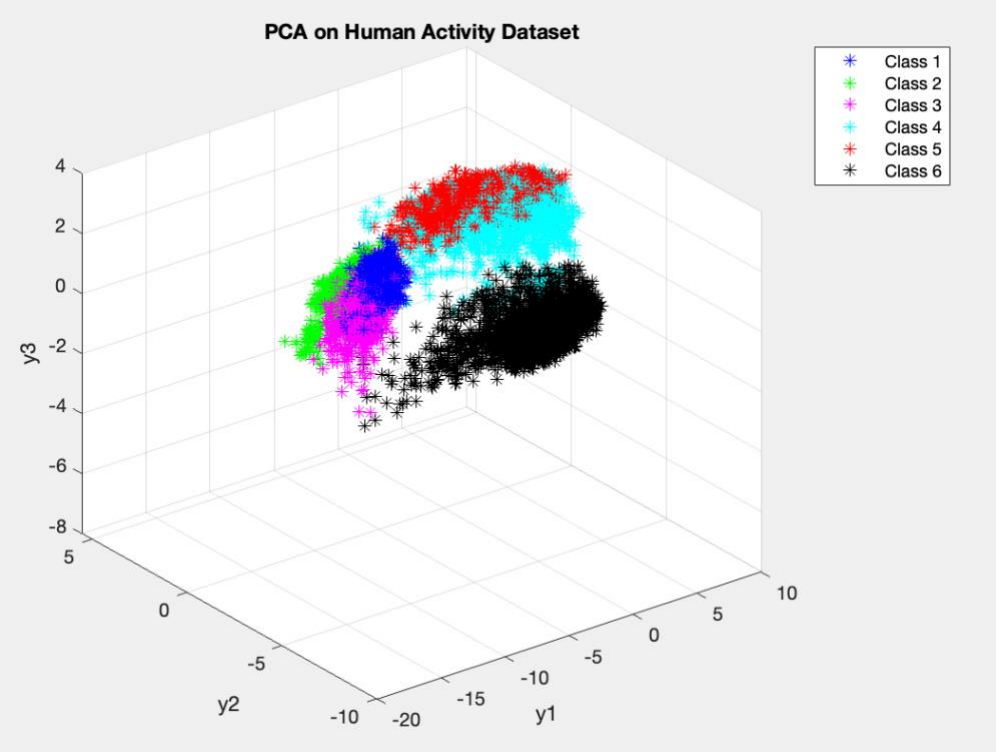

Figure 13

Appendix A

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%EECE5644 Spring 2022
%Homework #1
%Problem #1
%Significant parts of this code were derived from the following sources
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all;



%Initialize Parameters and Generate Data
N = 10000;  %Number of data points
n = 4;       %Dimensions of data
p0 = 0.7;  %Prior for label 0
p1 = 0.3;  %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
%mu: mean vector
%singma: covariance matrix
mu0 = [-1;-1;-1;-1];
Sigma0 = [2,-0.5,0.3,0;
      -0.5,1,-0.5,0;
      0.3,-0.5,1,0;
      0,0,0,2];
mu1 = [1;1;1;1];
Sigma1 = [1,0.3,-0.2,0;
```

```
        0.3,2,0.3,0;
        -0.2,0.3,1,0;
        0,0,0,3];


%Generate data as prescribed in assignment description
r0 = mvnrnd(mu0, Sigma0, N0);
r1 = mvnrnd(mu1, Sigma1, N1);


%Combine data from each distribution into a single dataset
x=zeros(N,n);
x(label==0,:)=r0;
x(label==1,:)=r1;


%Part A: ERM Classification with True Knowledge
discScore=log(evalGaussian(x',mu1,Sigma1)./evalGaussian(x',mu0,Sigma0));
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: Naive Bayesian Classifier
Sigma_NB=eye(4);    %Assumed covariance
%Generate data to illustrate assumptions
r0_NB = mvnrnd(mu0, Sigma_NB, N0);
r1_NB = mvnrnd(mu1, Sigma_NB, N1);


%Evaluate for different gammas
discScore_NB=...
```

```
    log(evalGaussian(x' ,mu1,Sigma_NB)./evalGaussian(x' ,mu0,Sigma_NB));
logGamma_NB=[min(discScore_NB)-0.1 sort(discScore_NB)+0.1];
for ind=1:length(logGamma_NB)
    decision=discScore_NB>logGamma_NB(ind);
    Num_pos_NB(ind)=sum(decision);
    pFP_NB(ind)=sum(decision==1 & label==0)/N0;
    pTP_NB(ind)=sum(decision==1 & label==1)/N1;
    pFN_NB(ind)=sum(decision==0 & label==1)/N1;
    pTN_NB(ind)=sum(decision==0 & label==0)/N0;


    pFE_NB(ind)=(sum(decision==0 & label==1) + sum(decision==1 & label==0))/(N0+N1);
    pFE2_NB(ind)=pFP(ind)*p0+pFN(ind)*p1;
end


%Estimated Minimum Error
[min_pFE_NB, min_pFE_ind_NB]=min(pFE_NB);
minGAMMA_NB=exp(logGamma(min_pFE_ind_NB));
min_FP_NB=pFP_NB(min_pFE_ind_NB);
min_TP_NB=pTP_NB(min_pFE_ind_NB);


%Plot Results
figure;
plot(pFP_NB,pTP_NB,'DisplayName','ROC Curve','LineWidth',2); hold all;
plot(min_FP_NB,min_TP_NB,'o','DisplayName',...
    'Estimated Min. Error','LineWidth',2);
xlabel('Prob. False Positive');
ylabel('Prob. True Positive');
title('Mininimum Expected Risk ROC Curve (Naive Bayesian)');
legend 'show';
grid on; box on;
```

```
figure;
plot(logGamma_NB,pFE_NB,'DisplayName','Errors','LineWidth',2);
hold on;
plot(logGamma_NB(min_pFE_ind_NB),pFE_NB(min_pFE_ind_NB),'ro',...
    'DisplayName','Minimum Error','LineWidth',2);
xlabel('Gamma');
ylabel('Proportion of Errors');
title('Probability of Error vs. Gamma (Naive Bayesian Estimate)')
grid on;
legend 'show';
fprintf('Estimated for NB: Gamma=%1.2f, Error=%1.2f%%\n', minGAMMA_NB,100*min_pFE_NB);


%Part 3: 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,'--',...
```

```
   'DisplayName','Tau for Min. Error','LineWidth',2); grid on;
xlabel('y');
title('Fisher LDA Projection');
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 (Fisher LDA)');
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 (Fisher LDA)')


grid on;
legend 'show';
fprintf('Estimated for LDA: Tau=%1.2f, Error=%1.2f%%\n', minTAU_LDA,100*min_pFE_LDA);
```

```
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 B

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%
%EECE5644 Sprint 2022
%Homework #1
%Problem #2
%Significant parts of this code were derived from the following sources
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%

clear all;
close all;

N = 10000; %number of samples
n = 3; %number of dimensions
C = 3; %number of classes

p = [0.3, 0.3, 0.4]; %class priors

%class conditional distributions
sigma(:,:,1) = [rand 0 0
          0 rand 0
          0 0 rand];
sigma(:,:,2) = [rand 0 0
          0 rand 0
          0 0 rand];
sigma(:,:,3) = [rand 0 0
          0 rand 0
          0 0 rand];
sigma(:,:,4) = [rand 0 0
          0 rand 0
          0 0 rand];
```

```
averageStdDev = trace(sum(sqrt(sigma),3))/16; %offset means by 2 std devs

mu(:,1) = [0; 0; 0];
mu(:,2) = [2*averageStdDev; 0; 0];
mu(:,3) = [averageStdDev; averageStdDev*sqrt(3); 0];
mu(:,4) = [averageStdDev; averageStdDev*(sqrt(3)/3); averageStdDev*sqrt(8/3)];

% 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)];

% plot generated data
figure
scatter3(x(1, label==1),x(2, label==1),x(3, label==1),'bo')
hold on
scatter3(x(1, label==2),x(2, label==2),x(3, label==2),'r*')
scatter3(x(1, label==3),x(2, label==3),x(3, label==3),'gx')
title('True Data Distributions')
legend('Class 1','Class 2','Class 3')
xlabel('x1')
ylabel('x2')
zlabel('x3')
hold off

%% Part A - MAP Classifier with True Knowledge

% 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
```

```
%% higher loss

% Loss matrix, Lambda_10
lossMatrix10 = [0 1 10
                1 0 10
                1 1 0];
expectedRisks10 = lossMatrix10*plgivenx;
[~,decisions10] = min(expectedRisks10,[],1);

% Make confusion matrix and plot data
figure
for i = 1:C %each decision
   for j = 1:C %each class label
      confusionMatrix10(i,j) = sum(decisions10==i & label==j)/NumClass(j);
      if i == j
         scatter(i,j) = scatter3(x(1,decisions10==i & label==j),...
         x(2,decisions10==i & label==j), ...
         x(3,decisions10==i & label==j), ...
         'g',shapes(j),'DisplayName', ...
         ['Class ' num2str(j) ' Correct Classification']);
         hold on
      else
         scatter(i,j) = scatter3(x(1,decisions10==i & label==j),...
         x(2,decisions10==i & label==j), ...
         x(3,decisions10==i & label==j), ...
         'r',shapes(j),'DisplayName', ...
         ['Class ' num2str(j) ' Incorrect Classification']);
         hold on
      end
   end
end
title('Correct vs. Incorrect Classification (Lambda=10)')
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

% Loss matrix, Lambda_100
lossMatrix100 = [0 1 100
                 1 0 100
                 1 1 0];
expectedRisks100 = lossMatrix100*plgivenx;
[~,decisions100] = min(expectedRisks100,[],1);

% Make confusion matrix and plot data
figure
for i = 1:C %each decision
   for j = 1:C %each class label
      confusionMatrix100(i,j) = sum(decisions100==i & label==j)/NumClass(j);
      if i == j
         scatter(i,j) = scatter3(x(1,decisions100==i & label==j),...
         x(2,decisions100==i & label==j),...
         x(3,decisions100==i & label==j), ...
         'g',shapes(j),'DisplayName', ...
         ['Class ' num2str(j) ' Correct Classification']);
         hold on
      else
         scatter(i,j) = scatter3(x(1,decisions100==i & label==j),...
         x(2,decisions100==i & label==j),...
         x(3,decisions100==i & label==j), ...
         'r',shapes(j),'DisplayName', ...
         ['Class ' num2str(j) ' Incorrect Classification']);
         hold on
      end
   end
```

```
end
title('Correct vs. Incorrect Classification (Lambda=100)')
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
```

Appendix C

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
%EECE5644 Spring 2022
%Homework #1
%Problem #3A
%Significant parts of this code were derived from the following sources
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%

%% Wine Dataset Class Conditional/Prior Estimation
clear all;
close all;
%%
% import data from excel
x = readmatrix('winequality-white.csv', 'Range', 'A2:K4899')';
label = readmatrix('winequality-white.csv', 'Range', 'L2:L4899')';
%%

N = size(x,2); %number of samples
n = size(x,1); %number of dimensions
C = 11; %number of classes
alpha = 0.1; %for regularization
sigmaTotal = cov(x'); %for regularization
%%

% Estimate class conditionals (gaussian) and class priors
for i = 1:C
   % estimate class prior
   p(i) = sum(label==i-1)/N;
   % estimate mean
   mu(:,i) = mean(x(:,label==i-1),2);
   % estimate covariance matrix
```

```
    sigma(:,:,i) = cov(x(:,label==i-1)');
    % regularize covariance matrix
    sigma(:,:,i) = sigma(:,:,i) + eye(size(sigma,1))*alpha*trace(sigmaTotal)/rank(sigmaTotal);
end

%% Wine Dataset ERM
% Evaluate class conditional pdfs
for i = 1:C
    if sum(isnan(sigma(:,:,i)))==0
        pxgivenl(i,:) = mvnpdf(x', mu(:,i)', sigma(:,:,i))';
    else
        pxgivenl(i,:) = zeros(1,4898); %zero for classes without data
    end
end

% 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(C,C)-eye(C);
expectedRisks = lossMatrix*plgivenx;
[~,decisions] = min(expectedRisks,[],1);
decisions = decisions-1; %because classes start at 0

% Total error probability estimate
countError = sum(label~=decisions);
pE = countError/N;

% Make confusion matrix
for i = 1:C %each decision
    for j = 1:C %each class label
        if sum(isnan(sigma(:,:,j)))==0
```

```
            confusionMatrix(i,j) = sum(decisions==i-1 & label==j-1)/sum(label==j-1);
        else
            confusionMatrix(i,j) = 0;
        end
    end
end

%% Wine Dataset PCA and Plotting
% Without classes, sample based estimates of distribution (gaussian)
muHat = mean(x,2);
% sigmaTotal is calculated above
% Make data zero-mean
xzm = x - muHat*ones(1,N);

% Get and sort eignvalues/eigenvectors
[Q,D] = eig(sigmaTotal);
[d,ind] = sort(diag(D),'descend');
Q = Q(:,ind);
D = diag(d);

% Calculate the first two principal components for visualization
y = Q(:,1:2)'*xzm;

% Percent of variance maintained
percentVar = trace(D(1:2,1:2))/trace(D);

% Plot data after PCA
figure
hold on
plot(y(1,label==3),y(2,label==3),'g*','DisplayName','Class 3')
plot(y(1,label==4),y(2,label==4),'m*','DisplayName','Class 4')
plot(y(1,label==5),y(2,label==5),'c*','DisplayName','Class 5')
plot(y(1,label==6),y(2,label==6),'r*','DisplayName','Class 6')
```

```
plot(y(1,label==7),y(2,label==7),'y*','DisplayName','Class 7')
plot(y(1,label==8),y(2,label==8),'b*','DisplayName','Class 8')
plot(y(1,label==9),y(2,label==9),'k*','DisplayName','Class 9')
title('PCA on Wine Dataset')
xlabel('y1')
ylabel('y2')
legend
hold off

% Average distance between means and average standard deviation
counterDist = 0;
counterSig = 0;
for i = 1:C
    if sum(isnan(sigma(:,:,i)))==0
        counterSig = counterSig+1;
        standardDev(i) = trace(sqrt(sigma(:,:,i)))/size(sigma,1);
    end
    for j = 1:C
        if sum(isnan(sigma(:,:,j)))==0
            if i < j
                counterDist = counterDist+1;
                distances(i) = sqrt(sum((x(:,i)-x(:,j)).^2));
            end
        end
    end
end
averageDistance = sum(distances)/counterDist;
averageStdDev = sum(standardDev)/counterSig;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
%EECE5644 Spring 2022
%Homework #1
```

```
%Problem #3B
%Significant parts of this code were derived from the following sources
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%
%% Human Activity Dataset Class Conditional/Prior Estimation
clear all;
close all;
%%

% import data from excel
x_test = readmatrix('X_test.txt', 'Range', 'B2:UP2948')';
x_train = readmatrix('X_train.txt', 'Range', 'B2:UP7353')';
label_test = readmatrix('y_test', 'Range', 'A2:A2948')';
label_train = readmatrix('y_train', 'Range', 'A2:A7353')';
x = horzcat(x_test,x_train);
label = horzcat(label_test,label_train);
N = size(x,2); %number of samples
n = size(x,1); %number of dimensions
C = 6; %number of classes
%%

alpha = 0.1; %for regularization
sigmaTotal = cov(x'); %for regularization

% Estimate class conditionals (gaussian) and class priors
for i = 1:C
  % estimate class prior
  p(i) = sum(label==i)/N;
  % estimate mean
  mu(:,i) = mean(x(:,label==i),2);
  % estimate covariance matrix
  sigma(:,:,i) = cov(x(:,label==i)');
  % regularize covariance matrix
```

```
    sigma(:,:,i) = sigma(:,:,i) + eye(size(sigma,1))*alpha*trace(sigmaTotal)/rank(sigmaTotal);
end

%% Human Activity Dataset ERM
% Evaluate class conditional pdfs
for i = 1:C
    if sum(isnan(sigma(:,:,i)))==0
        pxgivenl(i,:) = mvnpdf(x', mu(:,i)', sigma(:,:,i))';
    else
        pxgivenl(i,:) = zeros(1,4898); %zero for classes without data
    end
end

% 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(C,C)-eye(C);
expectedRisks = lossMatrix*plgivenx;
[~,decisions] = min(expectedRisks,[],1);

% Total error probability estimate
countError = sum(label~=decisions);
pE = countError/N;

% Make confusion matrix
for i = 1:C %each decision
    for j = 1:C %each class label
        if sum(isnan(sigma(:,:,j)))==0
            confusionMatrix(i,j) = sum(decisions==i & label==j)/sum(label==j);
        else
            confusionMatrix(i,j) = 0;
        end
```

```
    end
end

%% Human Activity Dataset PCA and Plotting
% Without classes, sample based estimates of distribution (gaussian)
muHat = mean(x,2);
% sigmaTotal is calculated above

% Make data zero-mean
xzm = x - muHat*ones(1,N);

% Get and sort eignvalues/eigenvectors
[Q,D] = eig(sigmaTotal);
[d,ind] = sort(diag(D),'descend');
Q = Q(:,ind);
D = diag(d);

% Calculate the first three principal components for visualization
y = Q(:,1:3)'*xzm;

% Percent of variance maintained
percentVar = trace(D(1:3,1:3))/trace(D);

% Plot data after PCA
figure
scatter3(y(1,label==1),y(2,label==1),y(3,label==1),'b*','DisplayName','Class 1')
hold on
scatter3(y(1,label==2),y(2,label==2),y(3,label==2),'g*','DisplayName','Class 2')
scatter3(y(1,label==3),y(2,label==3),y(3,label==3),'m*','DisplayName','Class 3')
scatter3(y(1,label==4),y(2,label==4),y(3,label==4),'c*','DisplayName','Class 4')
scatter3(y(1,label==5),y(2,label==5),y(3,label==5),'r*','DisplayName','Class 5')
scatter3(y(1,label==6),y(2,label==6),y(3,label==6),'k*','DisplayName','Class 6')
title('PCA on Human Activity Dataset')
```

```
xlabel('y1')
ylabel('y2')
zlabel('y3')
legend
hold off
% Average distance between means and average standard deviation
counterDist = 0;
counterSig = 0;
for i = 1:C
    if sum(isnan(sigma(:,:,i)))==0
        counterSig = counterSig+1;
        standardDev(i) = trace(sqrt(sigma(:,:,i)))/size(sigma,1);
    end
    for j = 1:C
        if sum(isnan(sigma(:,:,j)))==0
            if i < j
                counterDist = counterDist+1;
                distances(i) = sqrt(sum((x(:,i)-x(:,j)).^2));
            end
        end
    end
end
averageDistance = sum(distances)/counterDist;
averageStdDev = sum(standardDev)/counterSig;
```