Santosh Yadav Jada

16245773

Software used: MATLAB 2017

LDA:

```
clear all
clc
No_of_classes = 40;
Images_per_class = 10;
Training_images = 5;

%creation of arrays
A = [];
B = [];

for i = 1:No_of_classes
    fn = strcat('\\kc.umkc.edu\kc-users\home\s\sjzkt\My
Documents\MATLAB\att_faces\s',  num2str(i), '\');

    for j = 1:Training_images
        f = strcat(num2str(j) , '.pgm');
        filename = strcat(fn,f);
        img = imread(filename);

        A = img;
        A = (A(:));
        B = [B A];
    end
end
data=B;
[row,col]=size(data);
data=double(data)
%% calculating mean for train_data
m=mean(data,2);

%% Calculating Mean of Each Class%%
j=1;
for i=0:5:195
    n_m(:,j)=mean(data(:,i+1:i+5),2);
    mean_class(:,i+1:i+5)=repmat(n_m(:,j),[1,5]);
    j=j+1;
end;

%% Calculate the within class scatter (SW)
     tmp=zeros(10304,10304);
     wsca=zeros(10304,10304);
 for i =0:5:195
     tmp=(data(:,i+1:i+5)-mean_class(:,i+1:i+5))*((data(:,i+1:i+5)-
mean_class(:,i+1:i+5))');
```

```
    wsca=tmp+wsca;
end;

v=pinv(wsca);

 %%Calculating between scatter matrix%%
 temp=zeros(10304,10304);
 bsca=zeros(10304,10304);
for i=1:40
    temp=(tmp(:,i)-m)*((tmp(:,i)-m)');
    bsca=temp+bsca;

 end;


N = [];
O = [];
for i = 1:40
    fn = strcat('\\kc.umkc.edu\kc-users\home\s\sjzkt\My
Documents\MATLAB\att_faces\s', num2str(i) , '\');

    for j = 6:10
        f = strcat(num2str(j), '.pgm');
        filename1 = strcat(fn,f);
        img = imread(filename1);

        N = img;
        N = (N(:));
        O = [O N];
    end
end
disp(size(O));
O=double(O)

%% removing mean from train_data
m_data=data-repmat(m,1,200); %for the training set

%% Removing mean from test data
O=O-repmat(mean(O,2),1,200);% performing the mean of the test matrix and
subtracting the mean from each image(centering the data)

%% determining eiven values and eigen vectors
[evec,eval]=eig(v*bsca);

%% Sort the eigen vectors according to the eigen values
eigvalue = diag(eval);
[junk, index] = sort(eigvalue,'descend');

%% eigen values greater than 'o'
cnt=0;
for i=1:size(eigvalue,1)
    if(eigvalue(i)>0)
        cnt=cnt+1;
    end
```

```
end

%% And also we can use the eigen vectors that the corresponding eigen values
is greater than zero(Threshold) and this method will decrease the
% computation time and complexity
e_vec=evec(:,index(1:40)); %Number of principal components used

%% train data projection on new dimensions
data_pro=e_vec'*m_data;

%% Test data projection on new dimensions.
test_pro=e_vec'*O; %test projection
train_pro=abs(data_pro);
test_pro=abs(test_pro);

%% determining eucludian distance for train data
D=pdist2(train_pro',train_pro','Euclidean');

labels=zeros(200,200);
for i=1:200
 for j=1:200
 if((fix((i-1)/5)==fix((j-1)/5)))
         labels(i,j)=0;
 else
         labels(i,j)=1;
 end
 end
end
% using ezroc3 function
ezroc3(D,labels,2,'',1);
function
[roc,EER,area,EERthr,ALLthr,d,gen,imp]=ezroc3(H,T,plot_stat,headding,printInf
o)%,rbst
t1=min(min(min(H)));
t2=max(max(max(H)));
num_subj=size(H,1);

stp=(t2-t1)/500;   %step size here is 0.2% of threshold span, can be adjusted

if stp==0   %if all inputs are the same...
    stp=0.01;   %Token value
end
ALLthr=(t1-stp):stp:(t2+stp);
if (nargin==1 || (nargin==3 &&  isempty(T))||(nargin==2 &&
isempty(T))||(nargin==4 &&  isempty(T))||(nargin==5 &&  isempty(T)))  %Using
only H, multi-class case, and maybe 3D or no plot
    GAR=zeros(503,size(H,3));  %initialize for accumulation in case of
multiple H (on 3rd dim of H)
    FAR=zeros(503,size(H,3));
    gen=[]; %genuine scores place holder (diagonal of H), for claculation of
d'
    imp=[]; %impostor scores place holder (non-diagonal elements of H), for
claculation of d'
    for setnum=1:size(H,3); %multiple H measurements (across 3rd dim, where
2D H's stack up)
```

```
        gen=[gen; diag(H(:,:,setnum))]; %digonal scores
        imp=[imp; H(find(not(eye(size(H,2)))))]; %off-diagonal scores, with off-diagonal indices being listed by find(not(eye(size(H,2))))
        for t=(t1-stp):stp:(t2+stp),    %Note that same threshold is used for all H's, and we increase the limits by a smidgeon to get a full curve
            ind=round((t-t1)/stp+2);   %current loop index, +2 to start from 1
            id=H(:,:,setnum)>t;

            True_Accept=trace(id);  %TP
            False_Reject=num_subj-True_Accept;  %FN
            % In the following, id-diag(diag(id)) simply zeros out the diagonal of id
            True_Reject=sum( sum( (id-diag(diag(id)))==0 ) )-size(id,1); %TN, number of off-diag zeros. We need to subtract out the number of diagonals, as 'id-diag(diag(id))' introduces those many extra zeros into the sum
            False_Accept=sum( sum( id-diag(diag(id)) ) ); %FP, number of off-diagonal ones


GAR(ind,setnum)=GAR(ind,setnum)+True_Accept/(True_Accept+False_Reject); %1-FRR, Denum: all the positives (correctly IDed+incorrectly IDed)

FAR(ind,setnum)=FAR(ind,setnum)+False_Accept/(True_Reject+False_Accept); %1-GRR, Denum: all the negatives (correctly IDed+incorrectly IDed)
        end
    end
    GAR=sum(GAR,2)/size(H,3);   %average across multiple H's
    FAR=sum(FAR,2)/size(H,3);
elseif (nargin==2 || nargin==3 || nargin == 4 || nargin == 5),   %Regular, 1-class-vs-rest ROC, and maybe 3D or no plot
    gen=H(find(T)); %genuine scores
    imp=H(find(not(T))); %impostor scores
    for t=(t1-stp):stp:(t2+stp),    %span the limits by a smidgeon to get a full curve
        ind=round((t-t1)/stp+2);   %current loop index, +2 to start from 1
        id=H>t;

        True_Accept=sum(and(id,T)); %TP
        False_Reject=sum(and(not(id),T));   %FN

        True_Reject=sum(and(not(id),not(T)));   %TN
        False_Accept=sum(and(id,not(T)));   %FP

        GAR2(ind)=True_Accept/(True_Accept+False_Reject); %1-FRR, Denum: all the positives (correctly IDed+incorrectly IDed)
        FAR2(ind)=False_Accept/(True_Reject+False_Accept); %1-GRR, Denum: all the negatives (correctly IDed+incorrectly IDed)

    end
    GAR=GAR2';
    FAR=FAR2';
end
roc=[GAR';FAR'];
FRR=1-GAR;
```

```
[e ind]=min(abs(FRR'-FAR'));     %This is Approx w/ error e. Fix by linear
inerpolation of neigborhood and intersecting w/ y=x
EER=(FRR(ind)+FAR(ind))/2;
area=abs(trapz(roc(2,:),roc(1,:)));
EERthr=t1+(ind-1)*stp;%EER threshold

d=abs(mean(gen)-mean(imp))/(sqrt(0.5*(var(gen)+var(imp))));    %Decidability
or d'

if (nargin==1 || nargin==2 || nargin==3 || nargin == 4 || nargin == 5)
    if plot_stat == 2
        if printInfo == 1
            figure, plot(roc(2,:),roc(1,:),'LineWidth',3),axis([-0.002 1 0
1.002]),title(['ROC Curve: ' headding '   EER=' num2str(EER) ',   Area='
num2str(area) ',   Decidability=' num2str(d)]),xlabel('FAR'),ylabel('GAR');
        elseif printInfo == 0
            figure, plot(roc(2,:),roc(1,:),'LineWidth',3),axis([-0.002 1 0
1.002]),title(['ROC Curve: ' headding ' ']),xlabel('FAR'),ylabel('GAR');
        end
    elseif plot_stat == 3
        if printInfo == 1
            figure, plot3(roc(2,:),roc(1,:),ALLthr,'LineWidth',3),axis([0 1 0
1 (t1-stp) (t2+stp)]),title(['3D ROC Curve: ' headding '   EER=' num2str(EER)
',   Area=' num2str(area)  ',   Decidability='
num2str(d)]),xlabel('FAR'),ylabel('GAR'),zlabel('Threshold'),grid on,axis
square;
        elseif printInfo == 0
            figure, plot3(roc(2,:),roc(1,:),ALLthr,'LineWidth',3),axis([0 1 0
1 (t1-stp) (t2+stp)]),title(['3D ROC Curve: ' headding '
']),xlabel('FAR'),ylabel('GAR'),zlabel('Threshold'),grid on,axis square;
        end
    else
        %else it must be 0, i.e. no plot
    end
end
end
```

PCALDA:

```
clc
clear all
close all

% no of classes/subjects
class= 40;
% no of images in class
Total_images_in_class = 10;
% no of training images
Trainingimages = 5;
% declaration of arrays
A = [];
C = [];
```

```
% for loop to retrieve data from the ATT_face
for i = 1:class
    fn = strcat('\\kc.umkc.edu\kc-users\home\s\sjzkt\My
Documents\MATLAB\att_faces\s',  num2str(i), '\');
    B = [];
    for j = 1:5
        f = strcat(num2str(j) , '.pgm');
        file = strcat(fn,f);
        A = imread(file);
        A = (A(:));
        %horzcat for concatenation of the arrays
        B = horzcat(B,A);
    end
    C = horzcat(C,B);
end

%% PCA
data=double(C);
[row,col]=size(data);
% mean of the data
m=mean(data')';
d=data-(repmat(m,1,col));
d=double(d);
% covariance
co=d*d';
% Calculating eigen vectors and eigen values
[eigvector,eigvl] = eig(co);
 eigvalue = diag(eigvl);
% sorting of eigen values in descending order
[junk, index] = sort(eigvalue,'descend');
eigvalue = eigvalue(index);
eigvector = eigvector(:, index);
% Compute the number of eigen values that greater than zero (you can select
any threshold)
count1=0;
for i=1:size(eigvalue,1)
    if(eigvalue(i)>0)
        count1=count1+1;
    end
end
% We can use all the eigen vectors but this method will increase the
% computation time and complixity
vec=eigvector(:,1:count1);
% Compute the feature matrix (the space that will use it to project the
testing image on it)
x=vec'*d;

%% LDA
data=C;
[row,col]=size(data);
% Compute the mean of the data matrix "The mean of each row"
m=mean(data);

D=double(d)
C=double(C)
D= C-repmat(m,size(C,1),1);
```

```
C=double(C);

%% within class scatter
SW=D'*D;
INVSW=inv(SW);

%% between class scatter
N = size(C);
SB= 10304 * m'*m;
V=INVSW*SB;

%% Sorting the eigen vectors according to eigen values
[evec,eval]=eig(V);
eigvalue = diag(eval);
[junk, index] = sort(eigvalue,'descend');
eigvalue = eigvalue(index);
evec = evec(:, index);

%% eigen values greater than 0

count1=0;
for i=1:size(eigvalue,1)
    if(eigvalue(i)>0)
        count1=count1+1;
    end
end

vec=evec(:,1);

x=vec'*D';
%% If you have test data do the following
R = [];
T = [];
for i = 1:40
    fn = strcat('\\kc.umkc.edu\kc-users\home\s\sjzkt\My
Documents\MATLAB\att_faces\s', num2str(i) , '\');
    S = [];
    for j = 6:10
        f = strcat(num2str(j), '.pgm');
        filename1 = strcat(fn,f);
        img = imread(filename1);

        R = img;
        R = (R(:));
        S = [S R];
    end
    T = [T S];
end
disp(size(T));
T=double(T);
T=T-repmat(m,size(T,1),1);
%Project the testing data on the space of the training data
pro = vec'*T';

%To know what is the class of this test data use any classifier
```

```
D = pdist2(pro',x','euclidean');
labels=zeros(200,200);
for i=1:200
 for j=1:200
 if((fix((i-1)/5)==fix((j-1)/5)))
         labels(i,j)=0;
 else
         labels(i,j)=1;
 end
 end
end
% using ezroc3 function
ezroc3(D,labels,2,'',1);
function
[roc,EER,area,EERthr,ALLthr,d,gen,imp]=ezroc3(H,T,plot_stat,headding,printInf
o)%,rbst
t1=min(min(min(H)));
t2=max(max(max(H)));
num_subj=size(H,1);

stp=(t2-t1)/500;   %step size here is 0.2% of threshold span, can be adjusted

if stp==0   %if all inputs are the same...
    stp=0.01;   %Token value
end
ALLthr=(t1-stp):stp:(t2+stp);
if (nargin==1 || (nargin==3 &&  isempty(T))||(nargin==2 &&
isempty(T))||(nargin==4 &&  isempty(T))||(nargin==5 &&  isempty(T)))  %Using
only H, multi-class case, and maybe 3D or no plot
    GAR=zeros(503,size(H,3));  %initialize for accumulation in case of
multiple H (on 3rd dim of H)
    FAR=zeros(503,size(H,3));
    gen=[]; %genuine scores place holder (diagonal of H), for claculation of
d'
    imp=[]; %impostor scores place holder (non-diagonal elements of H), for
claculation of d'
    for setnum=1:size(H,3); %multiple H measurements (across 3rd dim, where
2D H's stack up)
        gen=[gen; diag(H(:,:,setnum))]; %digonal scores
        imp=[imp; H(find(not(eye(size(H,2)))))]; %off-diagonal scores, with
off-diagonal indices being listed by find(not(eye(size(H,2))))
        for t=(t1-stp):stp:(t2+stp),    %Note that same threshold is used for
all H's, and we increase the limits by a smidgeon to get a full curve
            ind=round((t-t1)/stp+2);   %current loop index, +2 to start from
1
            id=H(:,:,setnum)>t;

            True_Accept=trace(id);  %TP
            False_Reject=num_subj-True_Accept;  %FN
            % In the following, id-diag(diag(id)) simply zeros out the
diagonal of id
            True_Reject=sum( sum( (id-diag(diag(id)))==0 ) )-size(id,1); %TN,
number of off-diag zeros. We need to subtract out the number of diagonals, as
'id-diag(diag(id))' introduces those many extra zeros into the sum
            False_Accept=sum( sum( id-diag(diag(id)) ) ); %FP, number of off-
diagonal ones
```

```
GAR(ind,setnum)=GAR(ind,setnum)+True_Accept/(True_Accept+False_Reject); %1-
FRR, Denum: all the positives (correctly IDed+incorrectly IDed)

FAR(ind,setnum)=FAR(ind,setnum)+False_Accept/(True_Reject+False_Accept); %1-
GRR, Denum: all the negatives (correctly IDed+incorrectly IDed)
        end
    end
    GAR=sum(GAR,2)/size(H,3);   %average across multiple H's
    FAR=sum(FAR,2)/size(H,3);
elseif (nargin==2 || nargin==3 || nargin == 4 || nargin == 5),   %Regular, 1-
class-vs-rest ROC, and maybe 3D or no plot
    gen=H(find(T)); %genuine scores
    imp=H(find(not(T))); %impostor scores
    for t=(t1-stp):stp:(t2+stp),    %span the limits by a smidgeon to get a
full curve
        ind=round((t-t1)/stp+2);   %current loop index, +2 to start from 1
        id=H>t;

        True_Accept=sum(and(id,T)); %TP
        False_Reject=sum(and(not(id),T));   %FN

        True_Reject=sum(and(not(id),not(T)));   %TN
        False_Accept=sum(and(id,not(T)));   %FP

        GAR2(ind)=True_Accept/(True_Accept+False_Reject); %1-FRR, Denum: all
the positives (correctly IDed+incorrectly IDed)
        FAR2(ind)=False_Accept/(True_Reject+False_Accept); %1-GRR, Denum: all
the negatives (correctly IDed+incorrectly IDed)

    end
    GAR=GAR2';
    FAR=FAR2';
end
roc=[GAR';FAR'];
FRR=1-GAR;
[e ind]=min(abs(FRR'-FAR'));    %This is Approx w/ error e. Fix by linear
inerpolation of neigborhood and intersecting w/ y=x
EER=(FRR(ind)+FAR(ind))/2;
area=abs(trapz(roc(2,:),roc(1,:)));
EERthr=t1+(ind-1)*stp;%EER threshold

d=abs(mean(gen)-mean(imp))/(sqrt(0.5*(var(gen)+var(imp))));   %Decidability
or d'

if (nargin==1 || nargin==2 || nargin==3 || nargin == 4 || nargin == 5)
    if plot_stat == 2
        if printInfo == 1
            figure, plot(roc(2,:),roc(1,:),'LineWidth',3),axis([-0.002 1 0
1.002]),title(['ROC Curve: ' headding '   EER=' num2str(EER) ',   Area='
num2str(area) ',   Decidability=' num2str(d)]),xlabel('FAR'),ylabel('GAR');
        elseif printInfo == 0
            figure, plot(roc(2,:),roc(1,:),'LineWidth',3),axis([-0.002 1 0
1.002]),title(['ROC Curve: ' headding ' ']),xlabel('FAR'),ylabel('GAR');
```

```
        end
    elseif plot_stat == 3
        if printInfo == 1
            figure, plot3(roc(2,:),roc(1,:),ALLthr,'LineWidth',3),axis([0 1 0
1 (t1-stp) (t2+stp)]),title(['3D ROC Curve: ' headding '   EER=' num2str(EER)
',   Area=' num2str(area)  ',   Decidability='
num2str(d)]),xlabel('FAR'),ylabel('GAR'),zlabel('Threshold'),grid on,axis
square;
        elseif printInfo == 0
            figure, plot3(roc(2,:),roc(1,:),ALLthr,'LineWidth',3),axis([0 1 0
1 (t1-stp) (t2+stp)]),title(['3D ROC Curve: ' headding '
']),xlabel('FAR'),ylabel('GAR'),zlabel('Threshold'),grid on,axis square;
        end
    else
        %else it must be 0, i.e. no plot
    end
end
end
```