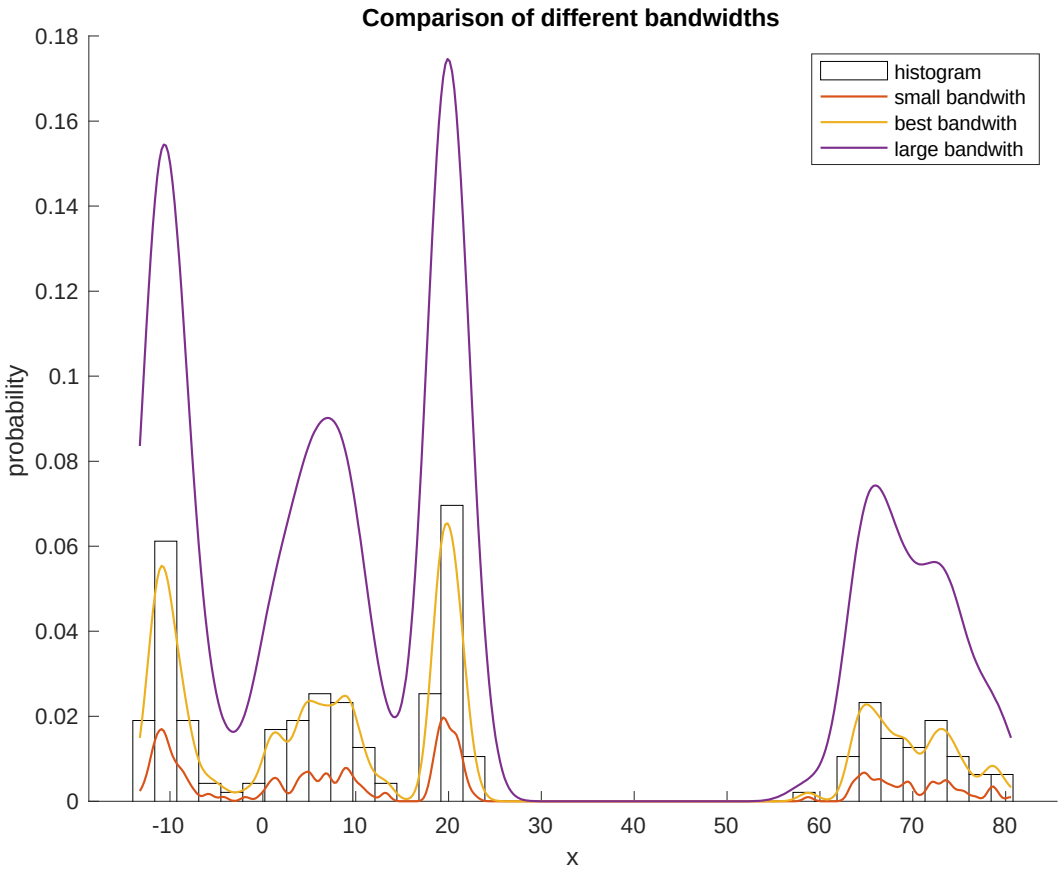
Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

1-)

(1)



The figure shows three KDE with different h values. The “best” fit has an h value of 1 and fits the data quite well, although we don’t know what is optimal yet until part (2) of this question. The large bandwidth is clearly oversmoothed and does not represent the data accurately, and neither does the small bandwidth. The code for this can be seen in the appendix

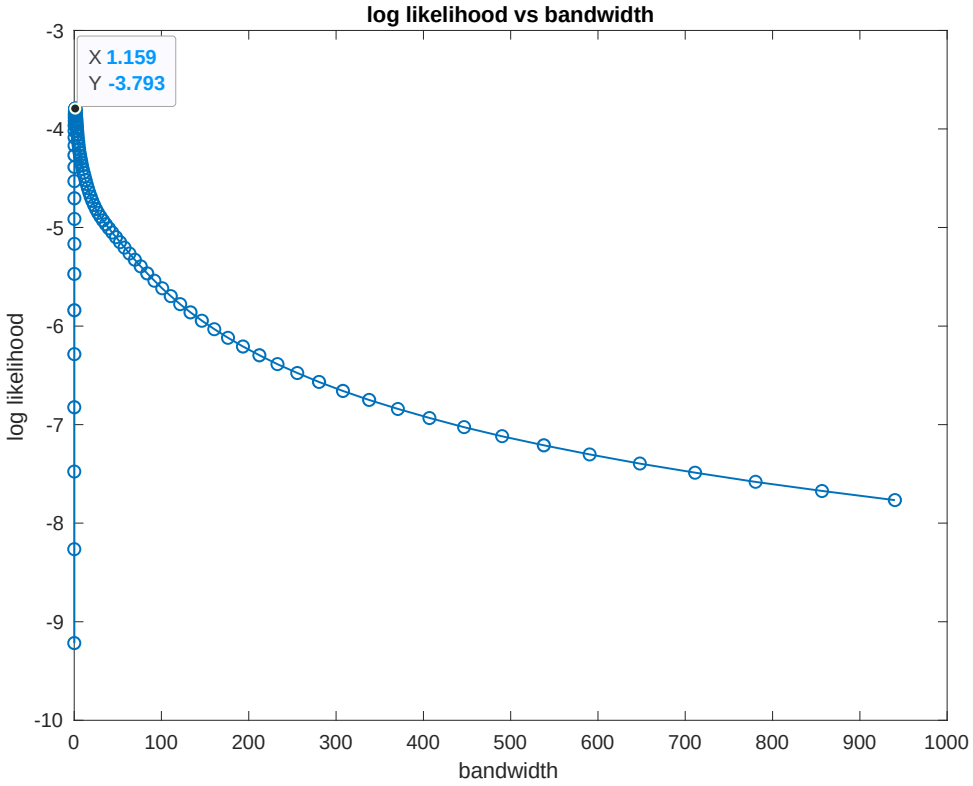
(2a&b)

The code in the appendix was run by taking the first data point out of the samples and using the remaining samples to build a KDE. The log likelihood of the validation was calculated at a value of 1.0153 using the formulas in lecture 7. The other points will be taken into consideration in part C of this problem.

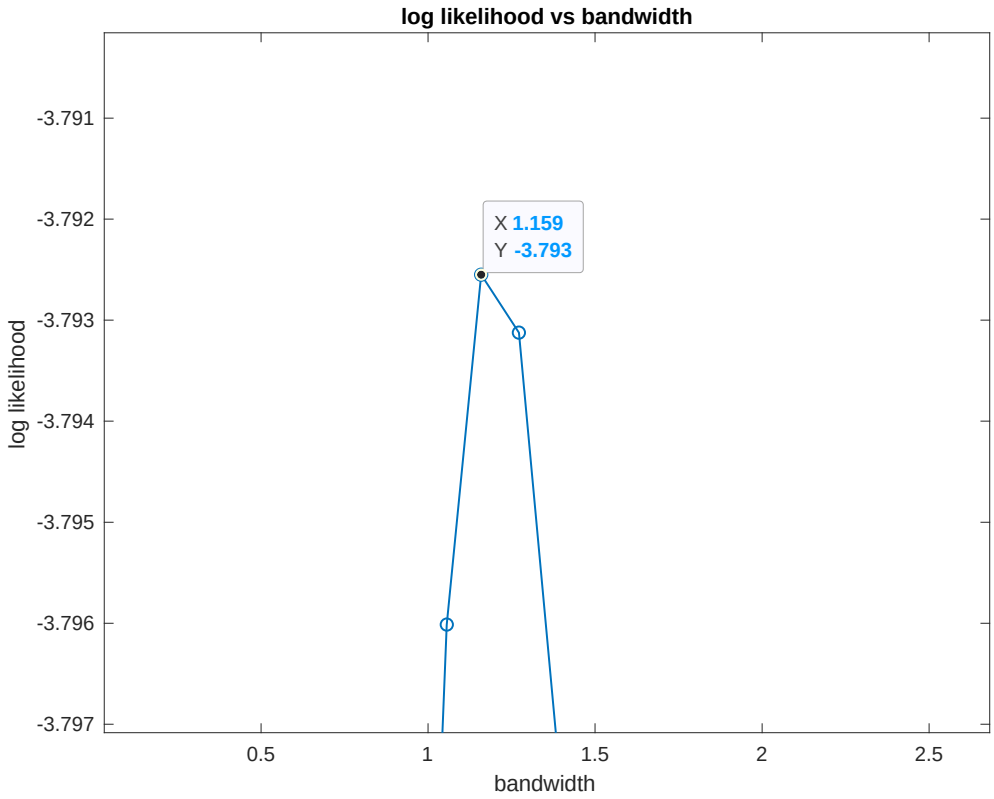
Assignment 2
Mahmood Mustafa Shilleh
02/24/2020
(2c)

Parts A&B were repeated for the whole data set and the average of the log was taken. The value turned out to be -3.800.

(2d-e)



Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

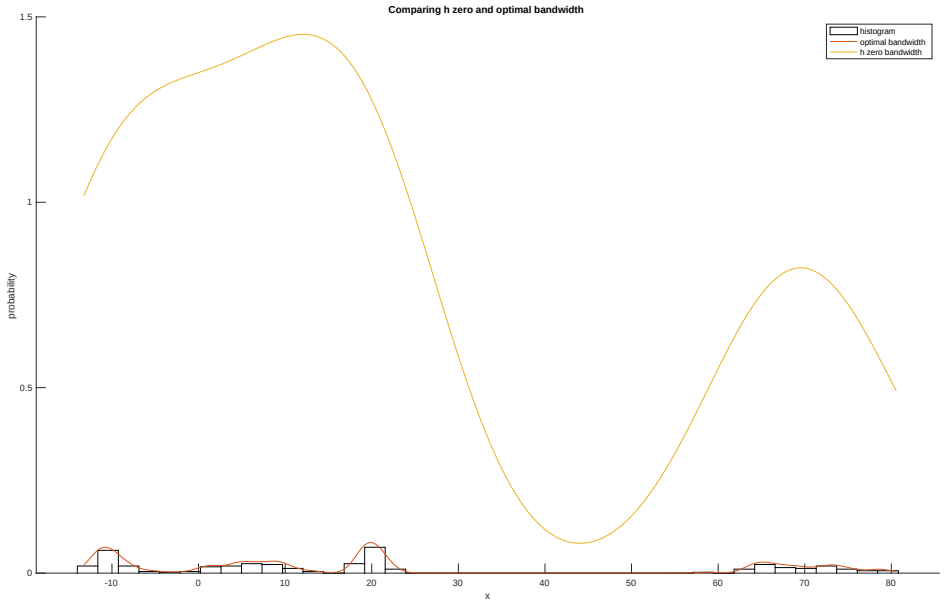


A bandwidth of 1.1591 generates the highest average log likelihood as seen in the figure above.

(2f)

I used the same code in part (1) of this problem, just changed the h vales, to generate the two KDE's on top of the histogram as shown in the figure below. The h0 value is 9.4014, which generates a very smooth curve as shown while the other bandwidth is much better fitting.

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020



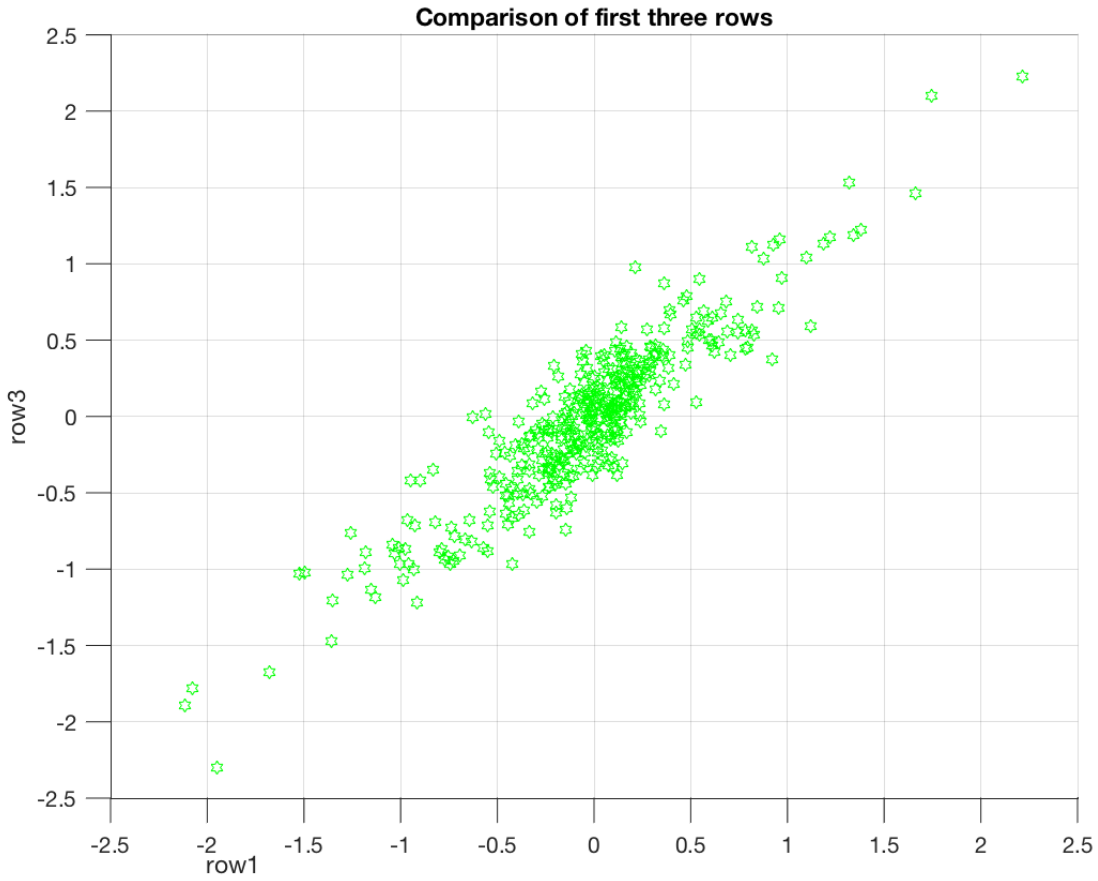
(2g)

After looking at the results we estimated the approximate optimal bandwidth by using Maximum likelihood cross validation, this proved significantly better than the mean square error method alone because that assumes that the data is gaussian, and we can see from the histogram alone that the data is far from being gaussian. Thus the method implemented in part d is more appropriate in this case

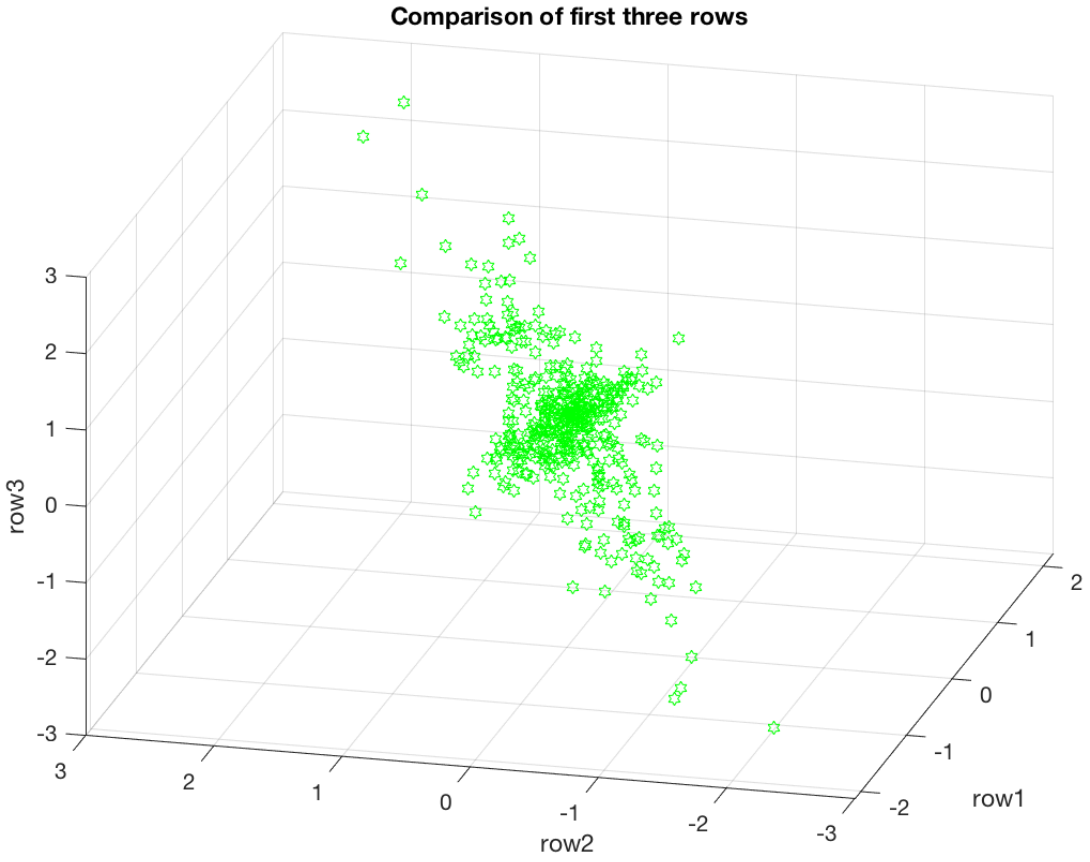
Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020
2-)

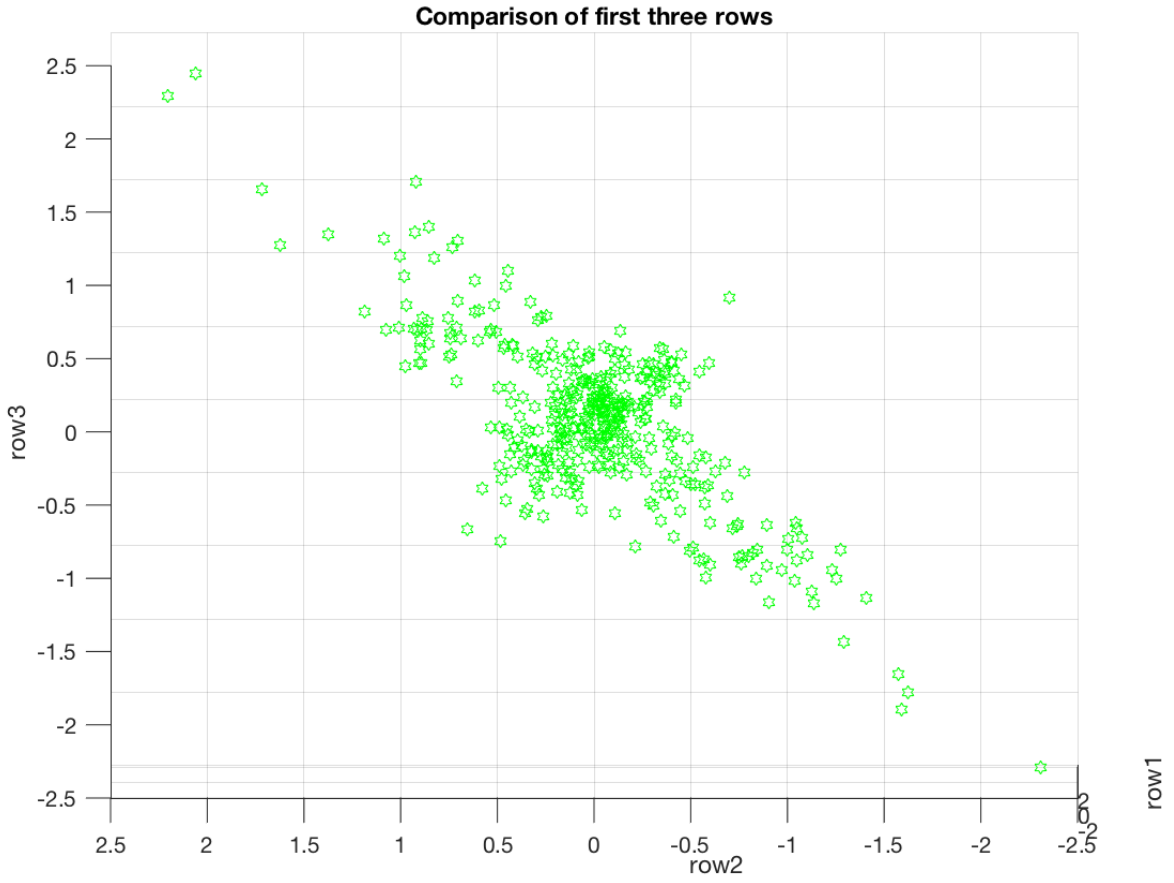
a-)



Assignment 2
Mahmood Mustafa Shilleh
02/24/2020



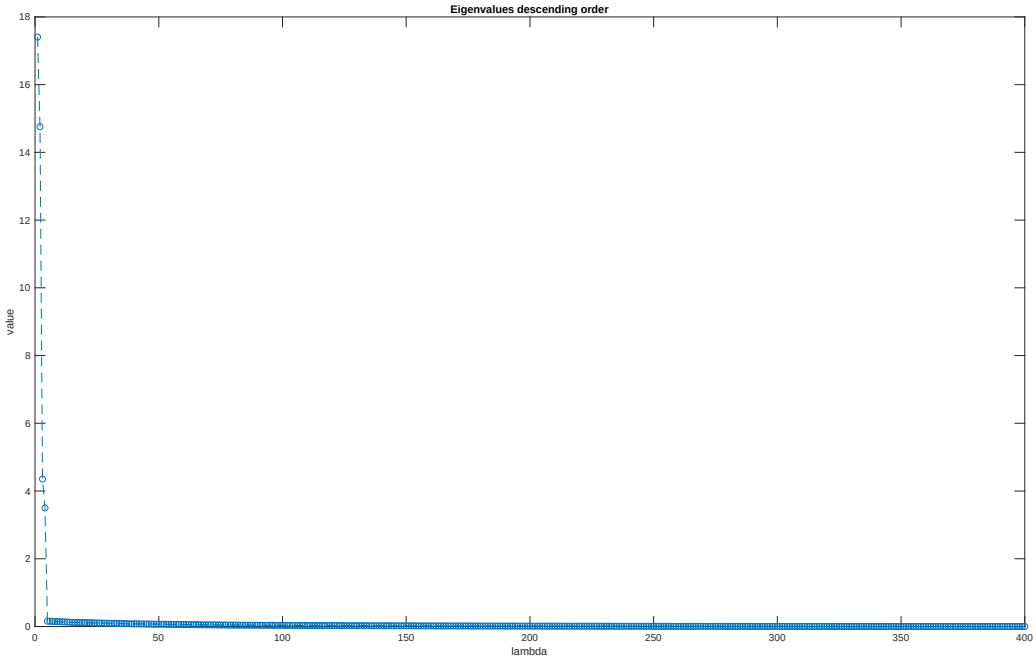
Assignment 2
Mahmood Mustafa Shilleh
02/24/2020



By analyzing the data in the three figures above it appears that the data has some structure in the sense that it is more dense in the center. It is also clear there is significantly more variance on certain axes which will be the principal axes.

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

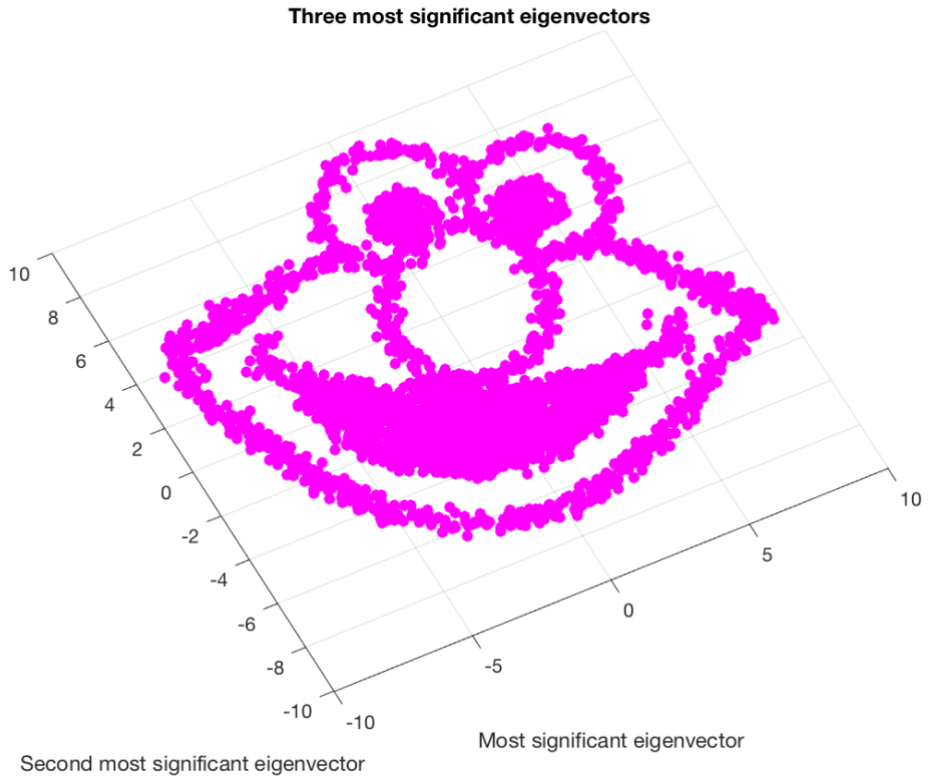
b-)



It looks to me that there are only about 4 eigen values that are responsible for most of the variance. This can be seen in the figure about which shows that the value of the first four eigenvalues are magnitudes higher than the others, thus capturing most of the variance!

c-)

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020



Yes the structure is clearly shown in the figure above as a figure of Elmo

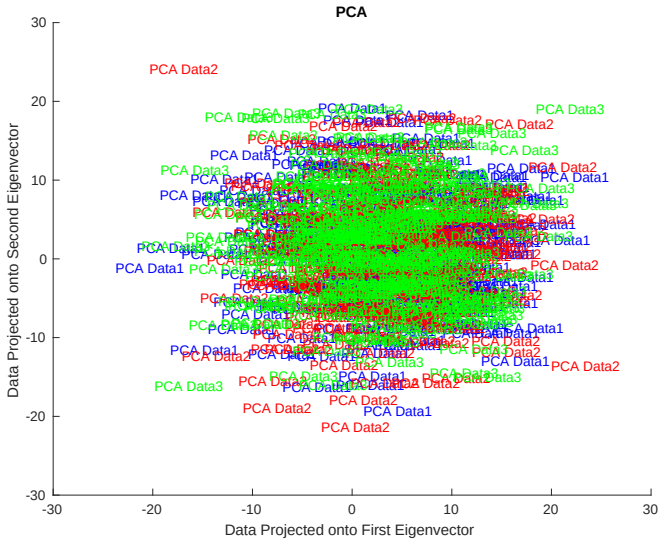
d-)
The findings in the problem indicate that PCA performs well in this case due to the separation we see in the Elmo figure. This is in agreement with part b, because only a small amount of eigenvectors contribute to a large chunk of the variance.

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

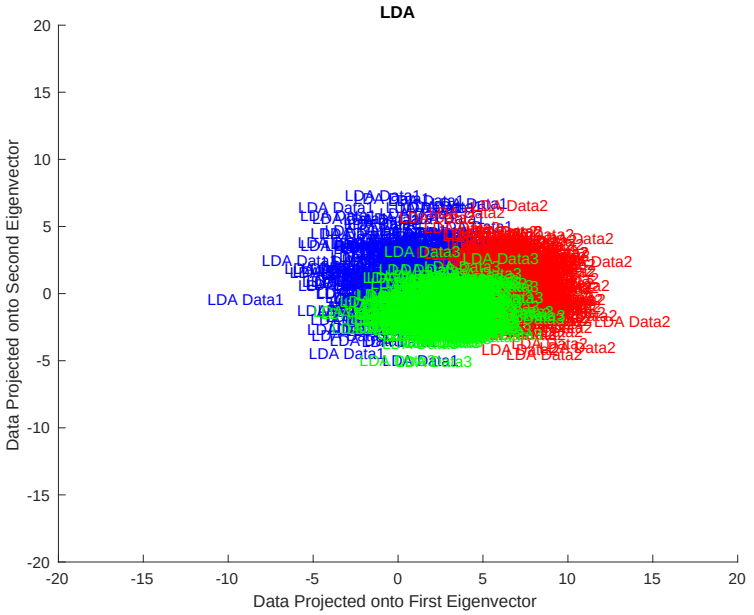
Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

3-)

a-)



b-)



Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

c-)
The results in the problem indicate that LDA performs much better in this case to separate the data. This is proven when analyzing the eigenvalues by using the code in the appendix. Unlike problem two where the first handful of eigenvalues take most of the variance, this is not true in this case.

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

4-)

a-c)

The data is split as instructed in part A (shown in the appendix). A classifier function is created. Running the quadratic for one trial the total performance of choosing the right class is 0.7937. However part C states to do multiple trials. In the code, there is a variable called “trials”. When this value is increased higher, lets say to 100, we get that the average classification rate is roughly 0.8250.

d-g)

I used the same script as in parts A-C, however I created a new function called “quadraticPCA”. After running it once I got a classification rate of 33.333 percent. After running it a 100 trials I got a rate of roughly 34.00 percent. This is not a good classifier in this case!

h-k)

I used the same script as in parts A-C, however I created a new function called “quadraticLDA”. After running it once I got a classification rate of 78.90 percent. After running it a 100 trials I got a rate of roughly 76.17 percent. This is not the best classifier out of the three!

l-)

Out of the three classifiers, classifying the data in its HD space proved to be the most accurate. While PCA proved to be the least accurate. This makes sense, due to the fact that the eigenvalues from the training data were all large and on the same order of magnitude, thus making PCA ineffective. LDA on the other hand proved to be much more effective but not as effective as the HD, this is due to the fact that in any dimensionality reduction problem we are removing information of the variance by simplifying the dimensions. LDA in this case was only projecting onto two eigenvectors, this does not capture all of the variance of the data but it is a good way to reduce computational costs when the dimensions get larger, hence the “curse of dimensionality”. But also, LDA does worse if information is not in the mean but in the variance.

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

5-)

a-c)

The KNN classifier for HD is created and shown in the appendix. It takes the input K as well so that the user can input a K that is less than or equal to the length of the training set. If we run the script for one trial and K=1, we get the performance of the KNN classifier to be as low as 0.29 in some cases. If we increase the trials to 100 while keeping the same K=1, the performance steadies out at 0.373. Now if we do one trial and increase K, to let's say 30, the performance gets much better at 0.4392, note that this is only for one trial! If we increase K to 100 for *one trial,* we get 0.4709. Let's try a number that is larger than the first training data set, that is, a number that is larger than 187 in this case (ex:400), then the performance goes down until 0.45. If we increase the number K to be the length of the training set, we get a performance of one, meaning the classifier is not working based on the logic. This is because it is being classified equally for each class, thus the algorithm is designed in this case to assume it belongs to class one. Thus, the last two sets of test data are classified completely incorrectly! The point is, we see an increase in the performance as K increases, but only up to a certain value, then it begins to decrease! A way to combat this problem would be to assign some sort of weight to the classifiers.

d-g)

After applying the KNN with PCA classifier, we observe that it is performing worse than the HD yet again, as was the case with the quadratic classifier. Performance is roughly 33 percent with one trial. If we increase the trials to 20 and increase K to 187, the rate increases slightly to 34.76 percent, not so good. Once again, it's because the eigenvalues capture the same amount of variance since they are all very close in value. Code is put in the appendix.

h-L)

After applying the LDA to the KNN and running it for one trial and K=1, I get a performance of 77 percent. Now if I increase both the trials to 20 and the K to 187 I get a performance of 75.74 percent, much better than the PCA.

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

6-)

a-)

After running the script in the appendix, I figured out that we need 46 eigenvectors to capture 90 percent of the variance in PCA, this is not good. In analyzing the LDA I see that the first eigenvalue is much larger than the second, which means that it's a better candidate for separating the data. We can only use two eigenvalues in LDA since we are only given two, because this is a three class problem (number of projections=C-1).

b&c-)

After going through a similar process as we did in questions 4 and 5. It turns out that the LDA performance in this case is better than both the HD and the PCA, and it also turns out that the KNN classifier produces the best performance with a maximum of 68.57 percent with a K value of 38 using LDA. The quadratic LDA produces a value of 65.27 percent, not as good as the KNN. KNN PCA produces a value at most 59 percent with a K value of 8 This is in agreement with the structure of the data, because as stated in part a, we need 46 eigenvalues to capture 90 percent of the variance! The HD performs the worst in this case with a value of 35.40 percent in the quadratic case and values beneath 40 percent for the KNN case.
That being said, I will go ahead and implement a KNN with LDA in part d!

d-)

Submitted in the code.

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

Appendix:

Problem 1 (1):

```
clear all
clc
close all
%Part(1)
[T1]=readvars('hw2p1_data.csv');
h=[0.5 1 2];
N=length(T1);
x=linspace(min(T1),max(T1),400)
pke=zeros(3,length(x))

for j=1:length(h)
for i=1:length(x)
    sum=0
for k=1:length(T1)
    kernelwindow=((x(i)-T1(k))/h(j));
    Kx=(2*pi)^(-1/2)*exp((-1/2)*kernelwindow*kernelwindow');
    sum=sum+Kx;
end
    pke(j,i)=(1/N*h(j))*sum
end
end

hold on
histogram(T1,40,'Normalization','pdf')
plot(x,pke(1,:),'LineWidth',1)
plot(x,pke(2,:),'LineWidth',1)
plot(x,pke(3,:),'LineWidth',1)
legend('histogram','small bandwith','best bandwith','large bandwith')
hold off
```

Problem 1 (2a&b):

```
clear all
clc
close all
%Part(2)a&b
[T1]=readvars('hw2p1_data.csv');
N=length(T1);
b = T1(T1~=T1(1))
h=1
summ=0
for i=1:length(b)
   kernelwindow=((T1(1)-b(i))/h);
   Kx=(2*pi)^(-1/2)*exp((-1/2)*kernelwindow*kernelwindow');
   summ=summ+Kx;
end
pkde=(1/(N-1)*h)*summ
logp_n=exp(pkde)
hstart=(1/N)*logp_n
```

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

Problem 1 (2c):

```
clear all
clc
close all
%Part(2c) steps a&b done for every data point
h=1
[T1]=readvars('hw2p1_data.csv');
N=length(T1);
r=iqr(T1)
sdev=std(T1)
A=min(r,sdev)
hzero=0.9*A*(N^(-1/5))
y1 = logspace(log(hzero/100)/log(10),log(hzero*100)/log(10),100)
for n=1:length(T1)
   b = T1(T1~=T1(n));
   summ=0;
for i=1:length(b)
   kernelwindow=((T1(n)-b(i))/h);
   Kx=(2*pi)^(-1/2)*exp((-1/2)*kernelwindow*kernelwindow');
   summ=summ+Kx;
end
   pkde(n)=(1/((N-1)*h))*summ;
end
logp_n=log(pkde);
hstart=(1/N)*sum(logp_n)
```

Problem 1 (2d-e):

```
clear all
clc
close all
%Part(2c) steps a&b done for every data point
h=1
[T1]=readvars('hw2p1_data.csv');
N=length(T1);
r=iqr(T1)/1.34;
sdev=std(T1);
A=min(r,sdev);
hzero=0.9*A*(N^(-1/5));
hlog = logspace(log(hzero/100)/log(10),log(hzero*100)/log(10),100);
for L=1:length(hlog)
for n=1:length(T1)
   b = T1(T1~=T1(n));
   summ=0;
for i=1:length(b)
   kernelwindow=((T1(n)-b(i))/hlog(L));
   Kx=(2*pi)^(-1/2)*exp((-1/2)*kernelwindow*kernelwindow');
   summ=summ+Kx;
end
   pkde(n)=(1/((N-1)*hlog(L)))*summ;
end
logp_n=log(pkde);
hstart(L)=(1/N)*sum(logp_n);
end
```

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

```
plot(hlog,hstart,'-o','LineWidth',1)
```

Problem 2 All Parts:

```
clear all
clc
%part(a)
a=readmatrix('hw2p2_data.csv');
figure(1)
scatter3(a(1,:),a(2,:),a(3,:),'g','h')
rotate3d
xlabel('row1')
ylabel('row2')
zlabel('row3')
title('Comparison of first three rows')
%Part b, I am centering the data at the origin and computing the
%eigenvalues and vectors through the covariance matrix
average=mean(a);
centereddata=a-average;
sig=cov(a);
sig2=cov(centereddata);
[vectors, values]=eig(sig2);
eigenvalues=diag(values)
%Sorted the eigenvalues from the line of code above from largest to
%smallest
[descendingorder, index]=sort(eigenvalues,'descend')
% I arranged the eigenvectors from largest to smallest, starting with the
% eigenvector that corresponds to the largest eigenvalue and etc.
[descendingordervectors]=vectors(index,:)

figure(2)
plot(descendingorder,'--o','LineWidth',1)
xlabel('lambda')
ylabel('value')
title('Eigenvalues descending order')

figure(3)
scatter3(descendingordervectors(:,400),descendingordervectors(:,399),descendi
ngordervectors(:,398),'fill','magenta')
xlabel('Most significant eigenvector')
ylabel('Second most significant eigenvector')
zlabel('Third most significant eigenvector')
title('Three most significant eigenvectors')
```

Problem 3 All Parts:

```
clc
clear all
close all
X=ones(48,1);
mu1=[5 0 5, X']
mu2=[4 6 7, X']
```

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

```
mu3=[6 2 4, X']
v=36*ones(51,1);
D1=diag(v)
D2=diag(v)
D3=diag(v)
sigma1=[5 -4 -2;-4 4 0;-2 0 5]
for k=1:3
for i=1:3
    D1(k,i)=sigma1(k,i)
end
end
sigma2=[3 0 0;0 3 0;0 0 3]
for k=1:3
for i=1:3
    D2(k,i)=sigma2(k,i)
end
end
sigma3=[6 5 6;5 6 7;6 7 9]
for k=1:3
for i=1:3
    D3(k,i)=sigma3(k,i)
end
end
data1=mvnrnd(mu1,D1,250)
data2=mvnrnd(mu2,D2,250)
data3=mvnrnd(mu3,D3,250)
alldata=[data1;data2;data3]
average=mean(alldata);
centereddata=alldata-average;
sig2=cov(centereddata);
[vectors, values]=eig(sig2);
eigenvalues=diag(values)
%Sorted the eigenvalues from the line of code above from largest to
%smallest
[descendingorder, index]=sort(eigenvalues,'descend')
% I arranged the eigenvectors from largest to smallest, starting with the
% eigenvector that corresponds to the largest eigenvalue and etc.
[descendingordervectors]=vectors(:,index)

figure(1)
hold on
text(data1*descendingordervectors(:,1),data1*descendingordervectors(:,2),'PCA
Data1','Color','blue')
axis([-30 30 -30 30])
xlabel('Data Projected onto First Eigenvector')
ylabel('Data Projected onto Second Eigenvector')
title('PCA')
text(data2*descendingordervectors(:,1),data2*descendingordervectors(:,2),'PCA
Data2','Color','red')
axis([-30 30 -30 30])
text(data3*descendingordervectors(:,1),data3*descendingordervectors(:,2),'PCA
Data3','Color','green')
axis([-30 30 -30 30])
hold off
```

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

```
class=[mu1;mu2;mu3]

[y, v, d] = tamu_lda(alldata, [ones(250,1);2*ones(250,1);3*ones(250,1)]);


figure(2)
hold on
text(data1*v(:,1),data1*v(:,2),'LDA Data1','Color','blue')
axis([-20 20 -20 20])
text(data2*v(:,1),data2*v(:,2),'LDA Data2','Color','red')
axis([-20 20 -20 20])
text(data3*v(:,1),data3*v(:,2),'LDA Data3','Color','green')
axis([-20 20 -20 20])
xlabel('Data Projected onto First Eigenvector')
ylabel('Data Projected onto Second Eigenvector')
title('LDA')
```

Problem 4 (a-c):

```
clc
clear all
close all
X=ones(48,1);
mu1=[5 0 5, X'];
mu2=[4 6 7, X'];
mu3=[6 2 4, X'];
v=36*ones(51,1);
D1=diag(v);
D2=diag(v);
D3=diag(v);
sigma1=[5 -4 -2;-4 4 0;-2 0 5];
for k=1:3
for i=1:3
    D1(k,i)=sigma1(k,i);
end
end
sigma2=[3 0 0;0 3 0;0 0 3];
for k=1:3
for i=1:3
    D2(k,i)=sigma2(k,i);
end
end
sigma3=[6 5 6;5 6 7;6 7 9];
for k=1:3
for i=1:3
    D3(k,i)=sigma3(k,i);
end
end

%I am repeating steps a&b for as many trials to get the average
%classification rate
trials=1
for I=1:trials
```

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

```
data1=mvnrnd(mu1,D1,250);
data2=mvnrnd(mu2,D2,250);
data3=mvnrnd(mu3,D3,250);
p = randperm(length(data1));
%Splitting Data into 75 percent for training and 25% for test data
%For data set 1
indicesoftestdata1= p(1,1:round(length(data1)*.25));
indicesoftrainingdata1= p(1,(round(length(data1)*.25))+1:end);
testdata1=data1(indicesoftestdata1,:);
trainingdata1=data1(indicesoftrainingdata1,:);
%For data set 2
indicesoftestdata2= p(1,1:round(length(data2)*.25));
indicesoftrainingdata2= p(1,(round(length(data2)*.25))+1:end);
testdata2=data2(indicesoftestdata2,:);
trainingdata2=data2(indicesoftrainingdata2,:);
%For data set 3
indicesoftestdata3= p(1,1:round(length(data3)*.25));
indicesoftrainingdata3= p(1,(round(length(data3)*.25))+1:end);
testdata3=data3(indicesoftestdata3,:);
trainingdata3=data3(indicesoftrainingdata3,:);
%stack the data in three dimensions to make it more organized by using the
cat command!
STACKEDtestdata = cat(3,testdata1,testdata2,testdata3);
STACKEDtrainingdata = cat(3,trainingdata1,trainingdata2,trainingdata3);

%Now I have all of the test and sample data in two arrays
[gx1,gx2,gx3,pf] = quadratic(STACKEDtestdata,STACKEDtrainingdata,1/3);

totalperformance(I)=pf;
end
totalperformance=sum(totalperformance)/length(totalperformance)

function [gx1,gx2,gx3,performance] = quadratic(x,trainingdata,P)

% x is the test sample, what you want to classify
% training data
% P is the prior

% The input here is a three dimensional array with thickness of 3, so were
% getting three mean vectors and 3 covariance matrices for the threee
% different classes of training data that we will use to classify the test
% data in this function!


mu=mean(trainingdata);
covarianceOFfirstTrainingData=cov(trainingdata(:,:,1));
covarianceOFsecondTrainingData=cov(trainingdata(:,:,2));
covarianceOFthirdTrainingData=cov(trainingdata(:,:,3));
SIZE=size(x);

classiferratetotal=zeros(1,SIZE(3));

for E=1:SIZE(3)
c1=0
```

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

```
c2=0
c3=0
for H=1:length(x)

%First classifier
gx1=-0.5*(x(H,:,E)-mu(:,:,1))*inv(covarianceOFfirstTrainingData)*(x(H,:,E)-
mu(:,:,1))'-(0.5)*log(det(covarianceOFfirstTrainingData))+log(P);
%Second classifier
gx2=-0.5*(x(H,:,E)-mu(:,:,2))*inv(covarianceOFsecondTrainingData)*(x(H,:,E)-
mu(:,:,2))'-(0.5)*log(det(covarianceOFsecondTrainingData))+log(P);
%Third Classifier
gx3=-0.5*(x(H,:,E)-mu(:,:,3))*inv(covarianceOFthirdTrainingData)*(x(H,:,E)-
mu(:,:,3))'-(0.5)*log(det(covarianceOFthirdTrainingData))+log(P);

if gx1>gx2 && gx1>gx3
    c1=c1+1;
elseif gx2>gx1 && gx2>gx3
    c2=c2+1;
elseif gx3>gx1 && gx3>gx2
     c3=c3+1;
end

end

if E==1
  classifierrate=c1/SIZE(1);
elseif E==2
  classifierrate=c2/SIZE(1);
else
    classifierrate=c3/SIZE(1);
end
classiferratetotal(E)=classifierrate;
end
performance=sum(classiferratetotal)/length(classiferratetotal);


end
```

Problem 4 (d-g):

```
function [gx1,gx2,gx3,performance] = quadraticPCA(x,trainingdata,P)

% x is the test sample, what you want to classify
% training data
% P is the prior

% The input here is a three dimensional array with thickness of 3, so were
% getting three mean vectors and 3 covariance matrices for the threee
% different classes of training data that we will use to classify the test
% data in this function!


mu=mean(trainingdata);
```

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

```
covarianceOFfirstTrainingData=cov(trainingdata(:,:,1));
covarianceOFsecondTrainingData=cov(trainingdata(:,:,2));
covarianceOFthirdTrainingData=cov(trainingdata(:,:,3));
SIZE=size(x);
classiferratetotal=zeros(1,SIZE(3));

%PCA STUFF (d-g)
trainingdata2dimensional=[trainingdata(:,:,1);trainingdata(:,:,2);trainingdat
a(:,:,3)]
mutraining=mean(trainingdata2dimensional)
covtraining=cov(trainingdata2dimensional)
[vectors, values]=eig(covtraining)
eigenvalues=diag(values)
[descendingorder, index]=sort(eigenvalues,'descend')
[descendingordervectors]=vectors(:,index)
%Projecting training Data
projtrainingdata1=[trainingdata(:,:,1)*descendingordervectors(:,1),trainingda
ta(:,:,1)*descendingordervectors(:,2),trainingdata(:,:,1)*descendingordervect
ors(:,3)]
projtrainingdata2=[trainingdata(:,:,2)*descendingordervectors(:,1),trainingda
ta(:,:,2)*descendingordervectors(:,2),trainingdata(:,:,2)*descendingordervect
ors(:,3)]
projtrainingdata3=[trainingdata(:,:,3)*descendingordervectors(:,1),trainingda
ta(:,:,3)*descendingordervectors(:,2),trainingdata(:,:,3)*descendingordervect
ors(:,3)]
meanprojection=cat(3,mean(projtrainingdata1),mean(projtrainingdata2),mean(pro
jtrainingdata3))
covprojection=cat(3,cov(projtrainingdata1),cov(projtrainingdata2),cov(projtra
iningdata3))
% Projecting test Data
pt1=[x(:,:,1)*descendingordervectors(:,1),x(:,:,1)*descendingordervectors(:,2
),x(:,:,1)*descendingordervectors(:,3)]
pt2=[x(:,:,2)*descendingordervectors(:,1),x(:,:,2)*descendingordervectors(:,2
),x(:,:,2)*descendingordervectors(:,3)]
pt3=[x(:,:,3)*descendingordervectors(:,1),x(:,:,3)*descendingordervectors(:,2
),x(:,:,3)*descendingordervectors(:,3)]
pt=cat(3,pt1,pt2,pt3)

%Recalculating the MEAN OF THE TRAINING DATA


for E=1:SIZE(3)
c1=0
c2=0
c3=0
for H=1:length(pt)

%First classifier
gx1=-0.5*(pt(H,:,E)-
meanprojection(:,:,1))*inv(covprojection(:,:,1))*(pt(H,:,E)-
meanprojection(:,:,1))'-(0.5)*log(det(covprojection(:,:,1)))+log(P);
%Second classifier
gx2=-0.5*(pt(H,:,E)-
meanprojection(:,:,2))*inv(covprojection(:,:,2))*(pt(H,:,E)-
meanprojection(:,:,2))'-(0.5)*log(det(covprojection(:,:,2)))+log(P);
```

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

```
%Third Classifier
gx3=-0.5*(pt(H,:,E)-
meanprojection(:,:,3))*inv(covprojection(:,:,3))*(pt(H,:,E)-
meanprojection(:,:,3))'-(0.5)*log(det(covprojection(:,:,3)))+log(P);

if gx1>gx2 && gx1>gx3
    c1=c1+1;
elseif gx2>gx1 && gx2>gx3
    c2=c2+1;
elseif gx3>gx1 && gx3>gx2
     c3=c3+1;
end

end

if E==1
  classifierrate=c1/SIZE(1);
elseif E==2
  classifierrate=c2/SIZE(1);
else
    classifierrate=c3/SIZE(1);
end
classiferratetotal(E)=classifierrate;
end
performance=sum(classiferratetotal)/length(classiferratetotal);


end
```

Problem 4 (h-L):

```
function [gx1,gx2,gx3,performance] = quadraticLDA(x,trainingdata,P)

% x is the test sample, what you want to classify
% training data
% P is the prior

% The input here is a three dimensional array with thickness of 3, so were
% getting three mean vectors and 3 covariance matrices for the threee
% different classes of training data that we will use to classify the test
% data in this function!


mu=mean(trainingdata);
covarianceOFfirstTrainingData=cov(trainingdata(:,:,1));
covarianceOFsecondTrainingData=cov(trainingdata(:,:,2));
covarianceOFthirdTrainingData=cov(trainingdata(:,:,3));
SIZE=size(x);
classiferratetotal=zeros(1,SIZE(3));
%LDA STUFF (h-L)
trainingdata2dimensional=[trainingdata(:,:,1);trainingdata(:,:,2);trainingdat
a(:,:,3)]
```

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

```
[y, v, d] = tamu_lda(trainingdata2dimensional,
[ones(length(trainingdata),1);2*ones(length(trainingdata),1);3*ones(length(tr
ainingdata),1)]);

%Projecting training Data
projtrainingdata1=[trainingdata(:,:,1)*v(:,1),trainingdata(:,:,1)*v(:,2)]
projtrainingdata2=[trainingdata(:,:,2)*v(:,1),trainingdata(:,:,2)*v(:,2)]
projtrainingdata3=[trainingdata(:,:,3)*v(:,1),trainingdata(:,:,3)*v(:,2)]
meanprojection=cat(3,mean(projtrainingdata1),mean(projtrainingdata2),mean(pro
jtrainingdata3))
covprojection=cat(3,cov(projtrainingdata1),cov(projtrainingdata2),cov(projtra
iningdata3))
% Projecting test Data
pt1=[x(:,:,1)*v(:,1),x(:,:,1)*v(:,2)]
pt2=[x(:,:,2)*v(:,1),x(:,:,2)*v(:,2)]
pt3=[x(:,:,3)*v(:,1),x(:,:,3)*v(:,2)]
pt=cat(3,pt1,pt2,pt3)

%Recalculating the MEAN OF THE TRAINING DATA


for E=1:SIZE(3)
c1=0
c2=0
c3=0
for H=1:length(pt)

%First classifier
gx1=-0.5*(pt(H,:,E)-
meanprojection(:,:,1))*inv(covprojection(:,:,1))*(pt(H,:,E)-
meanprojection(:,:,1))'-(0.5)*log(det(covprojection(:,:,1)))+log(P);
%Second classifier
gx2=-0.5*(pt(H,:,E)-
meanprojection(:,:,2))*inv(covprojection(:,:,2))*(pt(H,:,E)-
meanprojection(:,:,2))'-(0.5)*log(det(covprojection(:,:,2)))+log(P);
%Third Classifier
gx3=-0.5*(pt(H,:,E)-
meanprojection(:,:,3))*inv(covprojection(:,:,3))*(pt(H,:,E)-
meanprojection(:,:,3))'-(0.5)*log(det(covprojection(:,:,3)))+log(P);

if gx1>gx2 && gx1>gx3
    c1=c1+1;
elseif gx2>gx1 && gx2>gx3
    c2=c2+1;
elseif gx3>gx1 && gx3>gx2
     c3=c3+1;
end

end

if E==1
  classifierrate=c1/SIZE(1);
elseif E==2
  classifierrate=c2/SIZE(1);
```

```
else
    classifierrate=c3/SIZE(1);
end
classiferratetotal(E)=classifierrate;
end
performance=sum(classiferratetotal)/length(classiferratetotal);


end
```

Problem 5 (a-c):

```
function [performance] = KNNHD(x,trainingdata,P,K)

% x is the test sample, what you want to classify
% training data
% P is the prior
% K is the input for the KNN, how many points you want to include in the
% hypervolume

SIZE=size(x);

classiferratetotal=zeros(1,SIZE(3));

trainingdata2dimensional=[trainingdata(:,:,1);trainingdata(:,:,2);trainingdat
a(:,:,3)];

testdata2d=[x(:,:,1);x(:,:,2);x(:,:,3)];

class1=0
class2=0
class3=0
for H=1:length(testdata2d)
c1=0
c2=0
c3=0
for T=1:length(trainingdata2dimensional);
distance=norm(testdata2d(H,:)-trainingdata2dimensional(T,:));

d(T)= distance;


end
[B,I] = sort(d);

B=B(1,1:K);
I=I(1,1:K);
```

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

```
for NY=1:length(I);
    if I(NY)<=length(trainingdata2dimensional)/3
        c1=c1+1;
    elseif I(NY)<=length(trainingdata2dimensional)*2/3
        c2=c2+1;
    else
        c3=c3+1;
    end
end
%This if statement floors it to classify as class one if theyre all equal,
%and so on. I am not sure if this logic produces the best classifier, the
%only other way would be to assign weights instead of doing it randomly
%like this
if c1==c2 && c1==c3
    class1=class1+1
elseif c1==c2 && c1>c3
    class1=class1+1
elseif c2==c3 && c2>c1
    class2=class2+1
elseif c3==c1 && c3>c2
    class3=class3+1
elseif c1>c2 && c1>c3
     class1=class1+1
 elseif c2>c1 && c2>c3
     class2=class2+1
 elseif c3>c1 && c3>c2
     class3=class3+1
end

if H<=length(testdata2d)/3
    counter1=class1;
    class2=0;
    class3=0;
elseif H<=length(testdata2d)*2/3
    counter2=class2;
    class3=0;
else
    counter3=class3;
end

end
classifierrate=counter1/SIZE(1)+counter2/SIZE(1)+counter3/SIZE(1);
performance=classifierrate/3;

end
```

Problem 5 (d-g):

```
function [performance] = KNNPCA(x,trainingdata,P,K)

% x is the test sample, what you want to classify
% training data
% P is the prior
```

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

```
% K is the input for the KNN, how many points you want to include in the
% hypervolume

SIZE=size(x);

classiferratetotal=zeros(1,SIZE(3));

trainingdata2dimensional=[trainingdata(:,:,1);trainingdata(:,:,2);trainingdat
a(:,:,3)];

testdata2d=[x(:,:,1);x(:,:,2);x(:,:,3)];

%PCA STUFF (d-g)
mutraining=mean(trainingdata2dimensional)
covtraining=cov(trainingdata2dimensional)
[vectors, values]=eig(covtraining)
eigenvalues=diag(values)
[descendingorder, index]=sort(eigenvalues,'descend')
[descendingordervectors]=vectors(:,index)
%Projecting training Data
projtrainingdata1=[trainingdata(:,:,1)*descendingordervectors(:,1),trainingda
ta(:,:,1)*descendingordervectors(:,2),trainingdata(:,:,1)*descendingordervect
ors(:,3)]
projtrainingdata2=[trainingdata(:,:,2)*descendingordervectors(:,1),trainingda
ta(:,:,2)*descendingordervectors(:,2),trainingdata(:,:,2)*descendingordervect
ors(:,3)]
projtrainingdata3=[trainingdata(:,:,3)*descendingordervectors(:,1),trainingda
ta(:,:,3)*descendingordervectors(:,2),trainingdata(:,:,3)*descendingordervect
ors(:,3)]
STACKEDprojectedtrainingdata=[projtrainingdata1;projtrainingdata2;projtrainin
gdata3]
% Projecting test Data
pt1=[x(:,:,1)*descendingordervectors(:,1),x(:,:,1)*descendingordervectors(:,2
),x(:,:,1)*descendingordervectors(:,3)]
pt2=[x(:,:,2)*descendingordervectors(:,1),x(:,:,2)*descendingordervectors(:,2
),x(:,:,2)*descendingordervectors(:,3)]
pt3=[x(:,:,3)*descendingordervectors(:,1),x(:,:,3)*descendingordervectors(:,2
),x(:,:,3)*descendingordervectors(:,3)]

STACKEDprojectedtestdata=[pt1;pt2;pt3]




class1=0
class2=0
class3=0
for H=1:length(testdata2d)
c1=0
c2=0
c3=0
for T=1:length(STACKEDprojectedtrainingdata);
distance=norm(STACKEDprojectedtestdata(H,:)-
STACKEDprojectedtrainingdata(T,:));
```

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

```
d(T)= distance;


end
[B,I] = sort(d);

B=B(1,1:K);
I=I(1,1:K);


for NY=1:length(I);
    if I(NY)<=length(trainingdata2dimensional)/3
        c1=c1+1;
    elseif I(NY)<=length(trainingdata2dimensional)*2/3
        c2=c2+1;
    else
        c3=c3+1;
    end
end
%This if statement floors it to classify as class one if theyre all equal,
%and so on. I am not sure if this logic produces the best classifier, the
%only other way would be to assign weights instead of doing it randomly
%like this
if c1==c2 && c1==c3
    class1=class1+1
elseif c1==c2 && c1>c3
    class1=class1+1
elseif c2==c3 && c2>c1
    class2=class2+1
elseif c3==c1 && c3>c2
    class3=class3+1
elseif c1>c2 && c1>c3
     class1=class1+1
 elseif c2>c1 && c2>c3
     class2=class2+1
 elseif c3>c1 && c3>c2
     class3=class3+1
end

if H<=length(STACKEDprojectedtestdata)/3
    counter1=class1;
    class2=0;
    class3=0;
elseif H<=length(STACKEDprojectedtestdata)*2/3
    counter2=class2;
    class3=0;
else
    counter3=class3;
end

end
classifierrate=counter1/SIZE(1)+counter2/SIZE(1)+counter3/SIZE(1);
performance=classifierrate/3;
```

```
end
```

Problem 5 (h-L):

```
function [performance] = KNNLDA(x,trainingdata,P,K)

% x is the test sample, what you want to classify
% training data
% P is the prior
% K is the input for the KNN, how many points you want to include in the
% hypervolume

SIZE=size(x);

classiferratetotal=zeros(1,SIZE(3));

trainingdata2dimensional=[trainingdata(:,:,1);trainingdata(:,:,2);trainingdat
a(:,:,3)];

testdata2d=[x(:,:,1);x(:,:,2);x(:,:,3)];

%LDA STUFF (h-L)
[y, v, d] = tamu_lda(trainingdata2dimensional,
[ones(length(trainingdata),1);2*ones(length(trainingdata),1);3*ones(length(tr
ainingdata),1)]);

%Projecting training Data
projtrainingdata1=[trainingdata(:,:,1)*v(:,1),trainingdata(:,:,1)*v(:,2)]
projtrainingdata2=[trainingdata(:,:,2)*v(:,1),trainingdata(:,:,2)*v(:,2)]
projtrainingdata3=[trainingdata(:,:,3)*v(:,1),trainingdata(:,:,3)*v(:,2)]
STACKEDprojectedtrainingdata=[projtrainingdata1;projtrainingdata2;projtrainin
gdata3]
% Projecting test Data
pt1=[x(:,:,1)*v(:,1),x(:,:,1)*v(:,2)]
pt2=[x(:,:,2)*v(:,1),x(:,:,2)*v(:,2)]
pt3=[x(:,:,3)*v(:,1),x(:,:,3)*v(:,2)]

STACKEDprojectedtestdata=[pt1;pt2;pt3]




class1=0
class2=0
class3=0
for H=1:length(testdata2d)
c1=0
c2=0
c3=0
for T=1:length(STACKEDprojectedtrainingdata);
```

Assignment 2
Mahmood Mustafa Shilleh
02/24/2020

```
distance=norm(STACKEDprojectedtestdata(H,:)-
STACKEDprojectedtrainingdata(T,:));

d(T)= distance;



end
[B,I] = sort(d);

B=B'
I=I'
B=B(1,1:K);
I=I(1,1:K);



for NY=1:length(I);
    if I(NY)<=length(trainingdata2dimensional)/3
        c1=c1+1;
    elseif I(NY)<=length(trainingdata2dimensional)*2/3
        c2=c2+1;
    else
        c3=c3+1;
    end
end
%This if statement floors it to classify as class one if theyre all equal,
%and so on. I am not sure if this logic produces the best classifier, the
%only other way would be to assign weights instead of doing it randomly
%like this
if c1==c2 && c1==c3
    class1=class1+1
elseif c1==c2 && c1>c3
    class1=class1+1
elseif c2==c3 && c2>c1
    class2=class2+1
elseif c3==c1 && c3>c2
    class3=class3+1
elseif c1>c2 && c1>c3
     class1=class1+1
 elseif c2>c1 && c2>c3
     class2=class2+1
 elseif c3>c1 && c3>c2
     class3=class3+1
end

if H<=length(STACKEDprojectedtestdata)/3
    counter1=class1;
    class2=0;
    class3=0;
elseif H<=length(STACKEDprojectedtestdata)*2/3
    counter2=class2;
    class3=0;
else
    counter3=class3;
end
```

```
end
classifierrate=counter1/SIZE(1)+counter2/SIZE(1)+counter3/SIZE(1);
performance=classifierrate/3;

end
```

Problem 6 (a):

```
close all
clear all
clc
Traininglabels = csvread('hw2p6_c1.csv');
A=sort(Traininglabels)
Trainingvalues = csvread('hw2p6_x1.csv');
Testlabels = csvread('hw2p6_c2.csv')
Testvalues = csvread('hw2p6_x2.csv')

MU=mean(Trainingvalues);
COV=cov(Trainingvalues)

[vectors, values]=eig(COV)
eigenvalues=diag(values)
[descendingorder, index]=sort(eigenvalues,'descend')
[descendingordervectors]=vectors(:,index)

sumeig=sum(eigenvalues)
PofV=0
H=0


while PofV<0.9
    H=H+1 %Counting how many eigenvalues we need in the PCA to capture 90
percent of the vartiance
    PofV=(PofV+(descendingorder(H)/sumeig))
end

 [y, v, d] = tamu_lda(Trainingvalues, Traininglabels); % Using LDA to see how
much of the variance is captured
```