Homework Set 3, CPSC 8420, Spring 2022

Last Name, First Name

Due 03/31/2022, Thursday, 11:59PM EST

Problem 1

Given data-points $\{\{1,3\},\{2,5\},\{3,4\},\{4,3\},\{5,2\},\{5,1\}\}$.

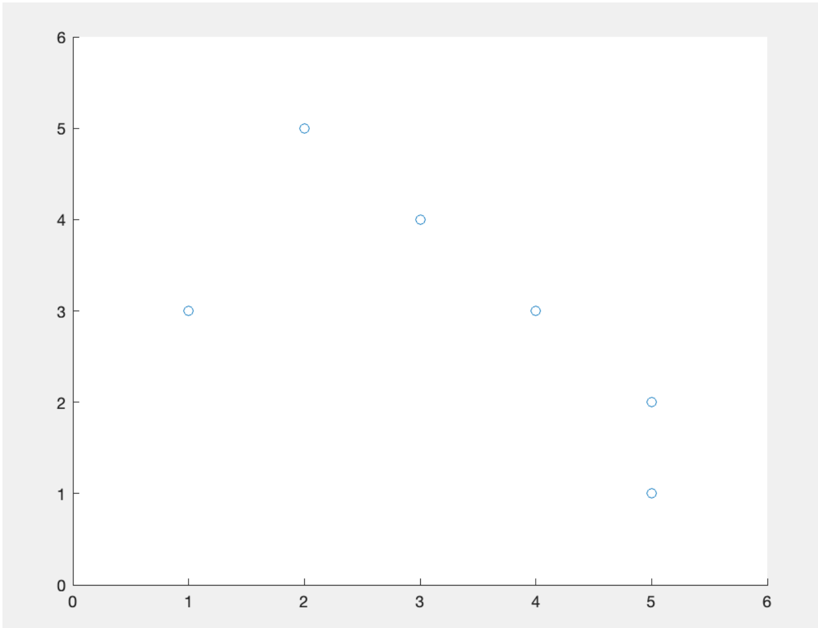
1. Please scatter-plot each data point within one figure (you can use Matlab, Python or any other programming language).

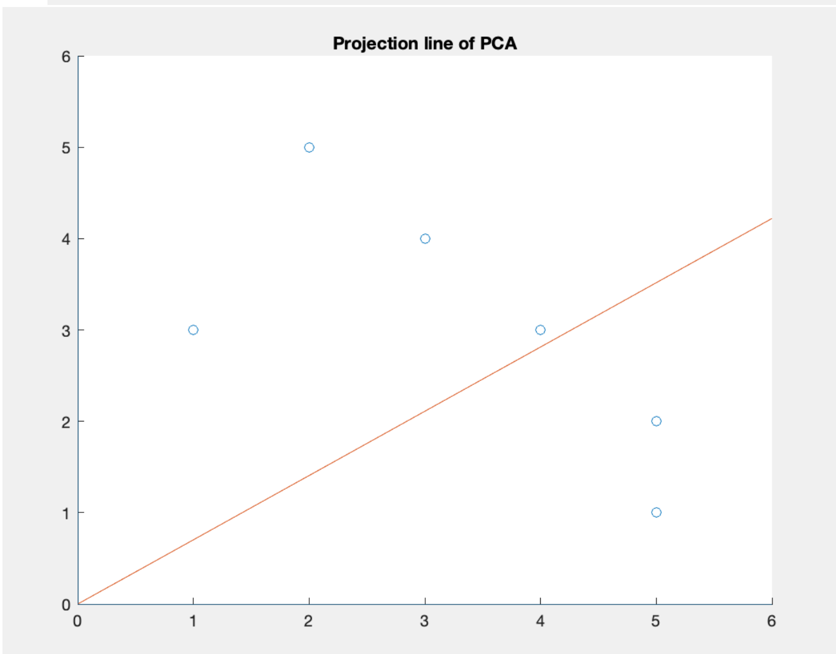
2. Now if we want to reduce the dimension from 2 to 1 by PCA, please determine the projection line which crosses the origin (please plot the line based on the scatter-plot figure above).

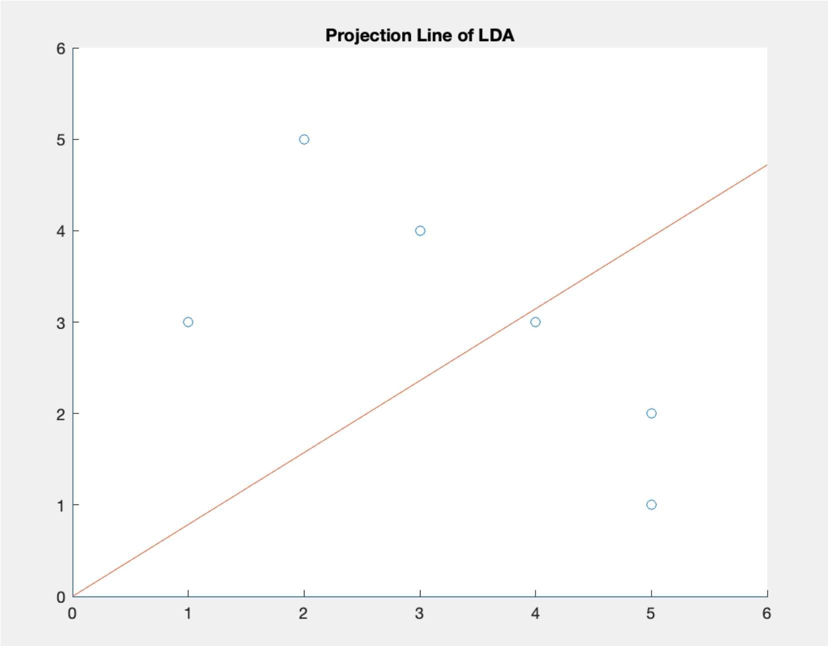
   For the PCA projection line. I calculated the Covariance matrix and then performed an SVD on that matrix. I took the slope between the two points of the first column of U and got a slope of 0.7034.

3. Assume the first 4 data points belong to one class, while the rest 2 belong to the other. Now if we want to reduce the dimension from 2 to 1 by LDA, please determine the projection line which crosses the origin (you are expected to plot the line based on the scatter-plot figure).

   For the LDA projection line I calculated the weights (W) of the data, multiplied that with the data and then calculated the slope using the first and last data point. Which I got to be 0.78

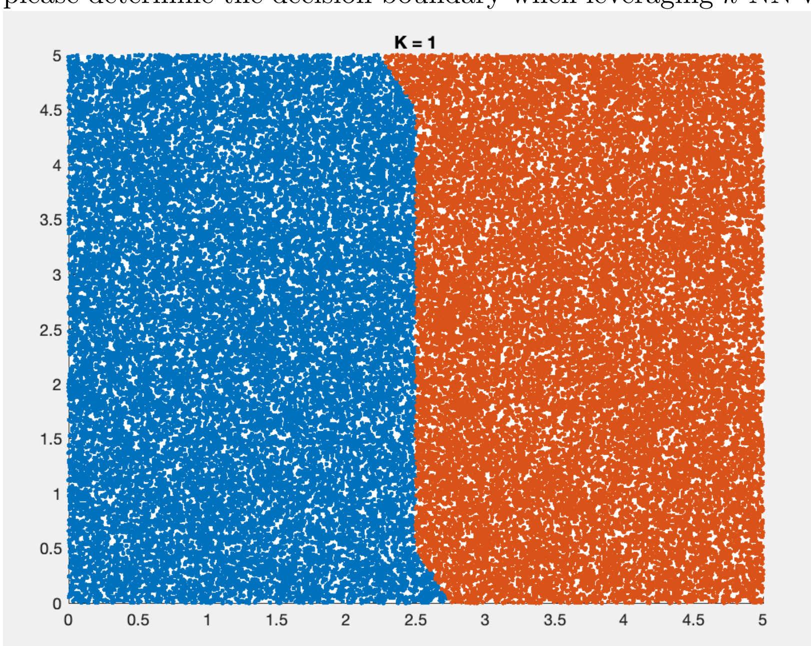

6
5
4
3
2
1
0
0
1
2
3
4
5
6

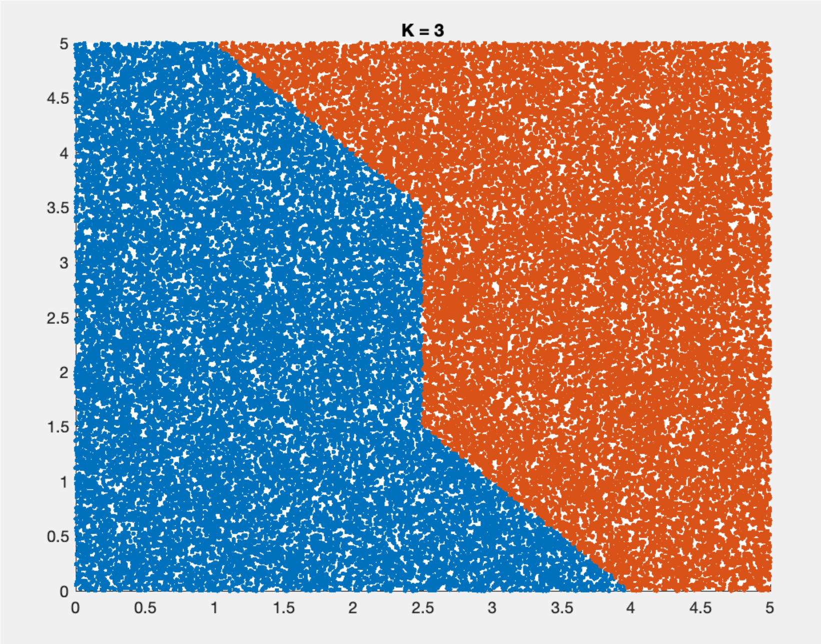

Projection line of PCA
6
5
4
3
2
1
0
0
1
2
3
4
5
6



Problem 2

Given positive data-set $\{\{1,1\},\{2,2\},\{2,3\}\}$, as well as negative data-set $\{\{3,2\},\{3,3\},\{4,4\}\}$, please determine the decision boundary when leveraging k-NN where $k=1$ and $k=3$ respectively.





Problem 3

Given X, Y, Z, now please follow the idea/method used in LDA/PCA to find the best solution to:

$$\begin{aligned} &\underbrace{arg\ max}_{a,b}\ \ a^T Z b \\ s.t.\ \ &a^T X a = 1,\ b^T Y b = 1 \end{aligned} \tag{1}$$

Since X, Y, Z are SPD we can expand our constraints $s.t.\ a^T X^{1/2} X^{1/2} a = 1,\ b^T Y^{1/2} Y^{1/2} b = 1$

Let $u = X^{1/2}a$ and $v = Y^{1/2}b$, therefore we have that $u^T u = 1$ and $v^T v = 1$

We can arrange to say that $a^T = u^T X^{1/2}$ and $b = Y^{1/2}v$ and therefore we $max\ u^T X^{-1/2} Z Y^{-1/2} v$

If we use SVD we find that $[U, \Sigma, V] = svd(X^{-1/2} Z Y^{-1/2})$. We can then let u be the first column of U, and v be the first column of V, and then find a and b.

Code

NOTE: For plotting data points on graph I just removed some aspects of the PCA projection code.
PCA Projection Line Code

```
clc; clear;
a = [[1,3];[2,5];[3,4];[4,3];[5,2];[5,1]];

```

```
M = mean(a);
lower_bound = 0;
upper_bound = 6;

a_new = a - M;
C = cov(a)
[U,S,V] = svd(C);
points = a * U(:, 1)
slope_of_projection_line = (U(2,1) - U(1,1))/2;

x = [lower_bound, upper_bound];
y = slope_of_projection_line * x;


h = figure;
scatter(a(:, 1), a(:, 2));
hold on
plot(x, y)
xlim([lower_bound, upper_bound])
ylim([lower_bound, upper_bound])
line([0 0], ylim);
line(xlim, [0 0]);
title("Projection line of PCA")
hold off
waitfor(h)
```

LDA Projection Line

```
clc; clear;
x_lower_lim = 0;
x_upper_lim = 6;
a = [[1,3];[2,5];[3,4];[4,3];[5,2];[5,1]];
%classes for data
M = mean(a);
%center data on axis
a_new = a - M;
a_val = [4;2];

[W, Y] =LDA(a_new, a_val);

slope = (Y(6) - Y(1))/5;
x = [x_lower_lim, x_upper_lim];
y = slope * x;


h = figure;
```

```
scatter(a(:, 1), a(:, 2));
hold on
%plot([-3, -2, -1, 1, 2, 3], Y)
plot(x, y);
xlim([x_lower_lim, x_upper_lim])
ylim([x_lower_lim, x_upper_lim])
line([0 0], ylim);
line(xlim, [0 0]);
title("Projection Line of LDA")
hold off
waitfor(h)
```

LDA Algorithm

```
function [W,y] = LDA(Data, a)
% This code is written by Alaa Tharwat (Frankfurt University of Applied
    Science - Germany)
% For more details about the code of the numerical example(s) see our
    paper "Tharwat, A., Gaber, T., Ibrahim, A.,
% & Hassanien, A. E. Linear discriminant analysis: A detailed tutorial.
    AI Communications,
% (Preprint), 1-22.?
% engalaatharwat@hotmail.com
%% Examples
% c1=[1 2;2 3;3 3;4 5;5 5]  % the first class (5 observations)
% c2=[4 2;5 0;5 2;3 2;5 3;6 3] % the second class (6 observations)
% data=[c1;c2] % te whole data
% a=[5;6] % the number of samples in each class
% [W,y] = LDA_ClassIndependent(data, a)
% Calculate the total number of all classes
c=unique(a);
pos=1;
for i=1:length(c)
    for j=1:a(i)
        Labels(pos,1)=i;
        pos=pos+1;
    end
end
% Calculate Mean of each class
for i=1:length(c)
    mu(i,1:size(Data,2))=mean(Data(Labels==i,:));
end
% Calculate the total mean of all classes
muTotal=zeros(size(mu(1,:)));
for i=1:length(c)
    muTotal=muTotal+a(i)*mu(i,:);
```

```
end
muTotal=muTotal/(sum(a));
% Subtract the originla data from the mean
D=zeros(size(Data,1),size(Data,2));
for i=1:length(c)
    D(Labels==i,:)=Data(Labels==i,:)-repmat(mu(i,:),a(i),1);
end
% Calculate the within class variance (SW)
SW=zeros(size(Data,2),size(Data,2));
for i=1:c
    SW=SW+D(Labels==i,:)'*D(Labels==i,:);
end
% Calculate the Between-class variance (SB)
SB=zeros(size(Data,2),size(Data,2));
for i=1:length(c)
   SB= SB+a(i)*(mu(i,:)-muTotal)'*(mu(i,:)-muTotal);
end
% Calculate J(W)
J=inv(SW)*SB;
% Calculate the eignevalues and eigenvectors of (J)
[evec,eval]=eig(J);
% Sort the eigenvectors according to their corresponding eigenvalues (descending order)
eval = diag(eval);
[junk, index] = sort(-eval);
eval = eval(index);
evec = evec(:, index);
% Slect the most largest c eigenvectors as a lower dimensional space
W=evec(:,1:length(c)-1);
% project the original data on thelower dimensional space (W)
y=Data*W;
```

KNN Boundary Code

```
clc;
rng(1)
positive = [[1,1];[2,2];[2,3]];
negative = [[3,2];[3,3];[4,4]];
data = [positive; negative];
value = [1;1;1;2;2;2];
x = rand(50000, 2) * 5;


k_list = [1,3];

for i = 1:2
```

```
    k = k_list(i);
    [pred, idx, acc] = KNN(k, data, value, x);

    class_1 = [];
    class_2 = [];
    for i=1:length(pred)
        if pred(i) == 1
            class_1 = [class_1; x(i, 1:2)];
        else
            class_2 = [class_2; x(i, 1:2)];
        end
    end

    h= figure;
    %scatter(data(:, 1), data(:, 2),500, '.');
    hold on
    scatter(class_1(:, 1), class_1(:, 2), 50, '.');
    scatter(class_2(:, 1), class_2(:, 2), 50, '.');
    xlim([0, 5])
    ylim([0, 5])
    t = "K = " + string(k);
    title(t)
    hold off
    waitfor(h)
end
```

KNN Algorithm

```
function [predicted_labels,nn_index,accuracy] = KNN(k,data,labels,t_data,t_labels)
%KNN_: classifying using k-nearest neighbors algorithm. The nearest neighbors
%search method is euclidean distance
%Usage:
%       [predicted_labels,nn_index,accuracy] = KNN_(3,training,training_labels,testing,testing_labels)
%       predicted_labels = KNN_(3,training,training_labels,testing)
%Input:
%       - k: number of nearest neighbors
%       - data: (NxD) training data; N is the number of samples and D is the
%       dimensionality of each data point
%       - labels: training labels
%       - t_data: (MxD) testing data; M is the number of data points and D
%       is the dimensionality of each data point
```

```
%       - t_labels: testing labels (default = [])
%Output:
%       - predicted_labels: the predicted labels based on the k-NN
%       algorithm
%       - nn_index: the index of the nearest training data point for each training sample (Mx1).
%       - accuracy: if the testing labels are supported, the accuracy of
%       the classification is returned, otherwise it will be zero.
%Author: Mahmoud Afifi - York University
%checks
if nargin < 4
    error('Too few input arguments.')
elseif nargin < 5
    t_labels=[];
    accuracy=0;
end
if size(data,2)~=size(t_data,2)
    error('data should have the same dimensionality');
end
if mod(k,2)==0
    error('to reduce the chance of ties, please choose odd k');
end
%initialization
predicted_labels=zeros(size(t_data,1),1);
ed=zeros(size(t_data,1),size(data,1)); %ed: (MxN) euclidean distances
ind=zeros(size(t_data,1),size(data,1)); %corresponding indices (MxN)
k_nn=zeros(size(t_data,1),k); %k-nearest neighbors for testing sample (Mxk)
%calc euclidean distances between each testing data point and the training
%data samples
for test_point=1:size(t_data,1)
    for train_point=1:size(data,1)
        %calc and store sorted euclidean distances with corresponding indices
        ed(test_point,train_point)=sqrt(...
            sum((t_data(test_point,:)-data(train_point,:)).^2));
    end
    [ed(test_point,:),ind(test_point,:)]=sort(ed(test_point,:));
end
%find the nearest k for each data point of the testing data
k_nn=ind(:,1:k);
nn_index=k_nn(:,1);
%get the majority vote
```

```
for i=1:size(k_nn,1)
    options=unique(labels(k_nn(i,:)'));
    max_count=0;
    max_label=0;
    for j=1:length(options)
        L=length(find(labels(k_nn(i,:)')==options(j)));
        if L>max_count
            max_label=options(j);
            max_count=L;
        end
    end
    predicted_labels(i)=max_label;
end
%calculate the classification accuracy
if isempty(t_labels)==0
    accuracy=length(find(predicted_labels==t_labels))/size(t_data,1);
end
```