Ellis Wright Feedback sheet: Assignment # 3

This feedback sheet is meant to help you to improve your assignment reports, and to understand the grading criteria.

Part of the work of an applied mathematician is to communicate with experts that may not be mathematicians. Therefore, it is important to develop one's reporting skills. In this course, the quality of the reporting is an important element in grading.

Below, a list of important points concerning the reporting are given, and it also serves as a rubric for feedback.

1. The report should be self-explanatory, not an answer to a question. This means that you have to explain what the problem is that you are addressing. Preferably, do not just copy the assignment but explain in your own words what the assignment asks you to do. **A printed code is not a sufficient answer!**

   Your grade:

   ☐ Excellent: No need to improve
   ☐ Good: Acceptable, leaving space for improvement
   ☐ Below standard : Requires more work

2. The report should be carefully typeset, paying attention to the layout. Sloppy reporting gives a negative impression and raises questions about the correctness of the results, so one needs to pay attention on the visual quality.

   Your grade:

   ☐ Excellent: No need to improve
   ☐ Good: Acceptable, leaving space for improvement
   ☐ Below standard : Requires more work

3. Well planned graphs and plots are an integral part of good reporting. When you include a plot in the report, pay attention to the following:

   (a) Make sure that the fonts are of readable size. The default font in Matlab is not acceptable, so increase the font size, e.g., using the command `set(gca,'FontSize',15)`.

   (b) Lines and curves need to be thick enough to be readable, and markers must be large enough.

   (c) Pay attention to axes. For instance, if you plot a circle, make sure that it does not appear as an ellipse, using commands like `axis('square')` or `axis('equal')`. Also, think about the scaling of the axis so that they convey the information you want. Sometimes a logarithmic scale is more informative.

   (d) Select the graphical output carefully. Is a histogram better than a continuous curve? Discrete quantities may be better represented by dots or bars than by continuous curves etc.

(e) Consider marking the axes with a label, using commands `xlabel, ylabel`.

Your grade:

- ☐ Excellent: No need to improve
- ☐ Good: Acceptable, leaving space for improvement
- ☐ Below standard : Requires more work

4. Technical correctness. Each assignment will be checked for correctness. If you get a grade less than excellent, go and check the comments on the report.

   Your grade:

   - ☐ Excellent: No need to improve
   - ☐ Good: Acceptable, leaving space for improvement
   - ☐ Below standard : Requires more work

5. Code: Including a code, or a piece of it is recommended when applicable. Pay attention on your code: A code with no comments is below standard. Well documented code is a great asset for yourself, too!

   Your grade:

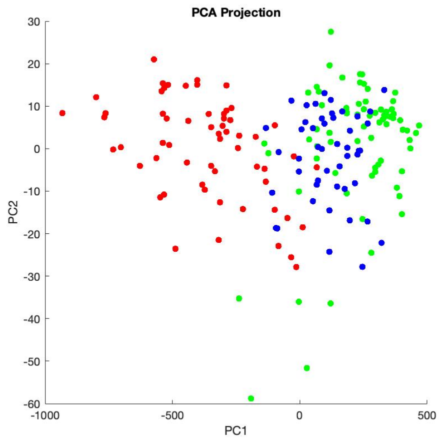
   - ☐ Excellent: No need to improve
   - ☐ Good: Acceptable, leaving space for improvement
   - ☐ Below standard : Requires more work

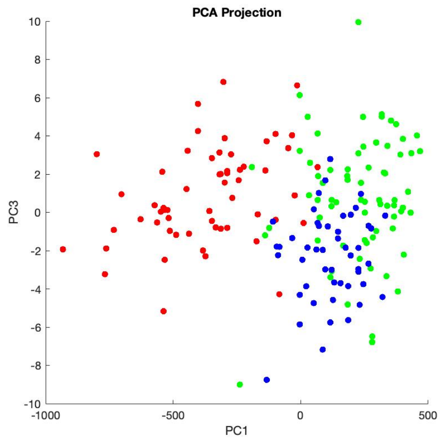
MATH 444 **Due Date: 02/27/2022**

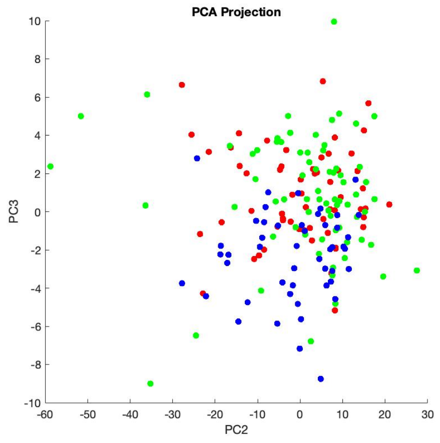
Assignment 3

Ellis Wright (enw30)

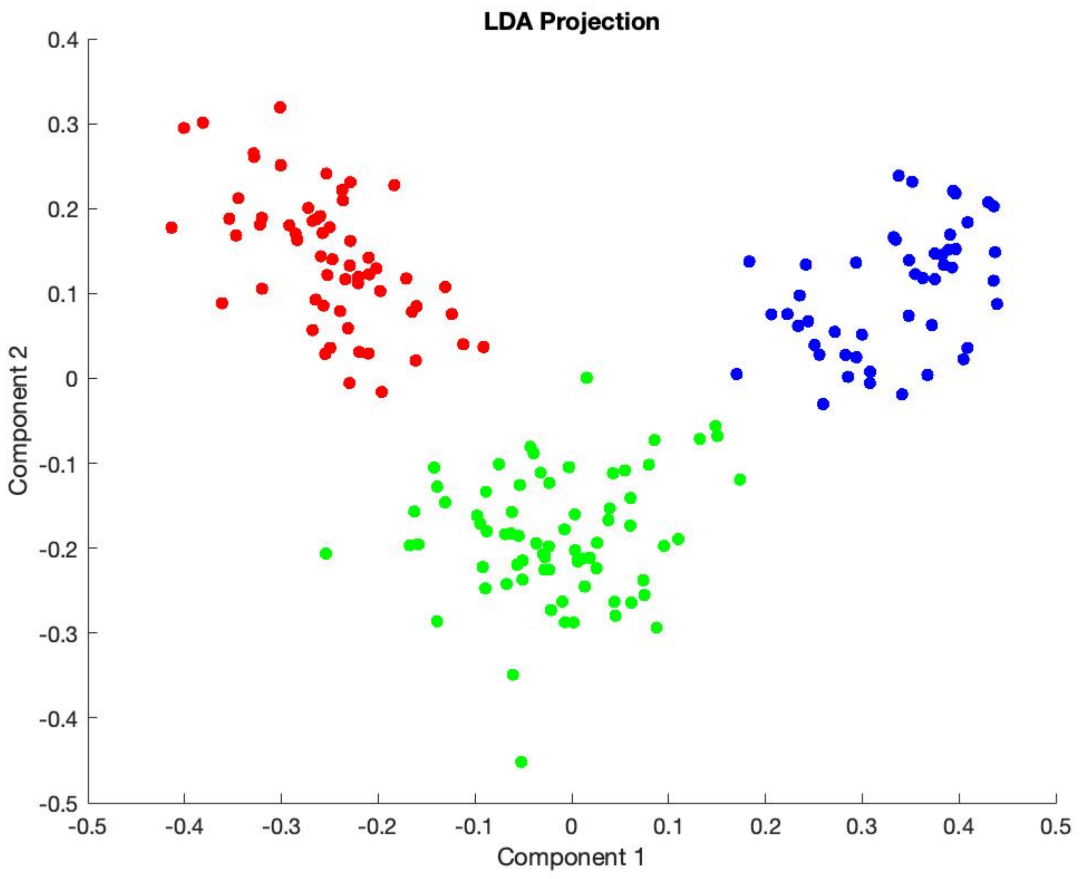
Building off of last assignment, I will compare how linear discriminant analysis (LDA) compares to Lloyd's algorithm on the Wine data set. This data comes from a chemical analysis of wines grown in the same region in Italy. The analysis included 30 attributes but the data consists of 13, you can learn more about it *here*. The goal is to classify which of the three cultivars a wine sample is derived from.
When viewing the results of the clusters formed from k-means it was clear the principle components did not create a clear distinction between the classes. Here are the plots for the first three principle components.





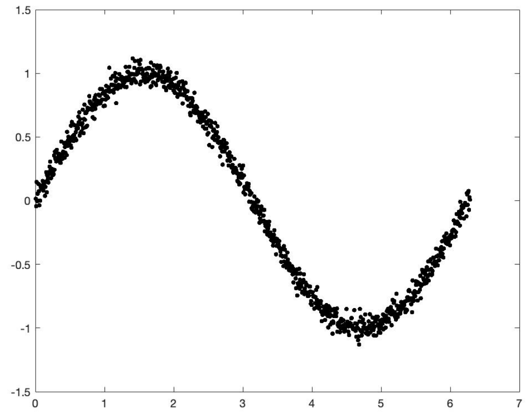
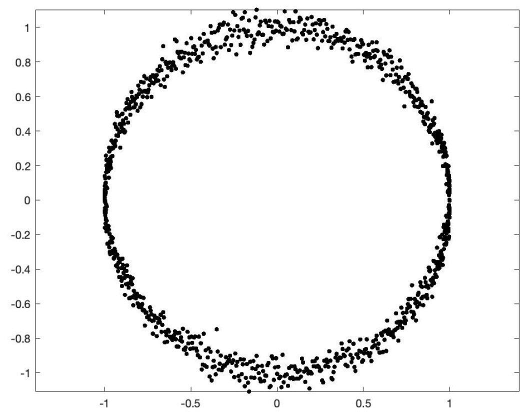
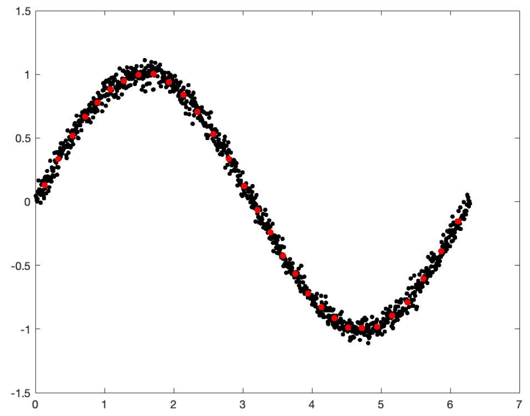
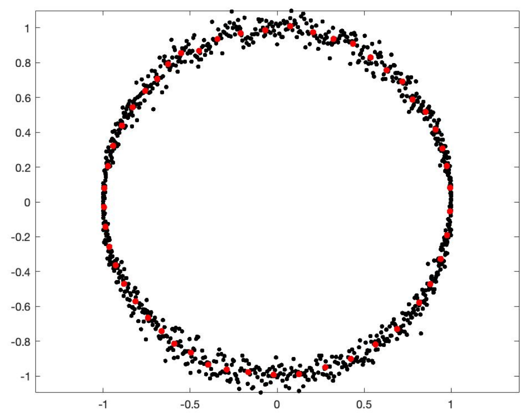


As you can see, there is no easy way to separate all three classes. However, when we view the LDA-reduced matrices we get a different view:

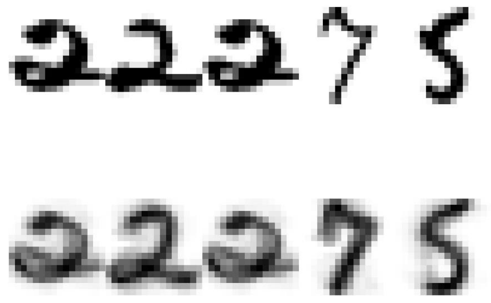


As you can see, LDA does a much better job at separating the classes than PCA does on the Wine data set. PCA is finding the direction that maximizes the spread of the data while LDA also takes into consideration the spreads between the clusters. I believe the eignevector with the largest magnitude corresponds to the nonflavanoid-phenols measurement, suggesting that nonflavanoid-phenols are an important indicator for this classification problem.

LDA is cool and all but next I will investigate how a self organizing map can learn the organization of some datasets. To start off I generated two data sets, a sine wave and a circle with some added noise.

The bottom figure has the prototypes in red layered on top of the original graph. I used 30 and 50 prototypes for the sine wave and circle, respectively. I get nearly the same results whether I use a one or two dimensional lattice. Next, I was interested to see how it does in much higher dimensional data. For this reason, I will investigate how well the SOM forms to the handwritten dataset. The first thing I did was take a subset of the data that contains only 2s, 5s, and 7s. Next, I created a 10x10 lattice for this data. After running the SOM algorithm I picked random prototypes, and displayed the number closest to it in the data, and then also what the prototype looks like. Here are the results:



The bottom row is five random prototypes and the top row is the closest data point to it. As you can see, the prototypes do a good job at approximating the images. This can be used for classification in the future.

Code

Here is my code for the algorithms and the figures. The LDA helper functions:

```
function [data,labels,centers,k] = process_data(X,I)

    %get all the classes
    classes = unique(I);
    k = length(classes);

    %store the sorted data and labels
    data = zeros(size(X));
    labels = zeros(1,length(data));

    %store the k centers
    n = length(X(:,1));
    centers = zeros(n,k);

    %keep track of the current index of the submatrix
    current_length = 1;

    for i = 1:k

        %get the cluster
        indices = (I == classes(i));
        cluster = X(:,indices);
        %calculate the mean
        centers(:,i) = mean(cluster,2);

        %count the size of this submatrix
        num_samples = sum(indices);
        final_length = current_length + num_samples-1;

        %put the data in order and center it, update the labels
        data(:,current_length:final_length) = cluster - centers(:,i);
        labels(current_length:final_length) = i;

        %update the new starting index
        current_length = final_length + 1;
    end
end
```

The LDA algorithm:

```
function directions = LDA(X,I)

    %format the dataset and get the centers of the clusters
    [data,labels,cluster_means,k] = process_data(X,I);

    %center the mean matrix and calclulate the between cluster scatter
    %matrix
    global_mean = mean(cluster_means,2);
    centered_means = cluster_means - global_mean;
    Sb = centered_means*centered_means';

    %calculate the within cluster scatter matrix and make sure it is
    %regularized to ensure it is spd
    Sw = data*data';
    eigen = eigs(Sw,1);
    error = eigen*(10^-10);
    Sw = Sw + error*eye(length(X(:,1)));

    %find the cholesky facotrization
    K = chol(Sw);

    %the eigen vectors of this expression maximize its value
    [eigen_vectors,~] = eigs(inv(K') * Sb * inv(K),k-1);

    %get the diretions corresponding to these vectors
    directions = inv(K)*eigen_vectors;
end
```

The submission generation:

```
load('WineData.mat')

%calculating directions and getting reduced matrix
Q = LDA(X,I);
Z = Q'*(X- mean(X,2));

%finding indices associated to each class
one = find(I == 1);
two = find(I == 2);
three = find(I == 3);

%plotting the reduced data
hold on
plot(Z(1,one),Z(2,one),'r.','MarkerSize',15)
plot(Z(1,two),Z(2,two),'g.','MarkerSize',15)
plot(Z(1,three),Z(2,three),'b.','MarkerSize',15)
xlabel 'Component 1'
ylabel 'Component 2'
title 'LDA Projection'

%getting the PCA of the data
Xc = X - mean(X,2);
[U,S,V] = svd(Xc);
ZZ = U'*Xc;

%plotting the PCs
figure(2)
title 'PCA Projection'
xlabel PC1
ylabel PC2
hold on
plot(ZZ(1,one),ZZ(2,one),'r.','MarkerSize',15)
plot(ZZ(1,two),ZZ(2,two),'g.','MarkerSize',15)
plot(ZZ(1,three),ZZ(2,three),'b.','MarkerSize',15)
axis square

figure(3)
title 'PCA Projection'
xlabel PC1
ylabel PC3
hold on
plot(ZZ(1,one),ZZ(3,one),'r.','MarkerSize',15)
plot(ZZ(1,two),ZZ(3,two),'g.','MarkerSize',15)
plot(ZZ(1,three),ZZ(3,three),'b.','MarkerSize',15)
axis square
```

```
figure(4)
title 'PCA Projection'
xlabel PC2
ylabel PC3
hold on
plot(ZZ(2,one),ZZ(3,one),'r.','MarkerSize',15)
plot(ZZ(2,two),ZZ(3,two),'g.','MarkerSize',15)
plot(ZZ(2,three),ZZ(3,three),'b.','MarkerSize',15)
axis square
```

SOM helper functions:

```
%this function creates a lattice and stores the points in a matrix
function lattice = create_lattice(size)
    x = size(1);
    y = size(2);
    lattice = zeros(2,x*y);
    counter = 1;
    for i = 1:x
        for j = 1:y
            lattice(:,counter) = [i,j];
            counter = counter + 1;
        end
    end
end

function D = create_distance_mat(X,medoids)
    n = length(X);
    D = nan(n);
    for i = 1:n
        for j = 1:n
            %if symmetric point is already filled in, copy it
            if(not(isnan(D(j,i))))
                D(i,j) = D(j,i);
            else
                if medoids
                    D(i,j) = medoid_dist(X(:,i),X(:,j));
                else
                    D(i,j) = norm(X(:,i)-X(:,j));
                end
            end
        end
    end
end
```

The SOM algorithm:

```
function prototypes = SOM(X,size,params,learning_time)

%initializing lattice and prototypes
lattice = create_lattice(size);
D = create_distance_mat(lattice,false);
K = size(1)*size(2);
T_max = 500*K;
p = length(X(1,:));
prototypes = X(:,randsample(p,K));

%this code is used for Question 3
%prototypes = zeros([256,K]);
%for i = 1:K
%   prototypes(:,i) = mean(X,2) + normrnd(0,.1,[1,256])';
%end

t = 1;

while t < T_max

    %pick a data point
    x = X(:,randi(p));

    %get index of best matching unit
    [~,BMU] = min(vecnorm(prototypes - x));

    %calculate the parameters
    alpha = max(params(1,1)*(1 - (t/learning_time)),params(1,2));
    gamma = max(params(2,1)*(1 - (t/learning_time)),params(2,2));

    %generate neighborhood matrix column for BMU
    h = exp((D(:,BMU).^2)/(-2*(gamma^2)));

    %update the prototypes
    for i = 1:K
        d = (x - prototypes(:,i));
        prototypes(:,i) = prototypes(:,i) + alpha*h(i)*d;
    end
    t = t + 1;
end
end
```

SOM submission generation:

```
%set the parameters
```

```
lattice = [1,50];
list = [1,30];

params1 = [.9, .01; max(list)/3, .5];
params2 = [.9, .01; max(lattice)/3, .5];

%create sine and circle data
t = 2*pi*[1:1000]/1000;

sinus = [t;sin(t)];
circle = [cos(t);sin(t)];

%add some noise
sinus(2,:) = sinus(2,:) + normrnd(0,.05,[1,1000]);
circle(2,:) = circle(2,:) + normrnd(0,.05,[1,1000]);

%calc approximations
m1 = SOM(sinus,list,params1,2000);
m2 = SOM(circle,lattice,params2,2000);

%plot the figures
figure(1)
plot(sinus(1,:),sinus(2,:),'k.','MarkerSize',10)
hold on
plot(m1(1,:),m1(2,:),'r.','MarkerSize',15)

figure(2)
plot(circle(1,:),circle(2,:),'k.','MarkerSize',10)
hold on
plot(m2(1,:),m2(2,:),'r.','MarkerSize',15)
axis equal
%% number 2
load('HandwrittenDigits.mat')
params = [.9, .01; 10/3, .5];

%get indices of 2,5,7
two = find(I == 2);
seven = find(I == 7);
five = find(I == 5);

%subset the data
sub = X(:,[two,five,seven]);

%get prototypes
M = SOM(sub,[10,10],params,1500);
```

```
V = zeros(256,100);
for i = 1:100
    [~,best] = min(vecnorm(X - M(:,i)));
    V(:,i) = X(:,best);
end

%generate plot of comparison
tiledlayout(2,5,'TileSpacing','none')
for j = 1:5
    k = randi(100);
    nexttile(j)
    imagesc(reshape(V(:,k),16,16)')
    axis square
    axis off
    nexttile(j+5)
    imagesc(reshape(M(:,k),16,16)')
    axis square
    axis off
    colormap(1-gray);
end
```