Math 444: Midterm Project

Iris Zhang

March 27, 2020

In this project, we want to assess the significance of the attributes in **WisconsinBreastCancer Dataset** that have been collected in determining whether a tumor is malignant or benign, rank them in terms of how well they captures the differences of interest, and more. In order to do that, we will be doing the following six questions.

Problem 1

Ignoring the labeling of the data, but assuming that the data belong to 2 different clusters, run the k-means and k-medoids algorithms developed in **Homework2** with k = 2. Since the algorithm that we developed for k-medoids in the previous homework uses 2-norm and the outcomes are not as good as we expected, here we modify that algorithm by using 1-norm instead.
First, we plot the some attributes of the data according to the original label given by experts, see Figure 1.

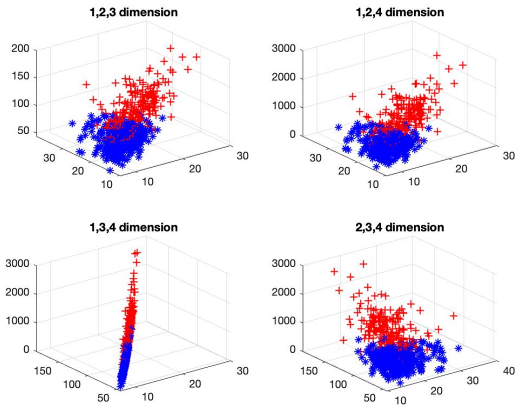


Figure 1: clustered data based on given label

Then, we plot the clustered data after we do k-means and k-medoids with the same attributes chosen for original label, see Figure 2 and 3.

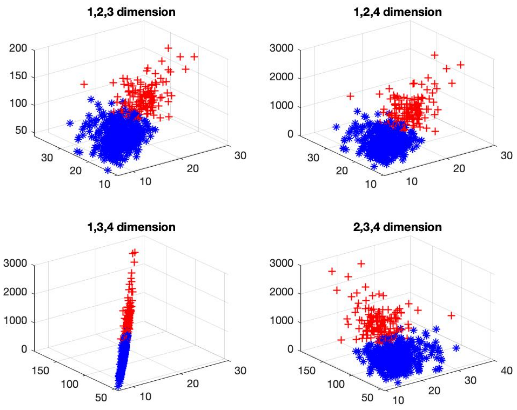


Figure 2: clustered data after doing k-means

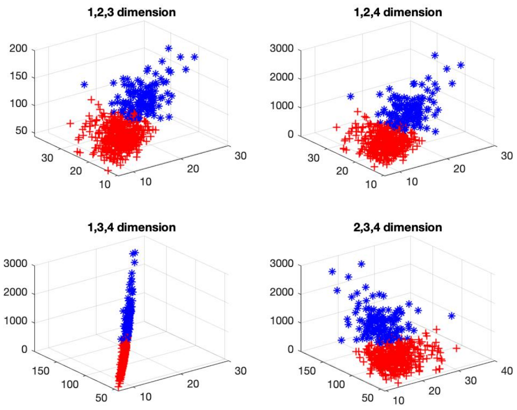


Figure 3: clustered data after doing k-medoids

From above graphs, we can see that the outcomes we get from doing k-means and k-medoids are both pretty good compared to the original label since the distribution of the selected attributes are pretty the same.

Problem 2

Then, we compute the LDA separator for the data based on the original labeling, which we have done in **Homework3**, see Figure 4.

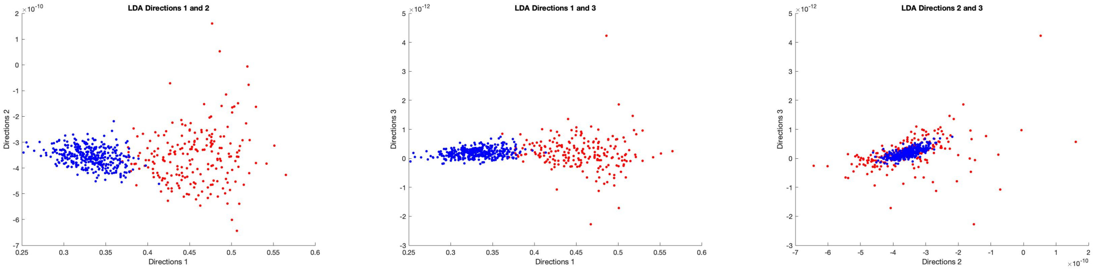


Figure 4: separation of WBC after doing LDA

From above graphs, we can see that the clusters for Direction 1 and 2 and Direction 1 and 3 are relatively clear but still not completely separated. And we can separate them by one-dimensional line in the data. But, there are no clear separations in the combination of Direction 2 and 3.

Problem 3

Now, we identify the attributes of the data that are most significant when it comes to separating Benign from Malignant and then plot the reduced data set with only the ten most significant attributes, see Figure 5.

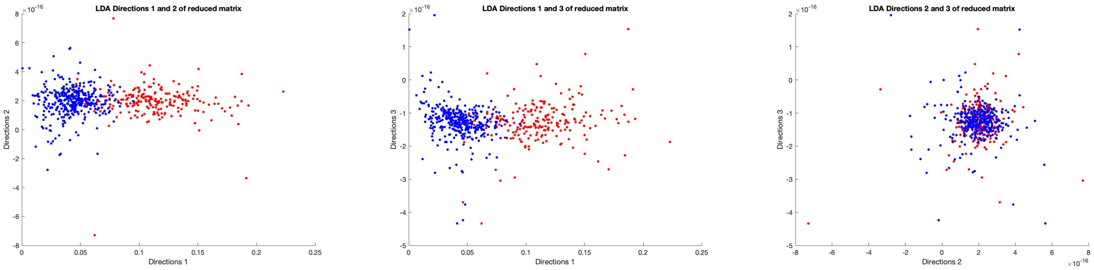


Figure 5: separation of WBC after doing LDA with ten most significant attributes

To be more clear whether the clusters is separated enough, we do the histogram plot, see Figure 6.

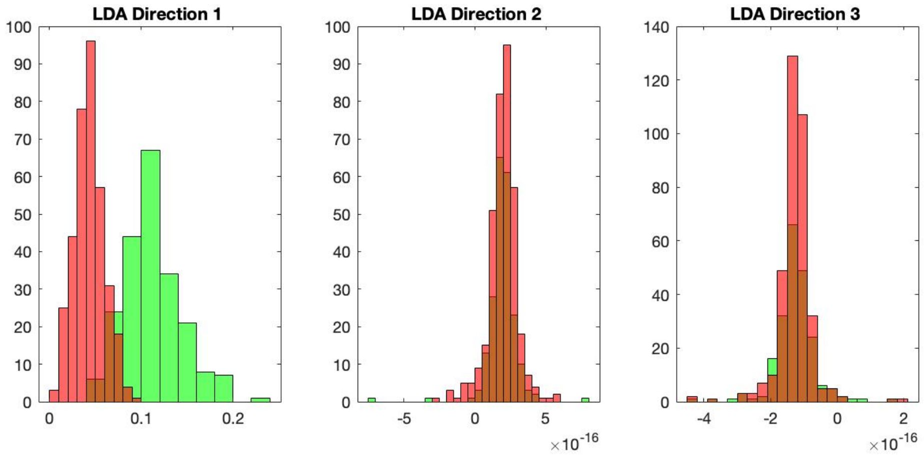


Figure 6: histogram plot of WBC after doing LDA with ten most significant attributes

From the histogram, we can see that the data can roughly separated in the first direction, but for Direction 2 and 3, the data are mixed together.

Problem 4

For this part, we take 50 Benign and 50 Malignant data points to be used as our Training set, and assign the remaining data as Test set.
(a)We construct a distance classifier and I include a snippet of my MATLAB code here:

```
I_distance = ones(1,tp);
Xb_c_bar = 1/50*sum(X_training_benign’);
Xm_c_bar = 1/50*sum(X_training_malignant’);
for i =1:tp
    norm_b = norm(X_testing(:,i)-Xb_c_bar’,2);
    norm_m = norm(X_testing(:,i)-Xm_c_bar’,2);
    if norm_b> norm_m
        I_distance(1,i)= 2;
    end
end
```

In order to see the performance of the distance classifier, we compute the confusion matrix C as following: C = [C11 C12; C21 C22] where C11 = number of data points whose original label is 1 and the cluster made by distance classifier is also 1; C12 = number of data points whose original label is 1 but the cluster made by distance classifier is 2; C21 = number of data points whose original label is 2 but the cluster made by distance classifier is 1; C22 = number of data points whose original label is 2 and the cluster made by distance classifier is also 2. The MATLAB code is attached here as well for clarification:

```
C = zeros(2,2);
C_11 = 0;
C_12 = 0;
C_21 = 0;
C_22 = 0;
for i = 1:n
```

```
    if (I_Label(i,1) == 1 & I_distance(1,i) == 1)
        C_11 = C_11+1;
    end
    if (I_Label(i,1) == 1 & I_distance(1,i) == 2)
        C_12 = C_12+1;
    end
    if (I_Label(i,1) == 2 & I_distance(1,i) == 1)
        C_21 = C_21+1;
    end
    if (I_Label(i,1) == 2 & I_distance(1,i) == 2)
        C_22 = C_22+1;
    end
end
C = [C_11 C_12; C_21 C_22]
```

Then, we get that for the testing data, C11+C22 = 429 and C12+C21 = 40, which shows that the distance classifier can roughly separate the two clusters but there still exists errors with about 91.5 percent accuracy.

(b)We construct k nearest neighbor classifier and I include a snippet of my MATLAB code here:

```
function I_knn = KNN(X_train,I_train, X_test, I_test, k)
    n_train=size(I_train,2);
    n_test=size(I_test,2);

    I_knn=zeros(1,n_test);

    for i=1:n_test
        distance=zeros(1, n_train);
        X_c= X_test(:,i);
        for j=1:n_train
            distance(j)=sqrt(sum((X_train(:,j)- X_c).^2));
        end
        [~,index]=sort(distance);
        k_nearest=I_train(index(1:k));
        I_knn(i)=mode(k_nearest);
    end
end
```

We apply this algorithm to our WBC data with k=3,4,5 and compute the confusion matrix in the same way as we did for distance classifier.
Then, we get that:
For k=3, C11+C22 = 432 and C12+C21 = 34;
For k=4, D11+D22 = 424 and D12+D21 = 45;
For k=5, E11+E22= 427 and E12+E21 = 42.
The performance of K-nearest neighbor algorithm can roughly separate the data into the targeted two clusters with about 92 percent accuracy.

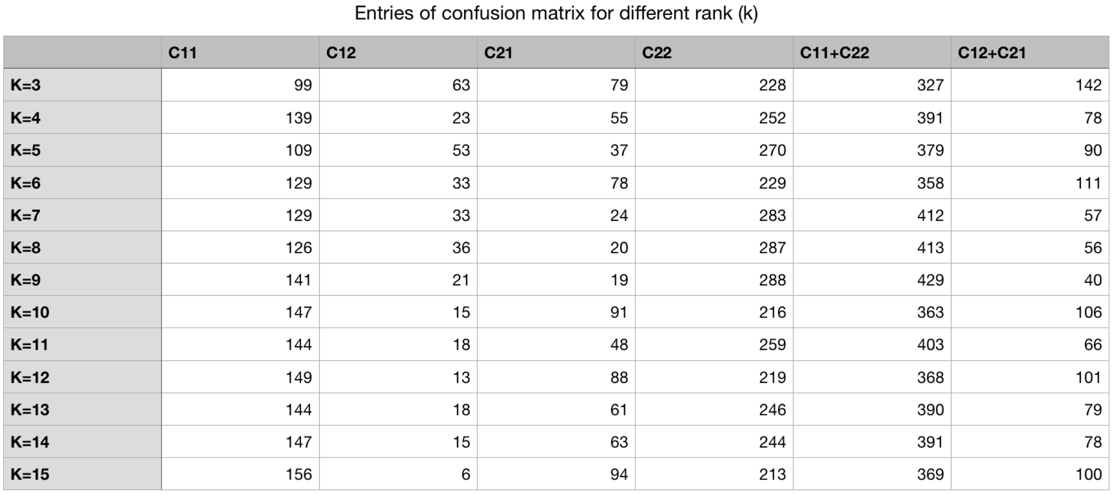
(c) Now, we construct a PCA classifier and here is the MATLAB code:

```
function I_pca = PCA(X_train,X_test,I_train,I_test,rank)
    clusters=unique(I_train);
    k=size(clusters,2);
    x=cell(1,k);

    for i=1:k
        x{i}=X_train(:,I_train==clusters(i));
    end
    u=cell(1,k);
    d=cell(1,k);
    v=cell(1,k);
    %Compute SVD
    for i=1:k
        [U,D,V]=svd(x{i});
        u{i}=U;
        d{i}=D;
        v{i}=V;
    end
    P=cell(1,k);
    for i=1:k
        u{i}=u{i}(:,1:rank);
        P{i}=u{i}* u{i}’;
    end
    [n,p]=size(X_test);
    clusters=unique(I_test);
    num_clusters=size(clusters,2);
    I_pca=zeros(1,p);

    for i=1:p
        norms=zeros(1,num_clusters);
        for j=1:num_clusters
            norms(j)=norm(P{j}*X(:,i), 2);
        end
        [˜,m]=max(norms);
        I_pca(i)=clusters(m);
    end
end
```

Then, we choose the different values of rank k and compute the confusion matrix for each of them. The results we get are the following:



Entries of confusion matrix for different rank (k)

	C11	C12	C21	C22	C11+C22	C12+C21
K=3	99	63	79	228	327	142
K=4	139	23	55	252	391	78
K=5	109	53	37	270	379	90
K=6	129	33	78	229	358	111
K=7	129	33	24	283	412	57
K=8	126	36	20	287	413	56
K=9	141	21	19	288	429	40
K=10	147	15	91	216	363	106
K=11	144	18	48	259	403	66
K=12	149	13	88	219	368	101
K=13	144	18	61	246	390	79
K=14	147	15	63	244	391	78
K=15	156	6	94	213	369	100

Figure 7: entries of the confusion matrix for different values of k

We can see that the performance of PCA classifier is not as good as the K-nearest neighbor algorithm, but it still has about 90 percent accuracy.

(d) Then, we construct an LDA classifier based on developed LDA alogirthm in Homework 3 using following MATLAB code:

```
function I_LDA  = LDA_classifier(x, Q, c, k)
    norms=zeros(1,k);
    for i=1:k
        norms(i)=norm(Q*x - c{i});
    end
    [~, index]=min(norms);
    I_LDA=index;
end
```

Here we use m = 2 to find the first two projection directions. I also include a snippet of MATLAB code.

```
[V,D]= LDA(X_training, I_training, 2);
Z=cell(1,2);
Z{1}=V'*X_training_benign;
Z{2}=V'*X_training_malignant;
C=cell(1,2);
for i=1:2
    C{i}=(1/size(Z{i},2)) * sum(Z{i},2);
end
I_LDA=zeros(1,tp);
for i=1:tp
    I_LDA(i)=LDA_classifier(X_testing(:,i), V', C,2);
end
```

Then, we get the confusion matrix with C11+C22= 447 and C12+C21=22, which is relatively good with 95 percent accuracy.

(e) Here we construct an LVQ classifier and MATLAB codes for it are here:

```
function [Prototype,I_Prototype] = LVQ(X,I,k,L,Data,I_label)
[n,p]=size(X);
t=1;
N_max = 1000;
alpha_0 = 0.9;
beta = log(10)/N_max;
% initialize prototypes
col= size(Data,2);
P = randperm(col);
selected = P(1:L*k);
Prototype = Data(:,selected);
I_Prototype = I_label(selected,1);
while t< N_max
    % draw a random colume from training matrix
    i = randi([1 p]);
    x_vector = X(:,i);
    i_vector = I(1,i);
    % Find the nearest prototype vector
    match = zeros(1,L*k);
    for j = 1:L*k
      match(1,j)=(norm(Prototype(:,j)-x_vector))^2;
    end
    [~,match_t_index] = min(match);
    match_t = Prototype(:,match_t_index);
    i_match_t_index = I_Prototype(match_t_index);
    % Compute the current changing parameters
    alpha_t = alpha_0*exp(-beta*t);
    % Update prototype
    if i_match_t_index == i_vector
        Prototype(:,match_t_index) = match_t+alpha_t*(x_vector-match_t);
    else
        Prototype(:,match_t_index) = match_t-alpha_t*(x_vector-match_t);
    end
   t=t+1;
end
end

function [I_LVQ] = LVQ_classifier(X,Prototype,L,k,I_Prototype)
[n,p]=size(X);
I_LVQ=zeros(1,p);
for i=1:p
    x_vector = X(:,i);
    match = zeros(1,L*k);
    for j = 1:L*k
      match(1,j)= (norm(Prototype(:,j)-x_vector))^2;
    end
    [~,match_t_index] = min(match);
    I_LVQ(1,i) = I_Prototype(match_t_index);
end
```

The confusion matrix we get for choosing L=10 of WBC has entries C11+C22 = 440 and C12+C21

=29, which indicates that LVQ with L=10 has accuracy about 94 percent. The reason that I choose L =10 is that in the previous LDA plot we made for problem3, we also choose the first ten significant attributes.

Problem 5

For this problem, we first compute H of WBC by using NMF developed in **Homework5** for different values of k, and the plot the entries of the rows of H for k=4,6,8 with red indicating labeled 1 and blue indicating labeled 2, see Figure 8,9,and 10. (For completeness, I do all the combinations of rows for each H)

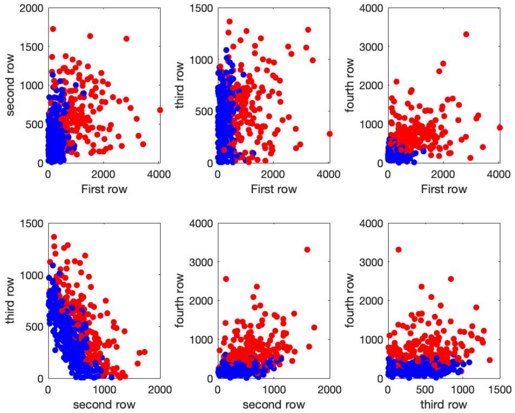


Figure 8: entries of rows of H for k =4

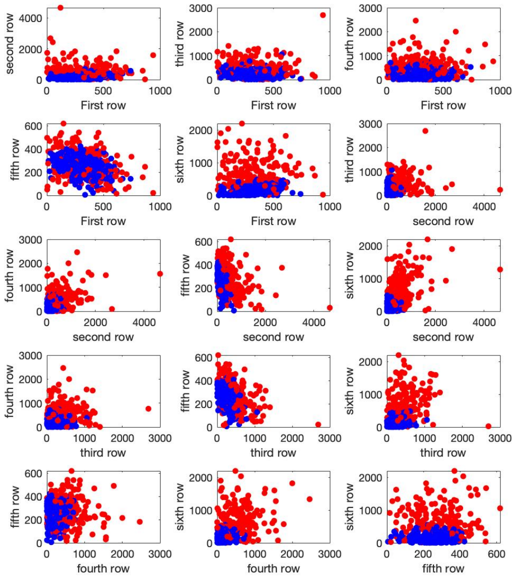


Figure 9: entries of rows of H for k =6

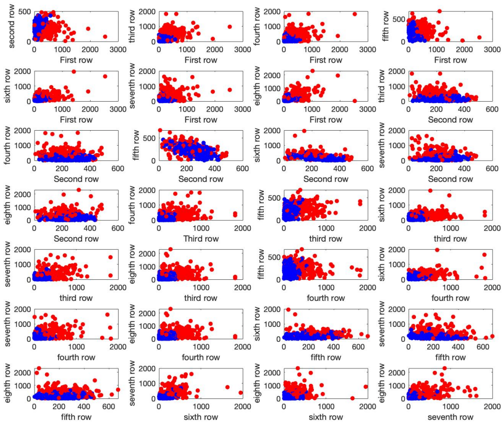


Figure 10: entries of rows of H for k =8

From above graphs, we can see that there are some separation between these two clusters but they are not completely separated. Furthermore, looking at the graphs, we can see that some feature vectors, like the first, fourth, sixth, seventh, can be associated with Benign or Malignancy.

Problem 6

Now, after collecting the Benign data in the matrix XB and the Malignant data in the matrix XM, we compute the NMF of XB and XM, respectively, with k = 4. In order to see whether NMF can be used to extract a profile for Benign or Malignant breast lesions, we plot the entries of H, see Figure 11.

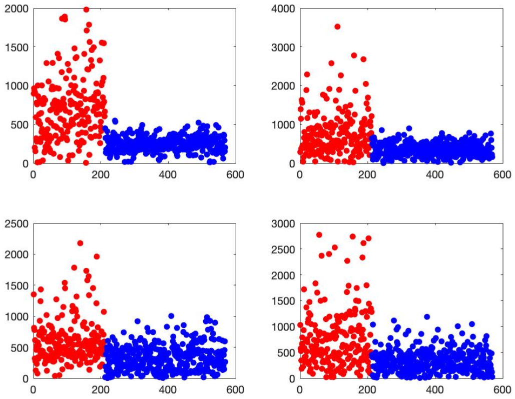


Figure 11: entries of H for k =4 withe independent NMF on XB and XM

From above, we can see that there are clear separations between those two clusters, which indicating that NMF is really a good tool to analyze the WBC dataset