Feedback sheet: Assignment

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: 02/13/2022**

ASSIGNMENT 2

Ellis Wright (enw30)

In this assignment, we will be implementing Lloyd's algorithm, also known as K-means clustering. Also, we will implement PAM, or K-medoids algorithm, an extension of k-means that allows for discrete features. Each algorithm initializes characteristic vectors in the data, then alternates iteratively updating the labels and characteristic vectors until a desired threshold is reached. Here is the code that generated the figures in this paper:

listings Here's an example of using the lstlisting environment from the listings package:

```
%given a current assignment, update the means of the clusters
function means = update_means(k,data,assignment)
    dim = size(data);
    means = zeros(dim(1),k);
    for j = 1:k
        cluster = data(:,assignment == j);
        means(:,j) = mean(cluster,2);
    end
end

%given the means and assignment, calculate the within cluster coherence
function coherence = within_cluster_coherence(data,assignment,means)
    k = length(means(1,:));
    coherence = zeros(1,k);
    for i = 1:k
        cluster = data(:,assignment==i);
        coherence(1,i) = sum(vecnorm(cluster - means(:,i)));
    end

%assign each data point to the closest mean
function assignment = update_assignment(means,data)
    p = length(data);
    assignment = 1:p;
    for i = 1:p
        [~,label] = min(vecnorm(means - data(:,i)));
        assignment(i) = label;
    end
end

%initialize the clusters
function [clusters,assignment,best_tightness] = init_clusters(data,k,n)
    p = length(data);
    best = [];
    best_tightness = Inf;
    for i = 1:n
        clusters = data(:,randsample(p,k));
        labels = update_assignment(clusters,data);
        tightness = sum(within_cluster_coherence(data,labels,clusters));
        if tightness < best_tightness
            best_tightness = tightness;
            best = clusters;
            assignment = labels;
        end
    end
end

%the k-means algorithm
function [classes,centers,list] = k_means(data,k,tau,n)
   dQ = Inf;
   list = [];

   %initialize clusters
   [characteristic_vecs,I_ell,Q] = init_clusters(data,k,n);
   counter = 1;

   while dQ > tau

        %update the means
        characteristic_vecs = update_means(k,data,I_ell);

        %update the assignment
        I_ell = update_assignment(characteristic_vecs,data);

        %calculate within and total coherence
        q1 = sum(within_cluster_coherence(data,I_ell,characteristic_vecs));
        dQ = abs(q1 - Q)/Q;
        Q = q1;
        list(counter) = dQ;
        counter = counter + 1;
   end

   classes = I_ell;
   centers = characteristic_vecs;
end
```

That was the code associated with k-means. Now here is the code for k-medoids:

```
function [labels,coherence,centers] = init_medoids(n,D,k)
```

```
    p = length(D);
    labels = [];
    coherence = Inf;
    for i = 1:n

        %pick k random points as the center
        medoids = randsample(p,k);
        d_bar = D(medoids,:);

        %calculate the within cluster coherence while assigning vectors to
        %min medoid
        [dist,pointers] = min(d_bar);
        Q = sum(dist);

        %only update if better
        if Q < coherence
            coherence = Q;
            labels = pointers;
            centers = medoids;
        end
    end

end

%dissimilarity index
function dist = medoid_dist(x,y)
    x_yes = (x == 1);
    y_yes = (y == 1);
    x_no = (x == -1);
    y_no = (y == -1);
    total_casted = sum( (x ~= 0) & (y ~= 0) );
    dist = sum((x_yes .* y_no) + (x_no .* y_yes))/total_casted;
end

function centers = update_centers(D,labels,k)

        centers = (1:k);

        %udate each of the k medoids
        for i = 1:k

            %get indices of current medoid
            indices = find(labels == i);

            %get subdistance matrix of medoids
            medoid = D(indices,indices);

            %the new center is the one with the minimum coherence
            [~,new_center] = min(sum(medoid));

            %update the center of the medoid
            centers(i) = indices(new_center);

        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


function [labels,centers,list] = k_medoids(data,k,tau,n,binary)

    %create the distance matrix
    D = create_distance_mat(data,binary);

    %initialize the medoids
    [labels,Q,centers] = init_medoids(n,D,k);

    %store the iteration number and  learning curve
    counter = 1;
    list = [];
    dQ = Inf;

    while dQ  > tau
```

```
            %update the centers
            centers = update_centers(D, labels ,k);

            %update the labels and within-medoid coherences
            medoids = D(centers ,:);
            [q, labels] = min(medoids);

            %calculate total coherence
            Q_new = sum(q);

            %calculate percent change in coherence
            dQ = abs(Q_new - Q)/Q;
            Q = Q_new;
            list(counter) = dQ;
            counter = counter + 1;
        end
end
```

After implementing the code, I tested it out on two data sets. The first data set contains a bunch of attributes about different wines and the goal is to be able to figure out which of the three cultivars made the wine. The next data set is a set containing all the votes of the house of representatives on 16 topics in 1984. The goal for this set is to predict party membership.
The wine dataset is made up of all real valued variables so both k-means and medoids can run. After running k-means, the output for the predictions was about 70% classification accuracy. The accuracy for k-medoids was 72% .The second and third wine cultivars have a lot of overlap as shown in the principal components graph below -at least in lower dimensions.

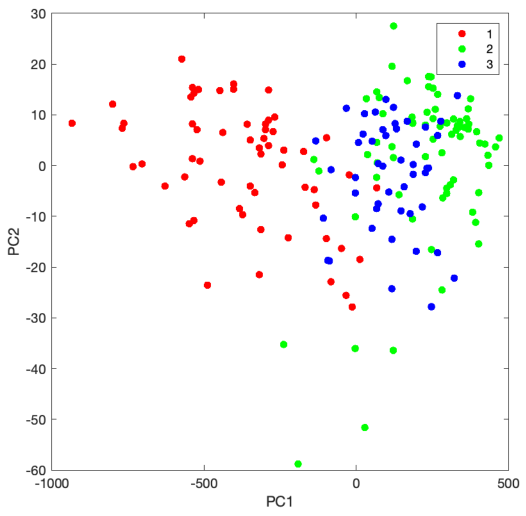

Figure 1: actual data

Here are the predictions for my implemented algorithms. As you can see from the bottom data point, the algorithms get different results but look quite similar. I am not sure why the outputs are

Figure 2: kmeans

Figure 3: kmedoids

able to cluster the principle components while the actual data does not look like it would be able to be clustered. Ultimately, with a 70% classification rate, I would say this data was difficult to classify.

The congressional dataset has the votes for, against and abstained on 16 issues in the 1984 house of representatives. We are trying to predict a binary variable with categorical variables with 3 instances each. Therefore, only the k-medoids algorithm can be run. It is hard to visualize the data so I created a confusion matrix. The k-medoid classifier resulted in:

$$\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix} = \begin{bmatrix} 162 & 75 \\ 5 & 192 \end{bmatrix}$$

I believe a negative represents a democrat and a positive represents a republican. Therefore, we can see that the classifier incorrectly classified a lot of democrats as republicans (75) but rarely classified republicans as democrats incorrectly (5). Compared to today, the house was significantly more bipartisan. Today, all you would have to do is look at one vote and there typically is no more than 15 swing voters. The representative for the first group only voted on two issues while the other representative voted no on most meausres. This demonstrates that the missing vote could be an informative measure because parties could have more power if you can get opponents to miss votes. Finding these patterns suggests gameplans for both parties.

A red flag in my code is that the algorithms seemed to have converged extremely fast taking at most 3 iterations to find the best centers.