Math 444: Project 2

Levent Batakci

February 28, 2021

The work in this project focuses on k-means and k-medoids as clustering methods. All of the mentioned implementations are coded in MATLAB and can be found in my public GitHub Repository.

Algorithm and Analysis Background

Throughout this project, we relied on two clustering algorithms to separate our data into groups. Both algorithms had a shared goal: they aimed to choose representative points $c_1, c_2, \ldots, c_k$ and a partition Π such that the data is split into k groups in a way that the overall coherence is minimized.

The overall coherence Q is defined to be the sum of the distances of the data points to their corresponding cluster representative. Both algorithms minimize Q by alternating optimization (iteratively). The algorithms stop and return the computed solution once the change in coherence is under some specified τ value. We've taken $\tau = 0.05$ for every clustering obtained in this report.

Though we will not discuss the details of either algorithm, it is important to understand the key difference between them. Specifically, the representative points chosen by kMeans need not be in the data set, whereas the representative points (medoids) chosen by kMedoids do. In certain contexts, it does not makes sense to use kMeans. For instance, if a data set only has integer values, it would not be faithful to represent clusters with non-integer points.

It is important to note that the means/medoids are initialized randomly and so the results of various runs of the algorithms can differ.

Additionally, given the true grouping of our data (into k groups), we evaluate our results with a summary matrix $C \in \mathbb{Z}^{k \times k}$ defined by

$$C_{i,j} = \text{The number of data points in group } i \text{ placed in cluster } j.$$

There are a few important features of this matrix. Firstly, the sum of the i-th row is the number of elements in group i. The sum of the j-th column is the number of elements in cluster j.

More complicated conclusions can be made by analyzing C. For instance, if one element in row i is significantly greater than the sum of the rest, then the classifier does a good job of placing group i into the same cluster.

Similarly, if the clustering does a good job, we can expect that each row will have a (significantly) largest element and that these elements will be in different columns. In this scenario, we can use this information to understand how the labeling of the clusters compares to the labeling of the true groups.

One final thing to note is that any C is as good as any version of itself with its columns permuted. This is because permuting the columns just corresponds to a renaming of the clusters.

As with most things, the interpretation of C depends heavily on the context.

Wine Cultivar Clustering

The first data set we analyzed consists of the chemical analysis of wines originating from three different cultivars. Specifically, the data is of the form

$$x^{(j)} = \begin{pmatrix} \text{Alcohol} \\ \text{Malic acid} \\ \text{ash} \\ \text{Alkalinity of ash} \\ \text{Magnesium} \\ \text{Total Phenols} \\ \text{Flavonoids} \\ \text{Nonflavonoid phenols} \\ \text{Proanthocyanins} \\ \text{Color intensity} \\ \text{Hue} \\ \text{OD280/OD315 of diluted wines} \\ \text{Proline} \end{pmatrix},$$

where the entries are concentrations/levels. The data consisted of 178 data points and was stored in a matrix $X \in \mathbb{R}^{13\times 178}$. Furthermore, the data is accompanied by an annotation matrix $I \in \mathbb{R}^{178\times 1}$ which denotes the cultivar from which each data point originated. That is, $x^{(j)}$ is data about a wine from cultivar k if and only if $I(j,1) = k$, where $1 \leq k \leq 3$ represents one of the three cultivars. Since we know that the data originate from three different cultivars, it makes the most sense to attempt to find 3 clusters within it.

Before attempting to cluster the data, we wanted to first visualize the correct grouping. Since the data is 13-dimensional, we needed to decrease its dimensionality. To do this while still remaining faithful to the features of the data, we computed the Singular Value Decomposition (SVD) and projected the data along the two first singular values. We chose not to center the data since we planned to used the euclidean norm to cluster the data. The euclidean distance between two points remains the same if both are shifted by the same amount. We made a color-coded scatter plot of the first two Principal Components (PCs) of the data. The red, green, and blue points denote data from cultivars 1, 2, and 3, respectively. This plot can be seen in **Figure 1**.

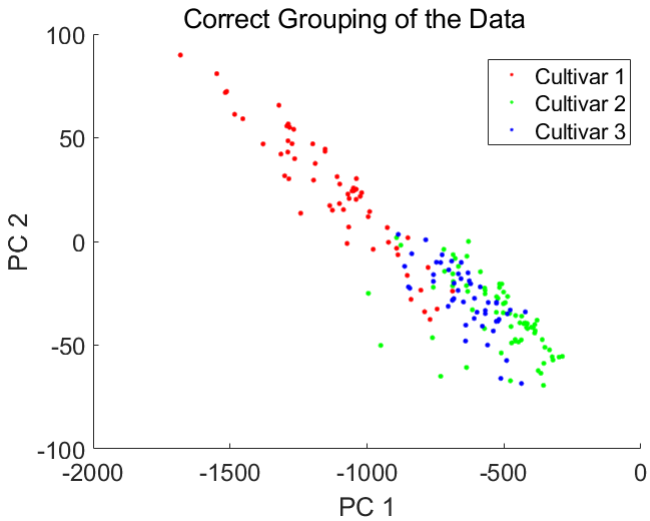

Figure 1: Grouping of the Wines into Cultivars by Given Annotation

Immediately, it's apparent that we might face issues when clustering this data. The data from cultivar 1 (shown in red) are relatively distinguishable from the others, but there is a lot of shared space between the data of cultivars 2 and 3 (green and blue, respectively). It's also important to note the massive difference of scale between the x and y axes. This is a result of the first singular value of the data being significantly larger than all of the rest.

We attempted to cluster this data via the kMedoids algorithm. The resultant clustering can be seen in **Figure 2** on the right. The medoids are boxed. given that we used the euclidean norm as a distance, the resultant clustering makes sense.

One thing to note is that the colors of the clustering given by the kMedoids algorithm are unimportant. It is clear that Cluster C (blue) matches up best to Cultivar 1 (red).

The summary matrix is $C = \begin{bmatrix} 0 & 11 & 48 \\ 50 & 19 & 2 \\ 18 & 30 & 0 \end{bmatrix}$. The clustering best identified wines from cultivar 1, which was expected. This is apparent from how 48 of the 59 wines from cultivar 1 were placed in the same cluster, and that only 2 wines not from cultivar 1 were placed in that cluster. Overall, we conclude that the cultivars of these data cannot be distinguished well by the kMedoids algorithm. It is possible that more sophisticated clustering techniques would be able to better distinguish these cultivars.

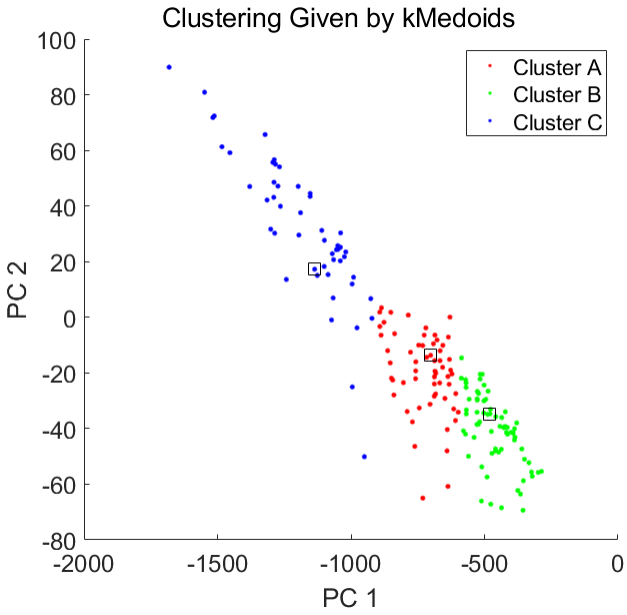

Figure 2: kMedoids Clustering of the Wine Data

Cardiac SPECT Clustering

The next data set we analyzed consists of binary features from cardiac Single Proton Emission Computed Tomography (SPECT) images. This data set is available here. The data are in the form $x^{(j)} \in \mathbb{R}^{22}$, with binary entries (0 or 1) corresponding to the presence of certain features. This data set consists of data from 187 patients and so it is stored in a matrix $X \in \mathbb{R}^{22\times 187}$. The data is accompanied by an annotation vector $I \in \mathbb{R}^{1\times 187}$ which labels each data point as normal (0) or abnormal (1). In light of this, we chose to attempt to separate the data into two clusters.

The first task we needed to accomplish was to come up with a way to compare the data points. Since the data points were binary, we decided to use a dissimilarity index. For any two data points u and v, we define the dissimilarity index as

$$d(u,v) = \frac{\text{diff}(u,v)}{22},$$

where $\text{diff}(u,v)$ is the number of entries for which u is different from v. Note that for any data points u and v, we have $0 \leq d(u,v) \leq 1$ since the data points all have 22 entries each. This dissimilarity index can easily be computed in MATLAB with

```
d = nnz(u ~= v) / 22;
```

Using this dissimilarity index, we can compute a distance matrix to store the distance between all pairs of data points. Then, using that distance matrix, we can efficiently compute the clustering according to the kMedoids algorithm. Doing so, we got the summary matrix

$$C = \begin{bmatrix} 31 & 72 \\ 45 & 39 \end{bmatrix}.$$

Immediately, it's clear that this clustering did a rather mediocre job. Only 72 of the 103 (69.9%) of the 'normal' data were placed in the same cluster(2), and 45 of the 84 (53.6%) of the 'abnormal' data were placed in the same cluster (1). Moreover, the clustering produced even worse results on subsequent runs.

It is clear that this dissimilarity index in conjunction with the kMedoids algorithm does not do a good job of clustering the data. It is likely that more robust methods would yield better clusterings, but for now we are led to conclude that the data attributes do not represent the state of the patients well.

Congressional Votes Clustering

The next data set we analyzed consists of congressional votes from 1984. The data are in the form $x^{(j)} \in \mathbb{R}^{16}$, where each entry represents the individual's vote on a certain issue. Specifically, 1 represents a vote in favor, -1 represents a vote against, and 0 represents an absence of a vote. This data set consists of 435 votes and so the data is stored in a matrix $X \in \mathbb{R}^{16\times 435}$.

Again, the data were accompanied with an annotation vector $I \in \mathbb{R}^{1\times 435}$. The annotation I separates the data into two groups - Democrats and Republicans. Labels of 0 and 1 correspond to Republicans and Democrats, respectively. Given this labeling, our analysis focused on separating the data into $k = 2$ clusters.

The data contained exactly one individual who voted on no issues. Seeing as this individual is unlikely to provide any insight, we removed them from the data. This means we ended up working with a data matrix $X' \in \mathbb{R}^{16\times 434}$.

To compare the data points, we decided to use a dissimilarity index. Specifically, we computed the dissimilarity of two voting records u and v to be

$$d(u, v) = \frac{n(u, v)}{m(u, v)},$$

where $n(u, v)$ is the number of voting differences between the issues that both u and v voted on, and $m(u, v)$ is the number of issues that both u and v voted on. One might notice that the dissimilarity index is not defined when u and v voted on none of the same issues. To address this issue, we define $d(u, v) = \frac{1}{2}$, a neutral value, for such pairs of voting records. We then have the property that $0 \leq d(u, v) \leq 1$ for any two voting records. This dissimilarity index can be computed in MATLAB with the following code:

```
if(nnz(u ~= 0 & v ~= 0) == 0)
    d = 1/2;
else
    d = nnz(u ~= v & u ~= 0 & v ~= 0) / nnz(u ~= 0 & v ~= 0);
end
```

Once we had this dissimilarity index defined, we computed a distance matrix $D \in \mathbb{R}^{434\times 434}$ of the data. The matrix D is defined by $D_{i,j} = d(x^{(i)}, x^{(j)})$, where $1 \leq i, j \leq 434$. We took advantage of the fact that D is symmetric and thus only computed the upper triangular portion of it.

Since the data are discrete voting records with integer entries, we decided that it made the most sense to use the kMedoids method to cluster this data set. Once again, we chose to summarize the clustering with the matrix C.

We computed $C = \begin{bmatrix} 7 & 160 \\ 195 & 72 \end{bmatrix}$. From this matrix, it is clear that Republicans and Democrats best correspond to Clusters 2 and 1, respectively. Note that the row sums, 167 and 267, represent the annotation group totals and the column sums, 202 and 232, represent the clustering group totals.

We see that 160 of the 167 (95.8%) Republicans were put in the same cluster, and that 195 of the 267 (73.0%) Democrats were put in the same cluster. This indicates that in general, Republicans are more partisan than democrats.

To further analyze the clustering, we looked at how well the two medoids represented the group vote. We assume from C that cluster 2 represents the Republicans and that cluster 1 represents the Democrats. Then, we computed

$$\hat{d}_{rep} = 0.1048 \text{ and } \hat{d}_{dem} = 0.2296,$$

where $\hat{d}_{rep}$ is the average distance of a Republican voting record from the medoid of cluster 2 and $\hat{d}_{rep}$ is the average distance of a Democrat voting record from the medoid of cluster 1. Immediately, we notice that $\hat{d}_{rep}$ is significantly smaller than $\hat{d}_{dem}$. This fact is consistent with the fact that a far fewer percentage of Democrats than Republicans were placed in the same cluster (73.0% vs 95.8%).

Note that these values are achieved by computing the average distance from the medoids to the annotated, *not clustered*, groups. This decision reflects that the we wanted to assess how well the chosen medoids represent the groups which we are trying to distinguish with the data.

To further investigate the clustering, we examined the medoids x_{dem} and x_{rep}. The record representing the Republicans had 6 absentee votes, while the record representing Democrats had an astounding 14 absentee votes.

The lead to a discussion of how we chose to handle absentee votes. Our dissimilarity index took into consideration only topic on which both data points had voted. Realistically, it is feasible that missing votes could also be informative. For instance, one party may be significantly more likely to vote on a given issue. We can compute this altered dissimilarity index in MATLAB identically to how we computed the dissimilarity for the Cardiac data set.

Doing so, we get the summary matrix $C = \begin{bmatrix} 14 & 153 \\ 223 & 44 \end{bmatrix}$. From this matrix, we see that 91.6% of the Republicans were put in the same cluster and 83.5% of the Democrats were put in the same cluster. Furthermore, recomputing the average distances, we get $\hat{d}_{rep} = 0.2141$ and $\hat{d}_{dem} = 0.3366$. While these values are larger than before, they are closer relatively. Furthermore, the neither medoid contained any absentee votes when dissimilarity was defined this way. It's clear then that absentee votes can in fact provide valuable insights when it comes to clustering.

Overall, this data lends itself to clustering by political party. We were able to distinguish Democrats and Republicans by voting record data with a fairly high degree of accuracy.

Iris Flower Clustering

The next data we analyzed was the Iris data set. This data sets consists of data from three different subspecies of Iris (flowers) - *Iris setosa*, *Iris virginica*, and *Iris versicolor*. We are interested to see if the data lends itself to a natural clustering which can distinguish between these three subspecies. The data is of the form

$$x^{(j)} = \begin{pmatrix} \text{petal length in cm} \\ \text{sepal width in cm} \\ \text{petal length in cm} \\ \text{petal width in cm} \end{pmatrix},$$

where $1 \leq j \leq 150$. The data is encoded as a matrix $X \in \mathbb{R}^{4\times 150}$ and is loaded from the file *IrisData.mat*. The data came alongside an annotation matrix $I \in \mathbb{R}^{1\times 150}$ denoting the subspecies to

which each data point belongs. Seeing as the data pertains to three different subspecies, we attempted to separate the data into 3 clusters.

As a first step, we chose to center the data and compute its SVD. We did this in order to faithfully reduce the dimensionality of the data so we could visualize the clusterings on 2 dimensional scatter plots. Specifically, we projected the centered data onto its first two feature vectors - so all of the following scatter plots have Principal Components (PCs) along the axes.

Before attempting to cluster the data, we made a color-coded scatter plot of the first two Principal Components (PCs) of the data. The red, green, and blue points denote the Setosa, Versicolor, and Virginica subspecies, respectively. This plot can be seen in **Figure 3**.

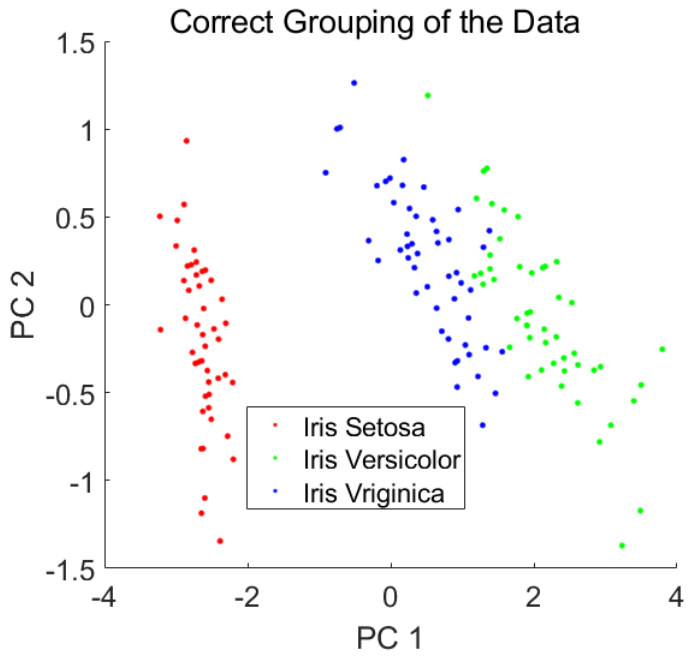

Figure 3: Grouping of the Iris into Subspecies by Given Annotation

Immediately, we see that the Setosa subspecies data (red) is most separated from the rest. Moreover, the other two subspecies aren't highly intertwined. Seeing this graph, we were fairly confident that out clustering methods would relatively successful.

Our first attempt at clustering the data was by using the kMeans algorithm. This method proved to be fairly consistent, with clustering being similar across different runs. A representative resultant clustering can be seen in **Figure 4** on the right. *The means are shown by the stars.*

To investigate this clustering more analytically, we chose to summarize it with a $k \times k = 3 \times 3$ matrix C_1. Specifically,

$$C_1 = \begin{bmatrix} 0 & 50 & 0 \\ 36 & 0 & 14 \\ 3 & 0 & 47 \end{bmatrix}.$$

From this matrix, we can see that all of the Iris Setosa data points were placed in Cluster B.

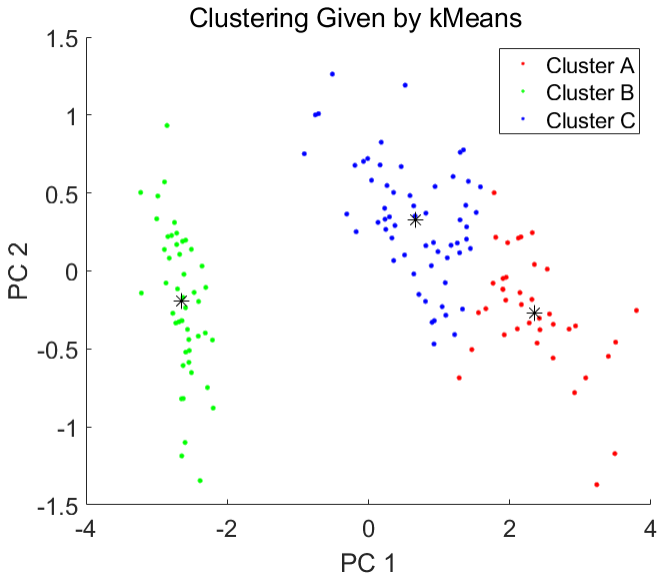

Figure 4: kMeans Clustering of Iris Data

Moreover, the only subspecies present in Cluster 2 is the Iris Setosa. This means that we were able to perfectly distinguish Iris Setosas from other subspecies in this data set.

Out of the 50 data points corresponding to the Versicolors, only 36 were placed into the same cluster (A). That being said, only 3 flowers were misidentified as being part of the Versicolor subspecies. We can be confident that new data classified as Versicolors are actually Versicolors. On the other hand, we cannot be confident that Versicolors will be correctly classified.

All but 3 of the 50 Iris Virginicas were placed into cluster C. That being said, 14 flowers were misidentified as being part of the Virginica subspecies. We can be fairly confident that this model will classify Virginica data correctly. On the other hand, we cannot be confident that a data point classified as Virginica is actually a Virginica.

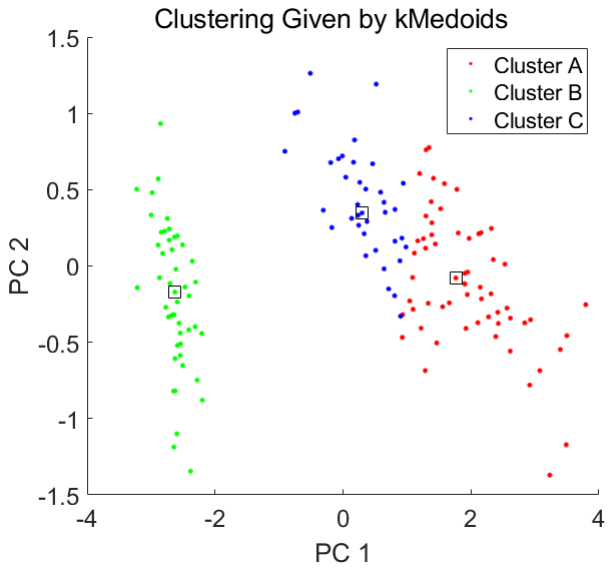

Figure 5: kMedoids Clustering of Iris Data

Our next attempt at clustering the data was by using the kMedoids algorithm. Overall, this method of clustering turned out to be significantly less consistent (**see appendix**). The best resultant clustering can be seen in **Figure 5** on the left. *The medoids are boxed in.* It's apparent that this clustering is also fairly accurate. All of the Setosas appear to be correctly grouped, and the only issues seem to be at the border of where the Versicolors and Virginicas meet.

To investigate this clustering more analytically, we chose to summarize it with a $k \times k = 3 \times 3$ matrix C_2. Specifically,

$$C_2 = \begin{bmatrix} 0 & 50 & 0 \\ 49 & 0 & 1 \\ 13 & 0 & 37 \end{bmatrix}.$$

The conclusions to be drawn from C_2 are analogous to those we drew from C_1. This is a result of the fact that C_2 is nearly identical to a version of C_1 where the first and third columns are switched then the second and third rows are switched. The column switch is irrelevant, but the row swap tells us that the conclusion we made about group 2 before now applies to group 3 and vice versa.

Overall, the Iris data set lends itself very well to a natural clustering into 3 groups. The kMeans method seems to produce good clusterings much more consistently than the kMedoids method. It is possible that by using even more robust clustering algorithms, even better clusterings are achievable.

Appendix

1 Iris kMedoids Inconsistency

The following clustering (**Figure 6**) obtained by running the kMedoids clearly show the inconsistency of the algorithm's results on this data. Other similarly bad results were produced.

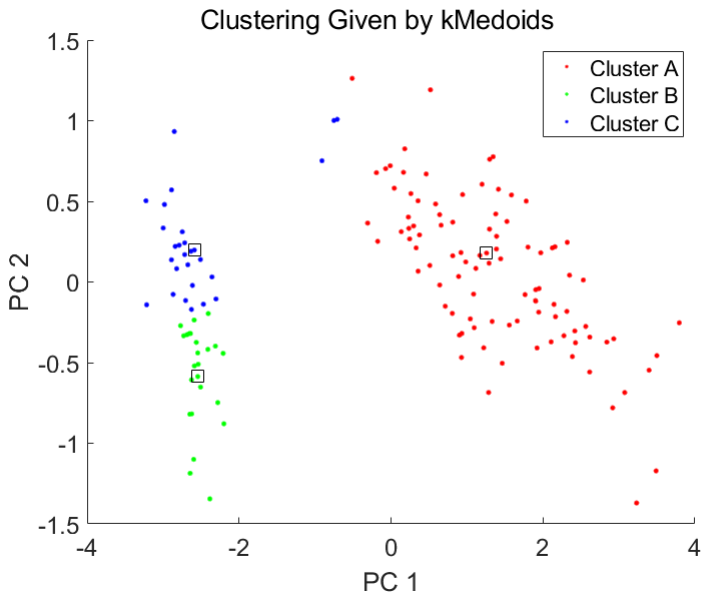

Figure 6: Bad kMedoids Clustering of Iris Data

2 Code

Many many helper functions were used to abstract the code as much as possible. Please see the public repo for details. The two main functions are provided below.

3 kMeans Code

```
%Levent Batakci
%2/22/2021
%MATH444 Assignment 2

function [I, C] = kMeans(k, D, tau, maxDepth, dist)
%KMEANS is a function that separates the data into
%  k clusters by using alternating optimization.
%This optimization is iterative, and stops when the
%  change in coherence < tau
%
%k = # of clusters
%D = data matrix
%tau = stop tolerance
%
%I = partition
%C = cluster means

    %Number of data pts
    p = size(D,2);

    %Randomize the intitial partition
```

```
    I = initIndexing(p, k);

    %Get the coherence
    C = zeros(size(D,1), k);
    for i = 1:k
        samples = D(:, I == i);
        C(:,i) = samples(:,1);
    end
    lastQ = totalCoherence(I, D, C, dist);

    %Initialize cluster means
    C = getMeans(I, D, k);

    %Repartition
    [I, Q] = repartition(D, C, dist);
    t=1;

    while abs(Q - lastQ) >= tau
        lastQ = Q;

        %Get cluster means
        C = getMeans(I, D, k);

        %Repartition
        [I, Q] = repartition(D, C, dist);

        t=t+1;
    end

    if(maxDepth >= 1)
        %Check that the code isn't giving a garbage answer
        for j = 1:k
            if(~any(I == j))
                [I, C] = kMeans(k, D, tau, maxDepth-1, dist);
            end
        end
    end

end
```

4 kMedoids Code

```
%Levent Batakci
%2/22/2021
%MATH444 Assignment 2

function [I, iC] = kMedoids_distMatrix(k, distMatrix, tau, maxDepth)
%KMEDOIDS_DISTMATRIX is a function that separates the data into
%  k clusters by using alternating optimization.
%This optimization is iterative, and stops when the
```

```
%  change in coherence < tau
%
%Unlike with kMeans, the representative vectors are
%  from of the data in this version
%k = # of clusters
%D = data matrix
%tau = stop tolerance
%
%I = partition
%C = cluster means

    %Number of data pts
    p = size(distMatrix,2);

    %Randomize the intitial partition
    I = initIndexing(p, k);

    %Randomize the intitial rep. vectors and get the Coherence
    i = 1:p;
    iC = datasample(i, k, 2, 'Replace', false);
    lastQ = totalCoherence_distMatrix(I, iC, distMatrix);

    %Initialize partition
    I = repartition_distMatrix(iC, distMatrix);

    %Choose new Medoids
    [iC, Q] = getMedoids_distMatrix(I, iC, distMatrix);
    t=1;

    while abs(Q - lastQ) >= tau
        lastQ = Q;

        %Update partition
        I = repartition_distMatrix(iC, distMatrix);
        Q = totalCoherence_distMatrix(I, iC, distMatrix);

        %Choose new Medoids
        iC = getMedoids_distMatrix(I, iC, distMatrix);
        Q = totalCoherence_distMatrix(I, iC, distMatrix);

        t=t+1;
    end

    if(maxDepth >= 1)
        %Check that the code isn't giving a garbage answer
        for j = 1:k
            if(~any(I == j))
                [I, iC] = kMedoids_distMatrix(k, distMatrix, tau, maxDepth-1);
                return;
            end
        end
    end
end
```

5 Wine Code

```
%Levent Batakci
%Analyze the data!

%Clear vars
clc
clear all

%Load WINE data
load WineData.mat
I = I';

k=3;
tau=0.05;
maxDepth=5;

%Compute the SVD
Xc = X - sum(X,2) / size(X,2) * ones(1, size(X,2));
[U, D, V] = svd(X);
Z2 = [U(:,1) U(:,2)]' * X; %First two PCs of the centered data

figure(1) %Plot the singular values
plot(1:13, diag(D), '.', 'MarkerSize', 25); %Only two svs (or just 1) matter

%Plot the annotation of the data
figure(2)
k1 = Z2(:, I == 1);
k2 = Z2(:, I == 2);
k3 = Z2(:, I == 3);

hold on
scatter(k1(1,:), k1(2,:),125,'r.')
scatter(k2(1,:), k2(2,:),125,'g.')
scatter(k3(1,:), k3(2,:),125,'b.')
legend("Cultivar 1","Cultivar 2","Cultivar 3")
xlabel("PC 1");
ylabel("PC 2");
set(gca,'FontSize', 18);
sgtitle("Correct Grouping of the Data", 'FontSize', 20)
hold off

%Cluster using kMedoids
%Plot the clustering via the first two PCs
figure(3)
distMatrix = dMatrix(X, @norm2);
[I1, iC] = kMedoids_distMatrix(k, distMatrix, tau, maxDepth);
k1 = Z2(:, I1 == 1);
```

```
k2 = Z2(:, I1 == 2);
k3 = Z2(:, I1 == 3);

hold on
scatter(k1(1,:), k1(2,:),125,'r.')
scatter(k2(1,:), k2(2,:),125,'g.')
scatter(k3(1,:), k3(2,:),125,'b.')
scatter(Z2(1,iC(1,1)), Z2(2,iC(1,1)),150,'ks')
scatter(Z2(1,iC(1,2)), Z2(2,iC(1,2)),150,'ks')
scatter(Z2(1,iC(1,3)), Z2(2,iC(1,3)),150,'ks')
legend("Cluster A","Cluster B","Cluster C");
xlabel("PC 1")
ylabel("PC 2");
set(gca,'FontSize', 18);
sgtitle("Clustering Given by kMedoids", 'FontSize', 20)
hold off

disp("Clustering Matrix for kMedoids");
M = evaluateClustering(I,I1,k)
disp(" ");

%Cluster using kMeans
%Plot the clustering via the first two PCs
figure(4)
[Im, C] = kMeans(k, X, tau, maxDepth, @norm2);
k1 = Z2(:, Im == 1);
k2 = Z2(:, Im == 2);
k3 = Z2(:, Im == 3);
hold on
scatter(k1(1,:), k1(2,:),125,'r.')
scatter(k2(1,:), k2(2,:),125,'g.')
scatter(k3(1,:), k3(2,:),125,'b.')
legend("Cluster A","Cluster B","Cluster C");
xlabel("PC 1")
ylabel("PC 2");
set(gca,'FontSize', 18);
sgtitle("Clustering Given by kMeans", 'FontSize', 20)
hold off

disp("Clustering Matrix for kMeans");
M = evaluateClustering(I,Im,k)
disp(" ");
%%%
```

6 Cardiac Code

```
%Levent Batakci
```

```
%Analyze the data!

%Clear vars
clc
clear all

k=3;
tau=0.05;
maxDepth=5;

%Load Cardiac Data
load CardiacSPECT.mat
I=I+1; %Fix the annotation

k=2;
distMatrix = dMatrix(X, @dissimilarityCardiac);
[I2, iC2] = kMedoids_distMatrix(k, distMatrix, tau, maxDepth);

X(:,iC2);

evaluateClustering(I,I2,k)

%Show the representative vectors
```

7 Votes Code

```
%Levent Batakci
%Analyze the data!

%Load Congressional Vote Data
load CongressionalVoteData.mat
I=I+1;% Fix annotation

k=3;
tau=0.05;
maxDepth=5;

%Remove the interesting fella who elected not to vote
del = all(X==0);
X(:, del) = [];
I(:, del) = [];

k=2;
%distMatrix = dMatrix(X, @dissimilarityCardiac); Use to consider absentees
distMatrix = dMatrix(X, @dissimilarityVotes);
[I2, iC2] = kMedoids_distMatrix(k, distMatrix, tau, maxDepth);

X(:,iC2);

%Summary Matrix
evaluateClustering(I,I2,k)
```

```
%Show medoids
c1 = X(:, iC2(1,1)); %dem
c2 = X(:, iC2(1,2)); %repub

%Get average distances within groups
Q = zeros(1,k);
n = [nnz(I == 1) nnz(I == 2)];
for i = 1:size(I, 2)
    label = I(1, i);
    if(label == 1) %repub
        rep = iC2(1, 2);
        Q(1, label) = Q(1, label) + distMatrix(i, rep) / n(1, label);
    else %dem
        rep = iC2(1, 1);
        Q(1, label) = Q(1, label) + distMatrix(i, rep) / n(1, label);

    end
end
Q

%Show the representative vectors
```

8 Iris Code

```
%Levent Batakci
%Analyze the data!

%Clear vars
clc
clear all
clf

%Load IRIS data
load IrisData.mat
I = [1 * ones(1, 50) 2 * ones(1, 50) 3 * ones(1, 50)];

k=3;
tau=0.05;
maxDepth=5;

%Compute the SVD
Xc = X - sum(X,2) / size(X,2) * ones(1, size(X,2));
[U, D, V] = svd(Xc);
Z2 = [U(:,1) U(:,2)]' * Xc; %First two PCs of the centered data

figure(1) %Plot the singular values
plot(1:4, diag(D), '.', 'MarkerSize', 25); %Only two svs (or just 1) matter

%Plot the annotation of the data
figure(2)
```

```
k1 = Z2(:, I == 1);
k2 = Z2(:, I == 2);
k3 = Z2(:, I == 3);

hold on
scatter(k1(1,:), k1(2,:),125,'r.')
scatter(k2(1,:), k2(2,:),125,'g.')
scatter(k3(1,:), k3(2,:),125,'b.')
legend("Iris Setosa","Iris Versicolor","Iris Vriginica");
xlabel("PC 1");
ylabel("PC 2");
set(gca,'FontSize', 18);
sgtitle("Correct Grouping of the Data", 'FontSize', 20)
hold off

%Cluster using kMedoids
%Plot the clustering via the first two PCs
figure(3)
distMatrix = dMatrix(Xc, @norm2);
[I1, iC1] = kMedoids_distMatrix(k, distMatrix, tau, maxDepth);
k1 = Z2(:, I1 == 1);
k2 = Z2(:, I1 == 2);
k3 = Z2(:, I1 == 3);

hold on
scatter(k1(1,:), k1(2,:),125,'r.')
scatter(k2(1,:), k2(2,:),125,'g.')
scatter(k3(1,:), k3(2,:),125,'b.')
scatter(Z2(1,iC1(1,1)), Z2(2,iC1(1,1)),150,'ks')
scatter(Z2(1,iC1(1,2)), Z2(2,iC1(1,2)),150,'ks')
scatter(Z2(1,iC1(1,3)), Z2(2,iC1(1,3)),150,'ks')
legend("Cluster A","Cluster B", "Cluster C");
xlabel("PC 1")
ylabel("PC 2");
set(gca,'FontSize', 18);
sgtitle("Clustering Given by kMedoids", 'FontSize', 20)


hold off

disp("Clustering Matrix for kMedoids");
M = evaluateClustering(I,I1,k)
disp(" ");

%Cluster using kMeans
%Plot the clustering via the first two PCs
figure(4)
[Im, C] = kMeans(k, Xc, tau, maxDepth, @norm2);
C2 = [U(:,1) U(:,2)]' * C;
k1 = Z2(:, Im == 1);
k2 = Z2(:, Im == 2);
k3 = Z2(:, Im == 3);
hold on
```

```
scatter(k1(1,:), k1(2,:),125,'r.')
scatter(k2(1,:), k2(2,:),125,'g.')
scatter(k3(1,:), k3(2,:),125,'b.')
scatter(C2(1,1), C2(2,1),125,'k*')
scatter(C2(1,2), C2(2,2),125,'k*')
scatter(C2(1,3), C2(2,3),125,'k*')
xlabel("PC 1")
ylabel("PC 2");
set(gca,'FontSize', 18);
sgtitle("Clustering Given by kMeans", 'FontSize', 20)
legend("Cluster A","Cluster B","Cluster C");
hold off

disp("Clustering Matrix for kMeans");
M = evaluateClustering(I,Im,k)
disp(" ");
%%%
```