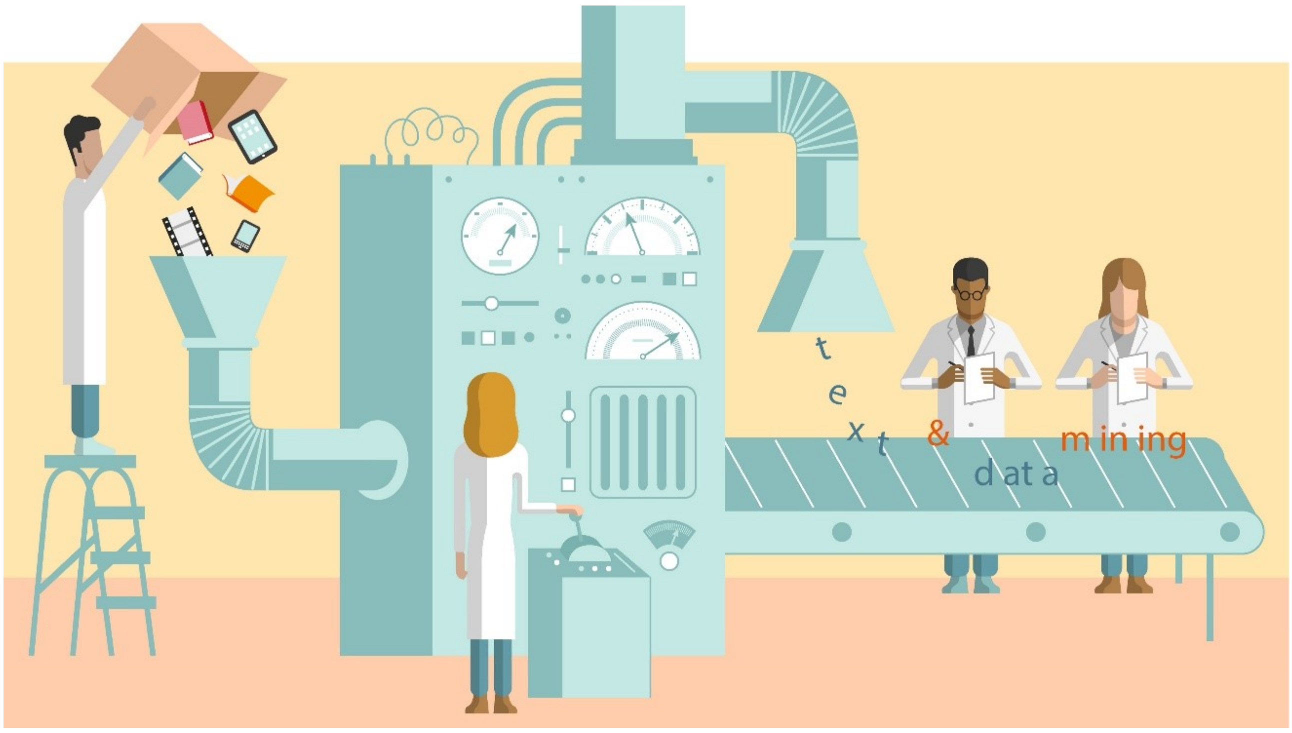
Amirkabir University of Technology

Computer engineering and IT department

Principles of Data minig

Assignment 2



Authors:

Mohammad Navid Shahsavari

Yasaman Mirmohammad

Sina Malakouti

Mohammad Hossein Goharinejad

Under Supervision of:

Prof. Ehsan Nazerfard

March 21, 2019

1 Theory

Question 1

Suppose that the data mining task is to cluster points (with .(x, y) representing location) into three clusters, where the points are:

$A_1(2,10), A_2(2,5), A_3(8,4), B_1(5,8), B_2(7,5), B_3(6,4), C_1(1,2), C_2(4,9).$

The distance function is Euclidean distance. Suppose initially we assign A1, B1, and C1 as the center of each cluster, respectively. Use the k-means algorithm to show only:

(a)The three cluster centers after the first round of execution.
(b)The final three clusters.

Question 2

Both k-means and k-medoids algorithms can perform effective clustering.
(a) Illustrate the strength and weakness of k-means in comparison with k-medoids.
(b) Illustrate the strength and weakness of these schemes in comparison with a hierarchical clustering scheme.

Question 3

Present conditions under which density-based clustering is more suitable than partitioning-based clustering and hierarchical clustering.
Explain properties of each approach , Give application examples to support your argument.

Question 4

Many partitional clustering algorithms that automatically determine the number of clusters, claim that this is an advantage. List two situations in which this is not the case.

Question 5

Use the similarity matrix in Table to perform single and complete link hierarchical clustering. Show your results by drawing a dendrogram. The dendrogram should clearly show the order in which the points are merged.
(consider similarity measure as MIN -two closest points in different clusters)

	p1	p2	p3	p4	p5
p1	1.00	0.10	0.41	0.55	0.35
p2	0.10	1.00	0.64	0.47	0.98
p3	0.41	0.64	1.00	0.44	0.85
p4	0.55	0.47	0.44	1.00	0.76
p5	0.35	0.98	0.85	0.76	1.00

Figure 1: Similarity matrix

*Question 6(Bonus Mark)

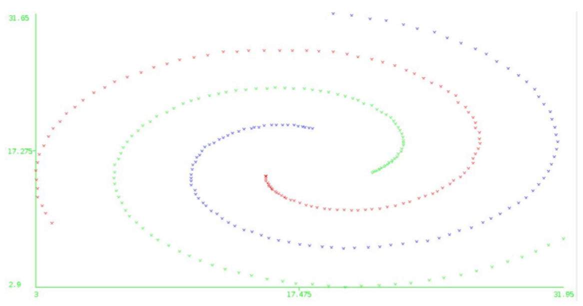
Human eyes are fast and effective at judging the quality of clustering methods for 2-D data. Can you design a data visualization method that may help humans visualize data clusters and judge the clustering quality for

3-D data?
What about for even higher-dimensional data?

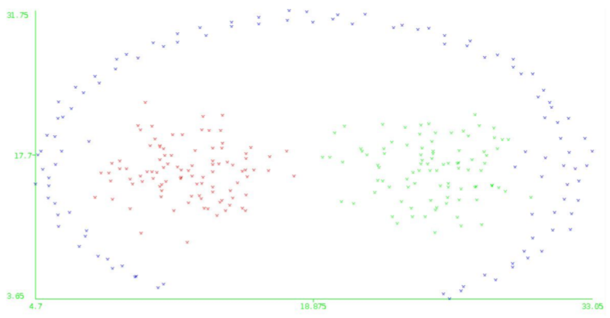
Question 7

Consider the following image showing data points belonging to three different clusters (indicated by colour). Which among the clustering algorithms will perform well in accurately clustering the given data? K-Means, DBSCAN, or both? explain why do you think an algorithm would perform well or wouldn't perform well.

A)



B)



Question 8

Answer the following problems.

A)

Explain when DBSCAN doesn't work well? And when works well? Why?

B)

Which of the following is/are true about DBSCAN clustering algorithm? Explain your Answer for each option.

1. For data points to be in a cluster, they must be in a distance threshold to a core point
2. It has strong assumptions for the distribution of data points in dataspace
3. It has substantially high time complexity of order O(n3)
4. It does not require prior knowledge of the no. of desired clusters
5. It is robust to outliers

C)

Explain advantages of DBSCAN algorithm.

D)

Explain dis-advantages of DBSCAN algorithm.

2 Implementation

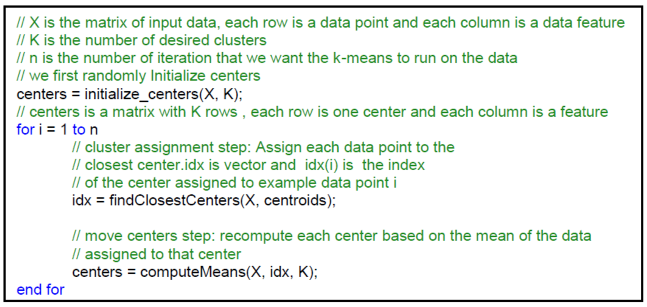
Task 1: K-Means Clustering:

1) Intro:

In this part you will implement and use the K-means clustering algorithm. First you will learn to cluster a simple 2-D dataset, next you will learn a method to evaluate the performance of clustering and finally you will learn about the restrictions of K-means by running it on a complex dataset.

2) Implementation of K-means:

As discussed in the course class, the main idea behind K-means is an iterative process that starts by guessing the initial cluster centers, and then improves this guess by repeatedly assigning data points to their closest cluster center and then recalculating the centers based on the assignment.The pseudo-code of K-means is as follows:



```
// X is the matrix of input data, each row is a data point and each column is a data feature
// K is the number of desired clusters
// n is the number of iteration that we want the k-means to run on the data
// we first randomly Initialize centers
centers = initialize_centers(X, K);
// centers is a matrix with K rows , each row is one center and each column is a feature
for i = 1 to n
        // cluster assignment step: Assign each data point to the
        // closest center.idx is vector and  idx(i) is  the index
        // of the center assigned to example data point i
        idx = findClosestCenters(X, centroids);

        // move centers step: recompute each center based on the mean of the data
        // assigned to that center
        centers = computeMeans(X, idx, K);
end for
```

Note that in order to implement the K-means you need to complete the findClosestCenters, computeMeans, initializecenters functions yourself. Also make sure that you use Euclidean distance to measure distance between data points.

3) Running K-means:

After completing the algorithm, run it on Dataset1. Set number of iteration to at least 15 and run K-means with k=2, 3, 4. After each run plot the clustered data points with each cluster having a different color like the Figure 2.

4) Evaluation:

As you know K-means is an unsupervised algorithm. As a result it may inherently lack the ability to present a deterministic and undeniable answer or solution. The predetermined number of clusters and the initial position of the cluster centers are two important factor that can greatly affect the performance of algorithm in this regard. To evaluate the Algorithm you should implement a function that does the following:

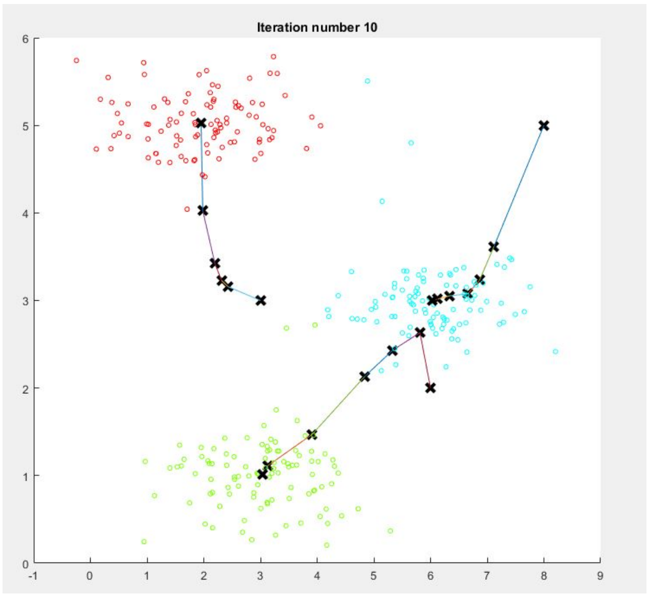


Figure 2: Result of clustering with 3 clusters

1. After the clustering is done, it computes for each cluster, the average distance between the cluster center and the data points in that cluster (this average distance is called cluster error).
2. Next, it computes the average cluster error and report it as the clustering error.

After implementing this evaluation function you should run the k-means with 0<k<15 on Dataset1 and compute the clustering error and plot these errors like the Figure 3.

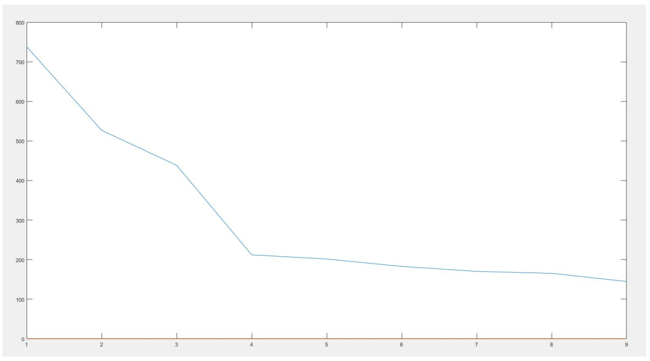


Figure 3: clustering error for different values of K for dataset in Figure 2

As you can see in the Figure 3, with more the 3 clusters the clustering error doesn't changes that much. In other word, creating more than 3 or 4 clusters doesn't improve the performance of algorithm.
The technic of finding the appropriate number of clusters by examining the decrease in the clustering error is called the elbow technic. In this exercise you use the whole data set to compute the clustering error, but in a real world scenario the clustering may take a very long time to finish so you should only chose a subset of data set for elbow technic.

5) Weaknesses and Restrictions of K-means:

In this part run K-means on Dataset2 with k=2, 3, 4, 5 and like the previous part plot the clustered data and the clustering error.
In the case that you have implemented everything correctly you will see that like Figure 4 the K-means fails

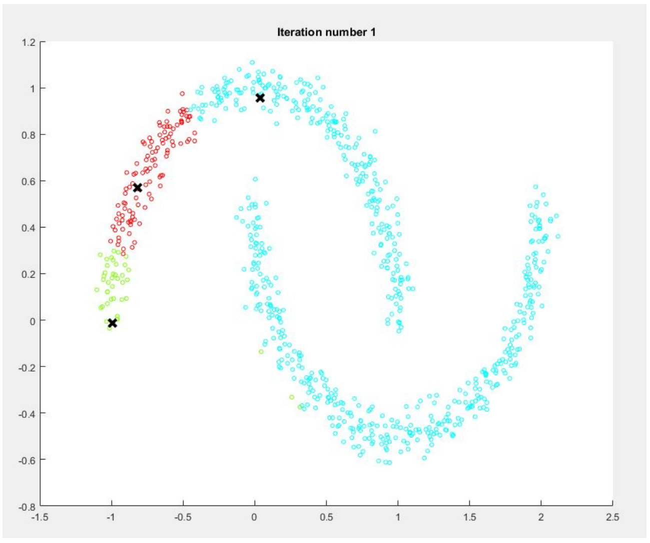


Figure 4: As a result of non-symmetric and non-homogeneous data, the K-means fails to cluster the data set

to effectively cluster the data set and the clustering error may get high. This is one of the shortcomings of K-means where it fails to operate well on data sets with non-symmetric and non-homogeneous data. In the next part you will learn about DBSCAN, an algorithm capable of handling these type of dataset better.

Task2: DBSCAN:

1) Intro:

In the previous part you got introduced to several inherent weaknesses of the K-means algorithm, namely its inability to effectively cluster data with non-symmetric clusters as well as its inability to exclude the outliners from the clustering.
In this part you start working with more sophisticated algorithm, called DBSCAN, which can somewhat handle problems mentioned above. Since DBSCAN is a little complicated you don't need to implement the algorithm in this part, and you can use available libraries (DBSCAN library in sklearn package is recommended) to complete this part.

2) Clustring data with DBSCAN:

In this section you will be using DBSCAN to cluster the data available in the Dataset2. As you already know DBSCAN requires 2 main parameters, eps and minPts.
eps is the maximum allowed distance between 2 data points in the same cluster, and minPts is the minimum number of data points to create a cluster. Using these two important parameters DBSCAN dynamically creates as many as needed clusters while discarding any clusters without enough data points and keeping the outliner data points out of any valid cluster. As a result changing the value of these to parameters changes the final number of clusters and the way the algorithm works. In this section after loading the Dataset2:

1. Run DBSCAN algorithm with values for eps and minPts, which cluster the data into one cluster, plot the data as you did in the first part and report values of eps and minPts.
2. Change the values of minPts and eps so that the data is clustered in the best possible way. Aging plot the clustered data and report eps and minPts.

3) Evaluation:

Now that you know how to use DBSCAN and how it works it's time to evaluate the algorithm with a numerical score and see how it compares to the K-means.
In this section you will use the Silhouette Score (available in sklearn package) which is a method to measure

the consistency in each cluster and similarity between data points and their cluster compared to other clusters. In this section:

1. Load Dataset1 and run K-means and DBSCAN on dataset. Set the parameters so that both algorithm create only 2 clusters, plot the data and report the silhouette score for both algorithms.
2. Repeat the previous task with as many clusters as you think is required.
3. Load Dataset2 and run K-means and DBSCAN on dataset. Set the parameters so that both algorithm create as many clusters as required, plot the data and report the silhouette score for both algorithms.

Comparing the results you will notice that while the 2 algorithms work similarly on Dataset1, DBSCAN has a slightly better performance compared to K-means because it excludes the outliners from clusters so the clusters are more coherent. But the same explanation can't be said about results from Dataset2. For results from Dataset2 you should explain the scores and plots and write them in your report.

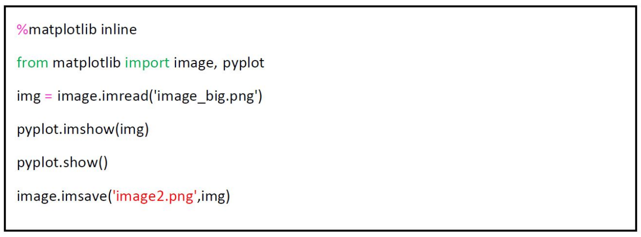
Task 3: Image Compression:

1) Intro:

Up to this point you have learnt about 2 famous and widely used clustering algorithms. In this part you will use clustering algorithm for a common real world problem.
In many RGB encodings each pixel is represented by 24 bits, 8 bits for each one of the main colors (Red, Green, Blue) ranging from 0 to 255, and therefore each pixel can have more than 16 million colors.
In this part you will use K-means algorithm to reduce the number of colors to 16 (or 256) so that the color of each pixel can be represented by only 4 bits (or 8 bits). In fact , you only need to store the RGB values of the 16 (or 256) selected colors in an array, and for each pixel in the image you now need to only store the index of the color in the array.Since reducing the number of the colors result in lower quality for the image, we use K-means to find the 16 (or 256) colors that best group pixels in the image.

2) Loading the image:

You can use the following python code to load, show and save an image in a jupyter notebook:



```
%matplotlib inline
from matplotlib import image, pyplot
img = image.imread('image_big.png')
pyplot.imshow(img)
pyplot.show()
image.imsave('image2.png',img)
```

Use the given code to load imageLarge.png. Note that by loading the image the variable img will be set to a 3-D numpy array with 800 columns and 800 rows corresponding to the location of each pixel. The third dimension of this numpy array will hold the values of Red, Green, and Blue of the color.
In order to use the K-means Algorithm, that you've already implemented, on the image, first you need to reshape this 3-D matrix into a 2-D one in which every row represents a pixel.

3) Compressing the Image:

In this section do the following steps:

1. Set K equal to 16 and Run your K-means algorithm on the Image data.
2. Replace the RGB value of each pixel with the RGB value of the center of its cluster.
3. Set K equal to 256 and repeat the previous steps.
4. Show the result images alongside the original image(like Figure 5) and compare the result of compression.

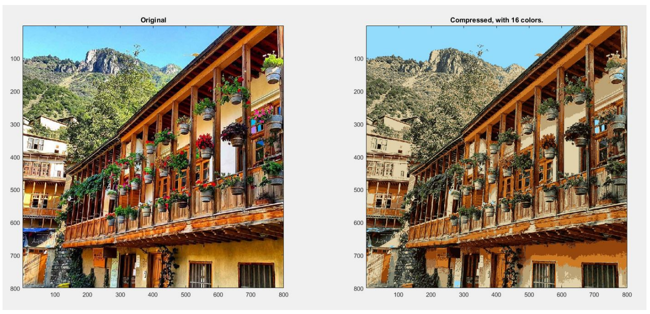
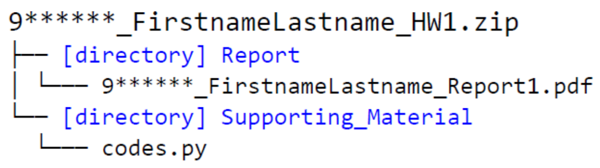


Figure 5: compressing the image by reducing the number of colors from 16 million to 16

The original image has 800x800 pixels and with 24 bits per pixel the image requires at least 800x800x24 bits to be stored. By reducing the number of colors to 16 and 256 the image only requires about 800x800x4 and 800x800x8 bits respectively to be stored.
By this point hopefully you have learnt the intuition behind the clustering and grasped a hands-on knowledge on how to use clustering for real world challenges.

3 Caution!!!

•The codes for each part must be in a different file and each file must be named after the part that it covers. So for this assignment,you need to upload 3 files named kmeas.ipynb/.py, dbscan.ipynb/.py and imagecompression.ipynb/py.
put them in the "Supporting Material" directory.
•Report is an important part of your grade. So write it completely and explain your analysis. Your report is only accepted in 'pdf' format. Put it in "report" folder. (There is no force on the language of the report)
•Your codes should be written in python. Put them in "supporting material" folder.
•For the Image compression part, you can use either the imageLarge.png or imageSmall.png. imageSmall.png has only 200x200 pixels so if your system isn't powerful you can still finish this part.
•For image compression part you must use the k-means that you have implemented yourself in the first part.
•Since your implementation of K-means and image compression will be tested by TAs on different data sets with different sizes, you need to make sure that your implementation is not dependent on the size of input data. Also try to use vectorized operation wherever you can to achieve higher performance and a slight bonus to your score.
•All the plots and images that you generate must be presented both in your pdf report and in the python notebook when TAs run your code.
•Deadline of the Theory part is on "18th of farvardin". Implentation part deadline would be extended til "31 of farvardin" you will lose 10 percent of your grade after that on each day of delay.
•Put all your folders and files like the sample format in a "zip" file and upload it on moodle(https://ceit.aut.ac.ir/courses/)
•Please upload your homework in this format:



```
9******_FirstnameLastname_HW1.zip
├── [directory] Report
│  └─── 9******_FirstnameLastname_Report1.pdf
└── [directory] Supporting_Material
   └─── codes.py
```