

CENG 3521 DATA MINING ASSIGNMENT 3

Gizem PESEN
pesengizem@gmail.com

Tuesday 29th December, 2020

1 Contents

- Declaration of Honor Code
- Clustering Task
- Visualization of steps in K-means
- Visualization of convergence of cluster centroids

2 Declaration of Honor Code

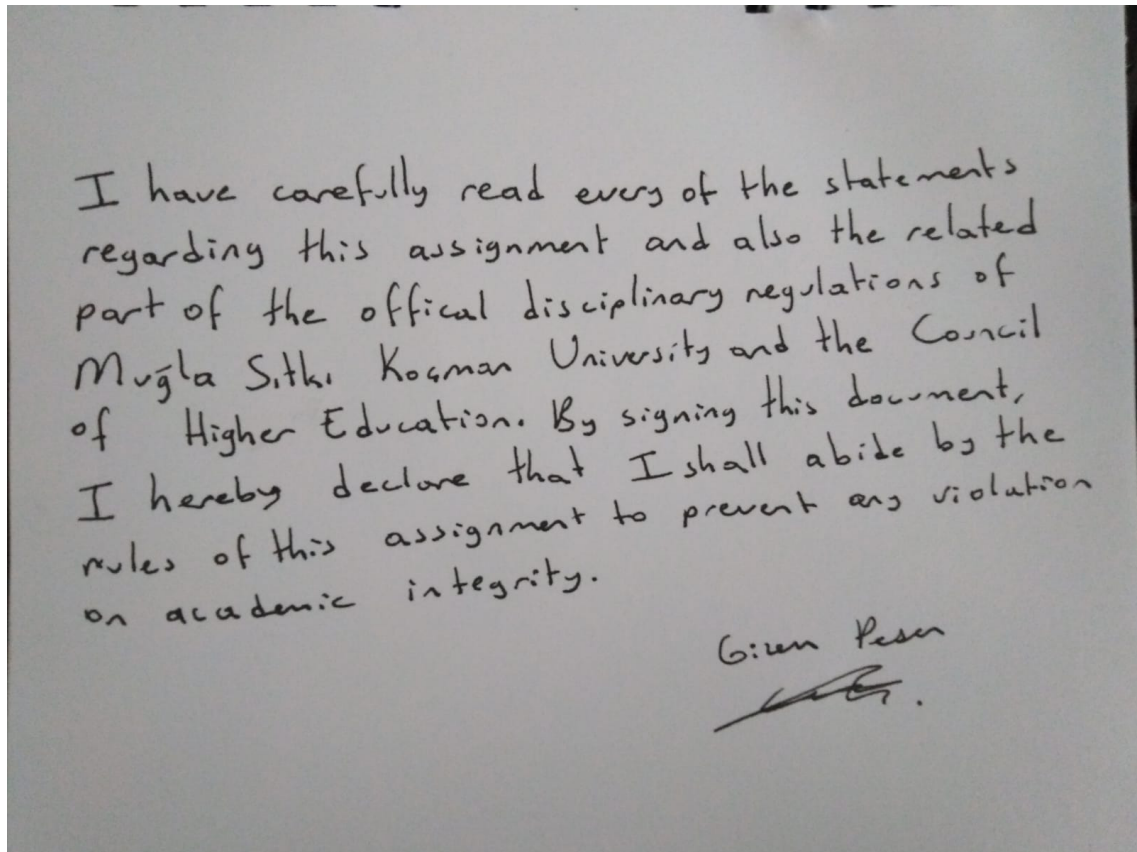
Student ID :170709050

Name Surname:Gizem Pesen

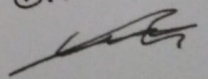
In the course of Data Mining (CENG 3521), I take academic integrity very seriously and ask you to do as well. That's why, this page is dedicated to some clear statements that defines the policies of this assignment, and hence, will be in force. Before reading this assignment booklet, please first read the following rules to avoid any possible violation on academic integrity.

- This assignment must be done individually unless stated otherwise.
- You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, you cannot copy code (in whole or in part) of someone else, cannot share your code (in whole or in part) with someone else either.
- The previous rule also holds for the material found on the web as everything on the web has been written by someone else. • You must not look at solution sets o
- You must not look at solution sets or program code from other years.
- You cannot share or leave your code (in whole or in part) in publicly accessible areas.

- You have to be prepared to explain the idea behind the solution of this assignment you submit.
- Finally, you must make a copy of your solution of this assignment and keep it until the end of this semester.



I have carefully read every of the statements regarding this assignment and also the related part of the official disciplinary regulations of Mugla Sıtkı Koşman University and the Council of Higher Education. By signing this document, I hereby declare that I shall abide by the rules of this assignment to prevent any violation on academic integrity.

Given Person


3 Clustering Task

Listing 1: Imports

```
# import statements
from sklearn.datasets import make_blobs
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

3.0.1 Visualization of steps in K-means

1. Load a ready-to-use dataset (D), or generate on your own. Be aware that there are more than five classes where data objects belong to.

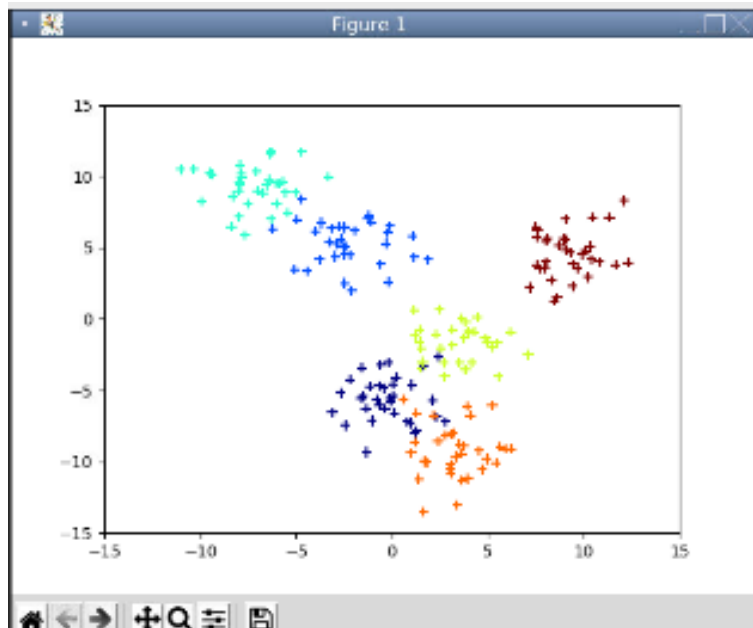
Listing 2: D

```
# create blobs
data = make_blobs(n_samples=200, n_features=6, centers=6, cluster_std=1.6,
                  random_state=50)
```

2. Show data objects with its original class distribution in a plot window. Then, save it as 'OriginalData.pdf'.

Listing 3: OriginalData

```
# create scatter plot
plt.scatter(data[0][:,0], data[0][:,1], c=data[1],
            cmap='jet',marker="+",label="Original Data")
plt.xlim(-15,15)
plt.ylim(-15,15)
plt.show()
```



- As for the demonstration of update of the cluster centroids as well as the assign of data objects to these clusters. This demonstration should include first four iterations by generating 2x2-axis figure as shown in Figure 1. Note that, the number of cluster to be set for K-means algorithm must be equal to the number of distinct classes in D.

```

First iteration points:
[[-0.24144 -5.57172525 -4.64929053 -1.96749214 -2.59156366 9.63709409]
 [-2.25162928 5.33188653 5.37634338 -3.87613592 -3.10827445 -3.21924393]
 [ 9.24674296 4.54007938 4.98403415 -0.60783229 8.60415226 -2.14022871]
 [ 3.24966127 -9.19249899 -3.55495134 -8.67603053 -6.06392195 8.4435154 ]
 [-7.20285612 9.25731267 8.73184568 1.06355484 -3.93745136 7.1829784 ]
 [ 3.53113018 -1.72379987 0.55969039 0.57738489 7.84001987 1.74011048]]

Second iteration points:
[[-0.24144 -5.57172525 -4.64929053 -1.96749214 -2.59156366 9.63709409]
 [ 3.53113018 -1.72379987 0.55969039 0.57738489 7.84001987 1.74011048]
 [-2.25162928 5.33188653 5.37634338 -3.87613592 -3.10827445 -3.21924393]
 [-7.20285612 9.25731267 8.73184568 1.06355484 -3.93745136 7.1829784 ]
 [ 9.24674296 4.54007938 4.98403415 -0.60783229 8.60415226 -2.14022871]
 [ 3.24966127 -9.19249899 -3.55495134 -8.67603053 -6.06392195 8.4435154 ]]

Third iteration points:
[[ 3.24966127 -9.19249899 -3.55495134 -8.67603053 -6.06392195 8.4435154 ]
 [ 9.24674296 4.54007938 4.98403415 -0.60783229 8.60415226 -2.14022871]
 [-7.20285612 9.25731267 8.73184568 1.06355484 -3.93745136 7.1829784 ]
 [-2.25162928 5.33188653 5.37634338 -3.87613592 -3.10827445 -3.21924393]
 [-0.24144 -5.57172525 -4.64929053 -1.96749214 -2.59156366 9.63709409]
 [ 3.53113018 -1.72379987 0.55969039 0.57738489 7.84001987 1.74011048]]

Forth iteration points:
[[ 3.24966127 -9.19249899 -3.55495134 -8.67603053 -6.06392195 8.4435154 ]
 [ 9.24674296 4.54007938 4.98403415 -0.60783229 8.60415226 -2.14022871]
 [-7.20285612 9.25731267 8.73184568 1.06355484 -3.93745136 7.1829784 ]
 [-2.25162928 5.33188653 5.37634338 -3.87613592 -3.10827445 -3.21924393]
 [-0.24144 -5.57172525 -4.64929053 -1.96749214 -2.59156366 9.63709409]
 [ 3.53113018 -1.72379987 0.55969039 0.57738489 7.84001987 1.74011048]]

```

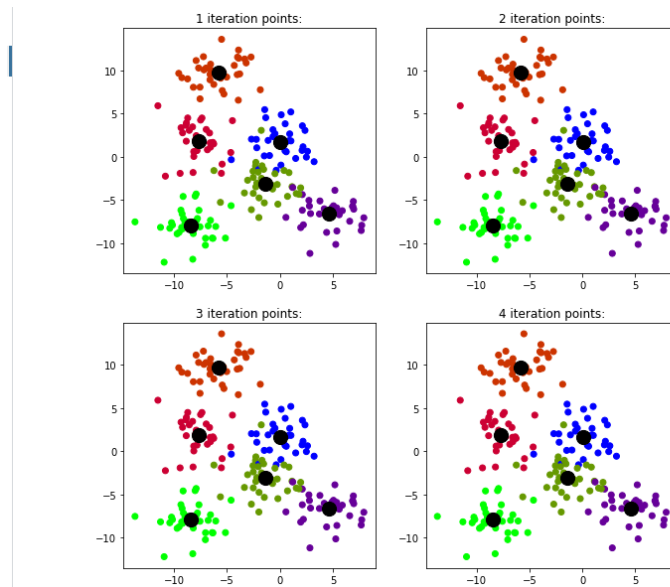
Listing 4: Iteration Points

```

print("First iteration points:")
kmeans = KMeans(n_clusters=6,random_state=0,max_iter=1)
kmeans.fit(data[0])
print(kmeans.cluster_centers_)
print("Second iteration points:")
kmeans = KMeans(n_clusters=6,random_state=0,max_iter=2)
kmeans.fit(data[0])
print(kmeans.cluster_centers_)
print("Third iteration points:")
kmeans = KMeans(n_clusters=6,random_state=0,max_iter=3)
kmeans.fit(data[0])
print(kmeans.cluster_centers_)
print("Forth iteration points:")
kmeans = KMeans(n_clusters=6,random_state=0,max_iter=4)
kmeans.fit(data[0])
print(kmeans.cluster_centers_)

```

4. Save this figure as 'KmeansDemonstration.pdf' and close3 figure window.



3.0.2 Visualization of convergence of cluster centroids

1. Load a ready-to-use dataset (D), or generate on your own. Be aware that there are more than five classes where data objects belong to.

Listing 5: D

```
data = make_blobs(n_samples=200, n_features=8, centers=6,
                  cluster_std=1.8, random_state=101)
```

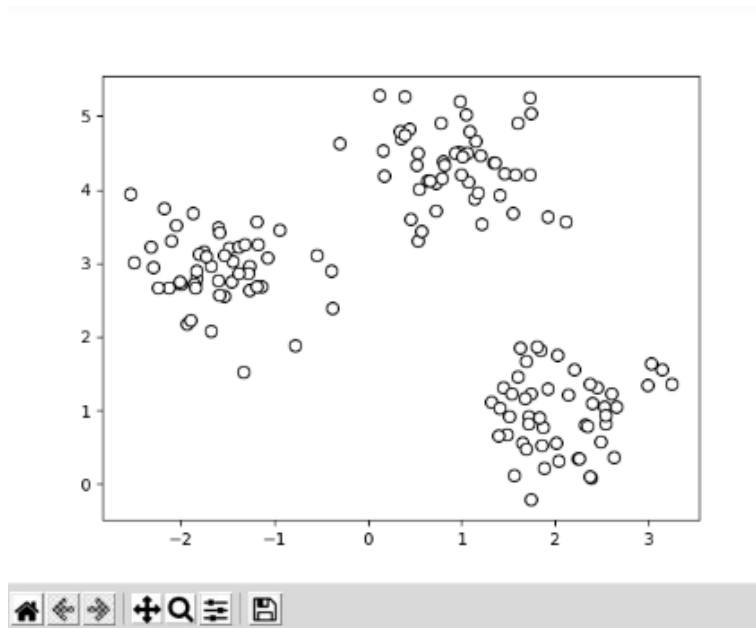
2. Apply K-means clustering algorithm with a cluster size equal to the number of classes in D.

Listing 6: K-means

```
km = KMeans(
    n_clusters=3, init='random',
    n_init=10, max_iter=300,
    tol=1e-04, random_state=0
)
y_km = km.fit_predict(X)
```

3. Generate 1×2-axis figure and show data objects with its original class distribution , show convergence of clusters that K-means obtained in each iteration . Be aware that cluster centroids at each iteration are shown by square markers whereas convergence between every centroid is indicated by a solid black line.

Original data:



Convergence of centroids:

