Name: Sidharth Banerjee
ID: 1001622703

CSE 4309: Introduction to Machine Learning

Assignment 6

Task 1

em_cluster('point_set1.txt', 2, 5)

```
After iteration 1:
weight 1 = 0.4333, mean 1 = (293.0000, 282.0000)
weight 2 = 0.5667, mean 2 = (327.0000, 320.0000)

After iteration 2:
weight 1 = 0.4890, mean 1 = (290.0000, 302.0000)
weight 2 = 0.5110, mean 2 = (334.0000, 305.0000)

After iteration 3:
weight 1 = 0.5106, mean 1 = (293.0000, 303.0000)
weight 2 = 0.4894, mean 2 = (332.0000, 304.0000)

After iteration 4:
weight 1 = 0.5266, mean 1 = (294.0000, 304.0000)
weight 2 = 0.4734, mean 2 = (332.0000, 303.0000)

After final iteration:
weight 1 = 0.5400, mean 1 = (294.0000, 305.0000)
Sigma 1 row 1 = 22355.0000, -1143.0000
Sigma 1 row 2 = -1143.0000, 31728.0000
weight 2 = 0.4600, mean 2 = (334.0000, 302.0000)
Sigma 2 row 1 = 26732.0000, -3014.0000
Sigma 2 row 2 = -3014.0000, 32086.0000
```

em_cluster('point_set1.txt', 3, 5)

```
After iteration 1:
weight 1 = 0.2667, mean 1 = (295.0000, 352.0000)
weight 2 = 0.2667, mean 2 = (206.0000, 310.0000)
weight 3 = 0.4667, mean 3 = (382.0000, 272.0000)

After iteration 2:
weight 1 = 0.2107, mean 1 = (274.0000, 338.0000)
weight 2 = 0.2128, mean 2 = (158.0000, 276.0000)
weight 3 = 0.5766, mean 3 = (383.0000, 301.0000)

After iteration 3:
weight 1 = 0.2605, mean 1 = (255.0000, 340.0000)
weight 2 = 0.2226, mean 2 = (168.0000, 284.0000)
weight 3 = 0.5170, mean 3 = (403.0000, 293.0000)

After iteration 4:
```

```
weight 1 = 0.3480, mean 1 = (249.0000, 344.0000)
weight 2 = 0.2139, mean 2 = (166.0000, 280.0000)
weight 3 = 0.4381, mean 3 = (434.0000, 283.0000)

After final iteration:
weight 1 = 0.4180, mean 1 = (245.0000, 344.0000)
Sigma 1 row 1 = 749.0000, 320.0000
Sigma 1 row 2 = 320.0000, 13688.0000
weight 2 = 0.1949, mean 2 = (158.0000, 270.0000)
Sigma 2 row 1 = 6143.0000, 3857.0000
Sigma 2 row 2 = 3857.0000, 43767.0000
weight 3 = 0.3871, mean 3 = (462.0000, 277.0000)
Sigma 3 row 1 = 20725.0000, -3362.0000
Sigma 3 row 2 = -3362.0000, 42581.0000
```

Task 2

This cannot be the final solution after k-means clustering. The mean for the blue cluster lies roughly in between the square. The distance from the blue cluster mean to the left and right corner blue dots is more than the distance from the mean of the blue cluster to the red dot. So, if the left and right corner blue dots have been assigned the blue cluster, then the red dot cannot be assigned the red cluster because it is closer to the mean of the blue cluster than the corner two dots.

Task 3

Part a: K-means can give different solutions with same k and on same dataset when run multiple times. This is because K-means randomly assigns clusters, and this leads to a locally optimal solution. This effect is highlighted when the data points are very random.

Example: a python script to show the execution of the same program twice.

Execution 1:

```
In [24]: import numpy as np
         import random
         from sklearn.cluster import KMeans

In [25]: X = np.array([[random.random(), random.random()],
                       [random.random(), random.random()],
                       [random.random(), random.random()],
                       [random.random(), random.random()],
                       [random.random(), random.random()],
                       [random.random(), random.random()]])

In [26]: kmeans = KMeans(n_clusters=2, random_state=0).fit(X)

In [27]: kmeans.labels_

Out[27]: array([0, 1, 1, 0, 0, 0])
```

Execution 2:

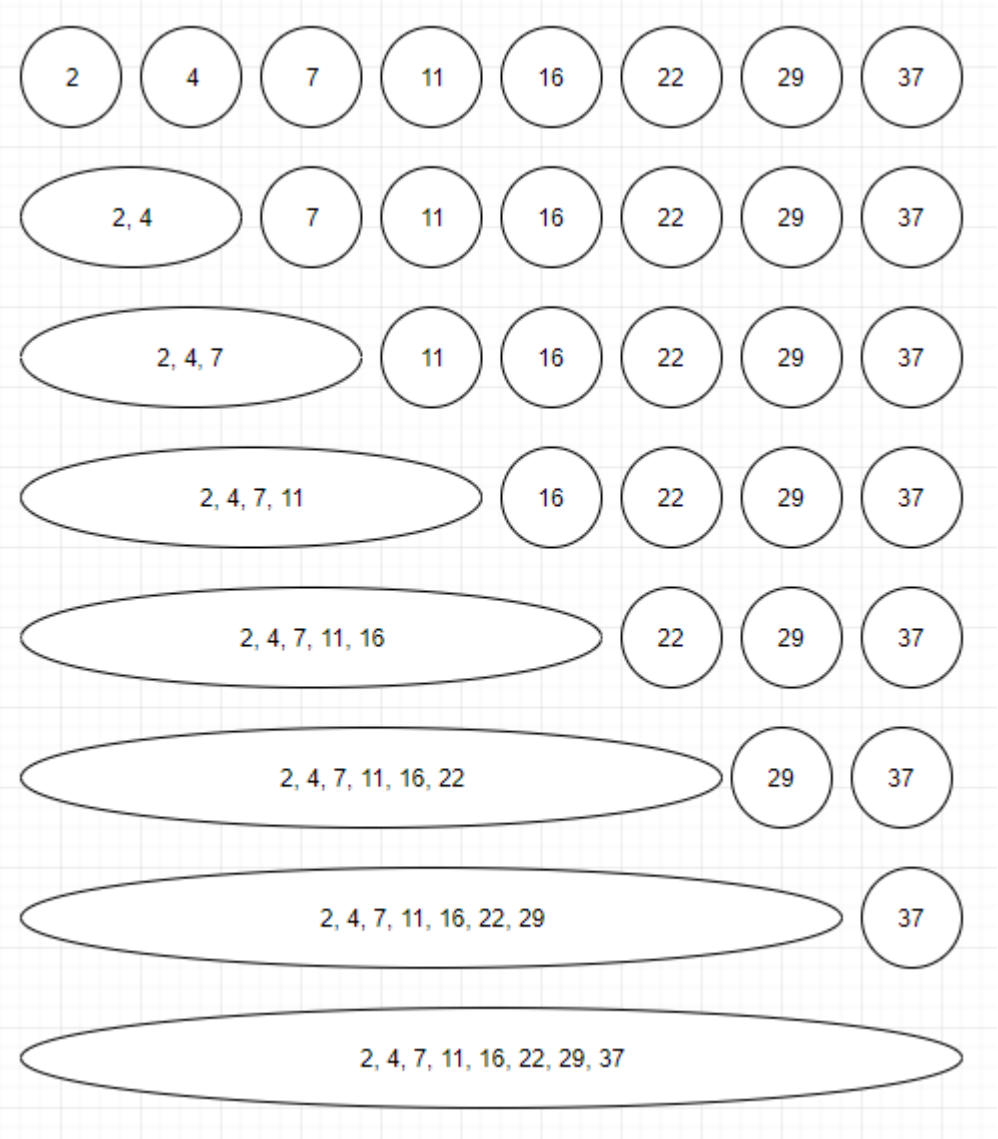
```
In [28]: import numpy as np
         import random
         from sklearn.cluster import KMeans

In [29]: X = np.array([[random.random(), random.random()],
                       [random.random(), random.random()],
                       [random.random(), random.random()],
                       [random.random(), random.random()],
                       [random.random(), random.random()],
                       [random.random(), random.random()]])

In [30]: kmeans = KMeans(n_clusters=2, random_state=0).fit(X)

In [31]: kmeans.labels_

Out[31]: array([0, 0, 1, 0, 0, 1])
```

In the first example, the clusters assigned to the 6 objects are different from the clusters assigned to the 6 objects in the second example.

Part b: Agglomerative clustering with the d_{min} will always produce the same results. This is because when we cluster objects together, we look at the absolute distance between two objects. If the dataset remains the same, then the clustering will be same for every iteration since the distance between the objects does not change.

Task 4

Part a: using d_{min}



Part b: using d_{max}: