HOMEWORK 2

1. K-MEANS CLUSTERING

ID	x	Y
1	2	10
2	2	5
3	8	4
4	5	8
5	7	5
6	6	4
7	1	2
8	4	9

a. Why is k-means clustering sub-optimal?
Ans: K-means algorithm doesn't guarantee to converge at global minimum. Several times, it just converges to any of the critical points i.e. local minima. Also, depending upon **number of clusters (k), densities, non-globular shapes, and outliers**; k-means isn't the optimal algorithm so far.

b. The k-means++ algorithm is a variant of k-means which chooses the initial centroids as follows.
The exact algorithm is as follows:

i. Choose one centroid at random from among the data points.
ii. For each data point x, compute D(x), the distance between x and the nearest centroid that has already been chosen.
iii. Choose one new data point at random as a new centroid, using a weighted probability distribution where a point x is chosen with probability proportional to $D(x)^2$.
iv. Repeat Steps 2 and 3 until k centroids have been chosen.
v. Now that the initial centroids have been chosen, proceed using standard k-means algorithm

Explain why this method will not only speed up the convergence of the k-means algorithm but also guarantee the quality of the final clustering results.
Ans:

- **Speeds up the convergence:** Random initialization of centroids may take two points from the same cluster as centroids. It will then take extra time to separate these clusters. As k-means++ algorithm starts with the data points using probability proportional $D(x)^2$, initial cluster centers, which are given as an input to k-means, are nicely spread out.
- **Quality of the result:** If the input points are well separated, it shows very promising results. If the points are not nicely separated, then k-means++ is still $O(\log k)$ competitive with the best possible clustering.

c. Based on the data points in Table 1, cluster the points (with (x, y) representing location) using k-means algorithm. The distance function is Euclidean distance. Suppose initial centroids are points with IDs **1, 4 and 7**. Report:

   i. The three cluster centers after the first iteration of execution.

   ii. The final three clusters.

Ans: Below are the steps of K-Means algorithm for points in table 1

Initial assignment of clusters:
C1 = (2, 10), C2= (5, 8), C3 = (1, 2)

Each data point will be assigned to the closest cluster centroid. Euclidean distance is used as the distance function.
E.g. For point 1 (2, 10):
Distance between P1 and C1 = $\sqrt{(2-2)^2+(10-10)^2} = 0$
Distance between P1 and C2 = $\sqrt{(2-5)^2+(10-8)^2} = 3.605$
Distance between P1 and C3 = $\sqrt{(2-1)^2+(10-2)^2} = 8.062$

C1 is the closest cluster centroid to P1, thus assigning C1 to P1.
Similarly, for all points, the closest centroids are assigned and are given below in the table.

First Iteration:

Point	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
d (Point, C1)	**0**	5	8.485	3.605	7.071	7.211	8.062	2.236
d (Point, C2)	3.605	4.242	**5**	**0**	**3.605**	**4.123**	7.211	**1.414**
d (Point, C3)	8.062	**3.162**	7.28	7.211	6.708	5.385	**0**	7.615
Nearest Cluster	**C1**	**C3**	**C2**	**C2**	**C2**	**C2**	**C3**	**C2**

C1 $=$ centroid of points within cluster 1 $=$ centroid of 1 $= (2, 10)$
C2 $=$ centroid of points within cluster 2 $=$ centroid of 3, 4, 5, 6, 8
$= (8+5+7+6+4)/5, (4+8+5+4+9)/5 = (6, 6)$
C3 $=$ centroid of points within cluster 3 $=$ centroid of 2, 7
$= (2+1)/2, (5+2)/2 = (1.5, 3.5)$

After first iteration,
C1 = (2, 10), C2 = (6, 6), C3 = (1.5, 3.5)

Second Iteration:

Point	1	2	3	4	5	6	7	8
d (Point, C1)	**0**	5	8.485	3.605	7.071	7.211	8.062	**2.236**
d (Point, C2)	5.657	4.123	**2.828**	**2.236**	**1.414**	**2**	6.403	3.605
d (Point, C3)	6.519	**1.581**	6.519	5.7	5.7	4.527	**1.581**	6.041
Nearest Cluster	**C1**	**C3**	**C2**	**C2**	**C2**	**C2**	**C3**	**C1**

C1 = centroid of points within cluster 1 = centroid of 1, 8
= (2 + 4)/2, (10 + 9)/2 = (3, 9.5)
C2 = centroid of points within cluster 2 = centroid of 3, 4, 5, 6
= (8 + 5 + 7 + 6)/4, (4 + 8 + 5 + 4)/4 = (6.5, 5.25)
C3 = centroid of points within cluster 3 = centroid of 2, 7
= (2 + 1)/2, (5 + 2)/2 = (1.5, 3.5)

After second iteration,
C1 = (3, 9.5), C2 = (6.5, 5.25), C3 = (1.5, 3.5)

Third Iteration:

Point	1	2	3	4	5	6	7	8
d (Point, C1)	**1.118**	4.609	7.433	**2.5**	6.02	6.264	7.762	**1.118**
d (Point, C2)	6.543	4.506	**1.952**	3.132	**0.559**	**1.346**	6.388	4.506
d (Point, C3)	6.519	**1.581**	6.519	5.7	5.7	4.527	**1.581**	6.041
Nearest Cluster	**C1**	**C3**	**C2**	**C1**	**C2**	**C2**	**C3**	**C1**

C1 = centroid of points within cluster 1 = centroid of 1, 4, 8
= (2 + 5 + 4)/3, (10 + 8 + 9)/3 = (3.667, 9)
C2 = centroid of points within cluster 2 = centroid of 3, 5, 6
= (8 + 7 + 6)/3, (4 + 5 + 4)/3 = (7, 4.333)
C3 = centroid of points within cluster 3 = centroid of 2, 7
= (2 + 1)/2, (5 + 2)/2 = (1.5, 3.5)

After third iteration, the final cluster centers are:
C1 = (3.667, 9), C2 = (7, 4.333), C3 = (1.5, 3.5)

Fourth Iteration:

Point	1	2	3	4	5	6	7	8
d (Point, C1)	**1.74**	4.798	7.001	**2.006**	5.599	5.974	7.96	**0.6**
d (Point, C2)	7.557	5.044	**1.053**	4.176	**0.667**	**1.053**	6.437	5.548
d (Point, C3)	6.519	**1.581**	6.519	5.7	5.7	4.527	**1.581**	6.041
Nearest Cluster	**C1**	**C3**	**C2**	**C1**	**C2**	**C2**	**C3**	**C1**

C1 = centroid of points within cluster 1 = centroid of 1, 4, 8
= (2 + 5 + 4)/3, (10 + 8 + 9)/3 = (3.667, 9)
C2 = centroid of points within cluster 2 = centroid of 3, 5, 6
= (8 + 7 + 6)/3, (4 + 5 + 4)/3 = (7, 4.333)
C3 = centroid of points within cluster 3 = centroid of 2, 7
= (2 + 1)/2, (5 + 2)/2 = (1.5, 3.5)

After fourth iteration, the final cluster centers are:
C1 = (3.667, 9), C2 = (7, 4.333), C3 = (1.5, 3.5)

The algorithm converges here as no cluster centroid have changed.

i. The three cluster centers after the first iteration of execution:
Ans: C1 = (2, 10), C2 = (6, 6), C3 = (1.5, 3.5)

ii. The final three clusters:
Ans: C1 = (3.667, 9), C2 = (7, 4.333), C3 = (1.5, 3.5)

2. HIERARCHICAL CLUSTERING

ID	x	Y
1	1	1
2	2	1
3	2	3
4	3	4
5	5	3

a. Write down the algorithm for basic agglomerative hierarchical clustering in a way that would enable you to reason about its time complexity and then do so.
Ans:

```
1. Compute the proximity matrix                          ... O(n^2)
2. Let each data point be a cluster
3. Repeat
      a. Merge the two closest clusters                  ... O(1)
      b. Update the proximity matrix                     ... O(n^2)
4. Until only a single cluster remains
```

This repeat would be for n - 1 times, making the algorithm $(n - 1) * n^2$ = **$O(n^3)$**

Space complexity = $O(n^2)$ for proximity matrix

b. Perform agglomerative hierarchical clustering on the data in Table 2. Use maximum norm distance function and centroid linkage criteria between sets of points. Merge only one pair of clusters in a step and resolve ties by merging sets containing points with smaller IDs first. Draw the final dendrogram to scale, be sure to label axes.
Ans:

First step:
Distance between two points is calculated by maximum norm as follows:
e.g. Distance between (1,1) and (2,1) = max {|1 - 2|, |1 - 1|} = 1

ID	**1**	**2**	**3**	**4**	**5**
1	0	1	2	3	4
2	1	0	2	3	3
3	2	2	0	1	3
4	3	3	1	0	2
5	4	3	3	2	0

Distance Matrix

Minimum distance is 1 (between 1 & 2, 3 &4). But 1 & 2 have smaller IDs, so would be given the preference. Thus, we need to merge those two points as a cluster.

Second step:
Centroid of {1 ∪ 2} = (1 + 2)/2, (1 + 1)/2 = (1.5, 1).

ID	**1 ∪ 2**	**3**	**4**	**5**
1 ∪ 2	0	2	3	3.5
3	2	0	1	3
4	3	1	0	2
5	3.5	3	2	0

Distance Matrix

Minimum distance is 1 (between 3 & 4). This, we need to merge those two points as a cluster.

Third step:
Centroid of {3 ∪ 4} = (2 + 3)/2, (3 + 4)/2 = (2.5, 3.5).

ID	**1 ∪ 2**	**3 ∪ 4**	**5**
1 ∪ 2	0	2.5	3.5
3 ∪ 4	2.5	0	2.5
5	3.5	2.5	0

Distance Matrix

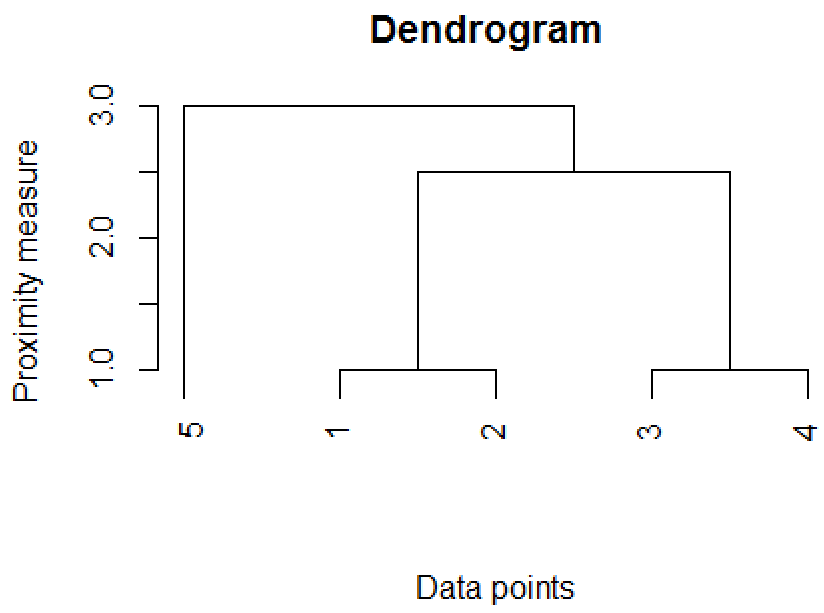
Minimum distance is 2.5 (between {1 ∪ 2} & {3 ∪ 4} and {3 ∪ 4} & 5). But {1 ∪ 2} & {3 ∪ 4} have smaller IDs, so would be given the preference. Thus, we need to merge those two clusters as a new cluster.

Fourth step:
Centroid of {3 ∪ 4} = (1 + 2 + 2 + 3)/4, (1 + 1 + 3 + 4)/4 = (2, 2.25).

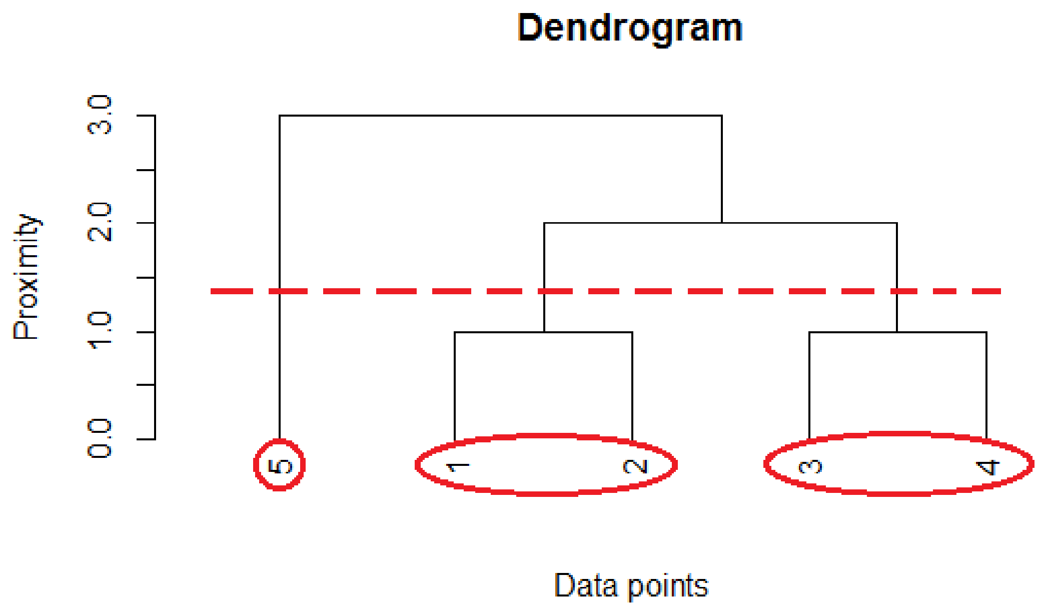
ID	**1 ∪ 2 ∪ 3 ∪ 4**	**5**
1 ∪ 2 ∪ 3 ∪ 4	0	3
5	3	0

Distance Matrix

Minimum distance is 2.5 (between {1∪ 2 ∪ 3 ∪ 4} & 5). Thus, we need to merge that cluster and the point as a new cluster.



c. "The dendrogram structure can be cut at different levels to achieve different clustering results, each with a different number of clusters". What advantage does this provide over k-means clustering?
Ans: For dendrogram structure, it is not mandatory to assume any number of clusters for clustering. **Any desired number of clusters** can be obtained by 'cutting' the dendrogram at the proper level. Cutting above dendrogram at level 1.5 would result into 3 clusters as follows:



3. DBSCAN CLUSTERING

a. Define core, noise, and border points.
Ans:
Core: Core point falls in the interior of density based cluster. A point is a core point if the number of points within a given neighborhood around the point exceeds a certain threshold, (MinPoints in this case). The distance function is calculated using the distance parameter ϵ.
Border points: A point is called border point if it is not a core point, but it falls within the neighborhood of a core point.
Noise: A noise is a point which is neither a core point nor border point.

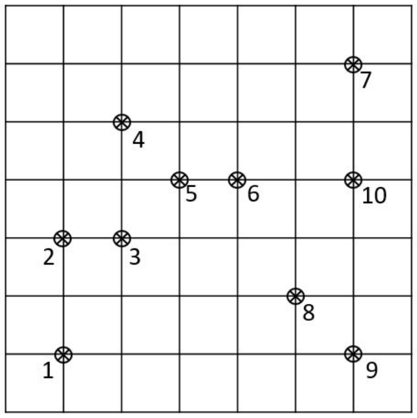
b. Discuss conditions (situations) where DBScan doesn't work well.
Ans:
- Situation where clusters have **widely varying densities**.
- Dealing with **high dimensional data**
- Situation where it is required to calculate **all the possible pairwise proximities** during nearest neighbor computation

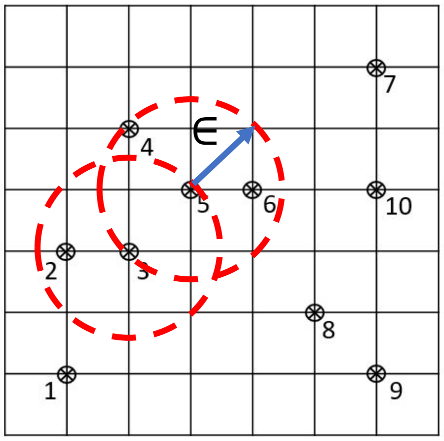
c. Now that you have identified conditions where DBScan doesn't work in (b), discuss measures to improve at least one of those conditions. Please remember to cite any papers or resources you have gone through to answer this question.
Ans: Dealing with high dimensional data can be hard for DBScan algorithm, because density can't be defined perfectly. This can be handled though by **SNN (Shared Nearest Neighbor) Density based clustering**.
Instead of directly applying DBScan, we first compute the SNN similarity graph. SNN similarity gives the number of shared neighbors by two points. In high dimensional data, if the direct similarity between two points is low, then it becomes difficult for clustering algorithms to separate these two points afterwards, if required, once they are put in the same cluster.
Once, SNN density based clustering is done, normal DBScan can be performed.
(Pang-Ning Tan, 2006)

d. Is DBScan deterministic? If yes, explain why. If no, describe how you can make it deterministic.
Ans: DBScan is non-deterministic. This non-determinism is because, it can't determine to which cluster **the border (non-core) point** belongs. It might be possible that it has a distance lower than ϵ to two core points in different clusters. If the triangular inequality is considered, those 2 core points must have at least ϵ, otherwise they would be in the same cluster. The border point is assigned to whichever cluster is generated first using random order.

One of the approach to make it deterministic is to treat border points as noise. And, thus they can be eliminated and the algorithm will be deterministic in nature now.

e. Mark core (C), border (B) and noise points for DBScan on the following dataset. Assume unit squares, MinPoints = 2 units, and radius (EpsilonDist) = 1.5 units. Please note, all data points marked (⊗) are at the intersection of grid lines. Use the table provided below to make it easier for the TAs to grade the question.





Point 3 has ≥ 2 data points (point 2 and point 5) within ϵ, thus it is a core point. Similarly, for point 5. Using the definitions given for core, border and noise, the following table is filled. (Assuming the point itself is not considered as a part of count)

Point	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
C/B/N	N	B	C	B	C	B	N	N	N	N

Bibliography

Pang-Ning Tan, M. S. (2006). Introduction to Data Mining. In *Introduction to Data Mining* (p. 769). Pearson Education, Inc.