DSAA 5002 - Data Mining and Knowledge Discovery in Data Science

(Fall Semester 2023)

Homework 3

Deadline: 22 Nov 2023 11:59pm

(Please hand in via Canvas.) Full Mark: 100 Marks

50015627 JIANG Zhuoyang

Q1 [20 Marks]

Apply the agglomerative hierarchical clustering algorithm with the following distance matrix and the single linkage. Plot the cluster tree and mark out all the merging levels.

	1	2	3	4
2	2.33			
3	3.15	1.30		
4	1.90	1.50	3.70	
5	3.01	$\boxed{0.47}$	1.40	1.82

Table 1： distance matrix

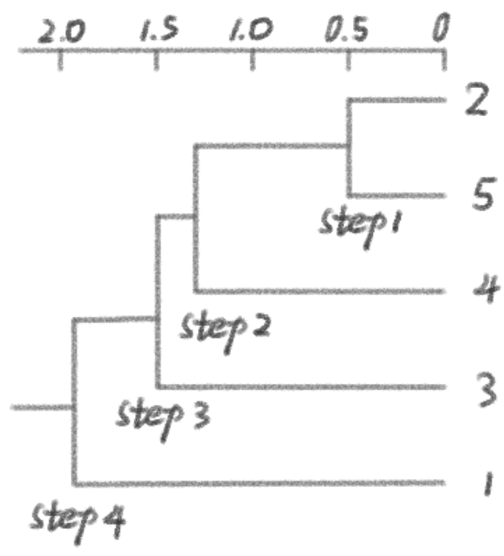
	1	(2,5)	3
⇒ (2,5)	2.33		
3	3.15	$\boxed{1.30}$	
4	1.90	1.50	3.70

	1	(2,3,5)
⇒ (2,3,5)	2.33	
4	1.90	$\boxed{1.50}$

	1
⇒ (2,3,4 5)	$\boxed{1.90}$

⇒ (1, 2, 3, 4, 5)

Cluster Tree:



Q2 [20 Marks]

Use the similarity matrix in Table 2 to perform single-link hierarchical clustering. Show your results by drawing a dendrogram. The dendrogram should clearly show the order in which the clusters are merged.

	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

Table 2: Similarity matrix for Q2

⇒

	P_1	(P_2, P_5)	P_3	P_4
P_1	1.0			
(P_2, P_5)	0.35	1.0		
P_3	0.41	0.85	1.0	
P_4	0.55	0.76	0.44	1.0

⇒

	P_1	(P_2, P_3, P_5)	P_4
P_1	1.0		
(P_2, P_3, P_5)	0.41	1.0	
P_4	0.55	0.76	1.0

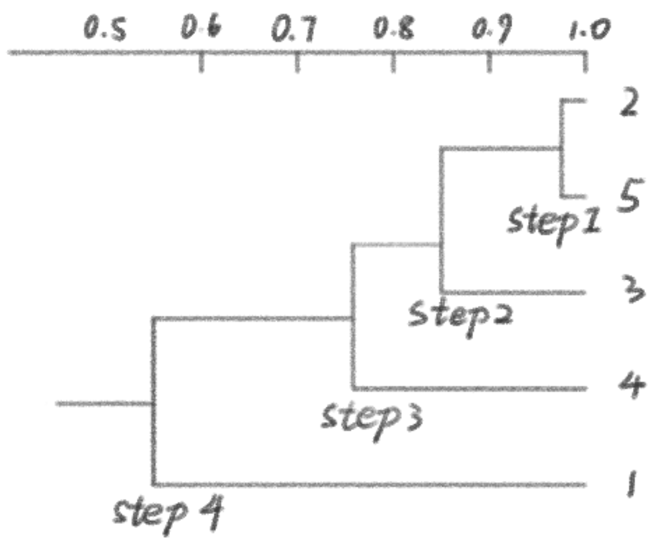
⇒

	P_1	(P_2, P_3, P_4, P_5)
P_1	1.0	
(P_2, P_3, P_4, P_5)	0.55	1.0

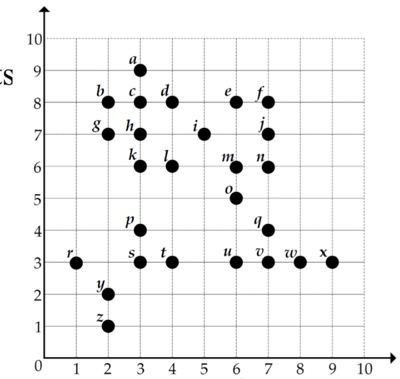
⇒

	$(P_1, P_2, P_3, P_4, P_5)$
$(P_1, P_2, P_3, P_4, P_5)$	1.0

Cluster Tree:



Q3 [30 Marks]



Apply DBSCAN with parameters MinPts=4 and Eps = $\sqrt{2}$ to get clustering results.

First, for every data point, answer if it is a core, a border, or an outlier.

Second, for data points that are not outliers, show the clusters detected.

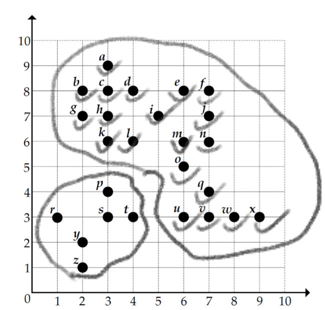
Third, show your detailed steps of DBSCAN process, including the content of the queue you maintain, whenever a new core is found.

1st. for each data point p, we calculate the $\sqrt{2}$-neighborhood of p — $N(p)$, and judge its type with MinPts = 4:

N(a) = {a.b.c.d}	core		N(n) = {j,m,n,o}	core
N(b) = {a,b.c,g.h}	core		N(o) = {m,n,o,q}	core
N(c) = {a.b.c.d,g.h}	core		N(p) = {p,s,t}	border
N(d) = {a,c,d,h,i}	core		N(q) = {o,q,u,v,w}	core
N(e) = {e.f,i,j}	core		N(r) = {r,y}	border
N(f) = {e.f.j}	border		N(s) = {p,s,t,y}	core
N(g) = {b.c,g,h,k}	core		N(t) = {p,s,t}	border
N(h) = {b.c,d,g,h,k,l}	core		N(u) = {q,u,v}	border
N(i) = {d.e,i,l,m}	core		N(v) = {q,u,v,w}	core
N(j) = {e,f,j,m,n}	core		N(w) = {q,v,w,x}	core
N(k) = {g,h,k,l}	core		N(x) = {w,x}	border
N(l) = {h,i,k,l}	core		N(y) = {r,s,y,z}	core
N(m) = {i,j,m,n,o}	core		N(z) = {y,z}	border

core points: {a,b,c,d,e,g,h,i,j,k,l,m,n,o,q,s,v,w,y}
border points: {f,p,r,t,u,x,z}   outlier: Φ

2nd. Clusters detected: [unprocessed points] = {a,b,c,..., x,y,z}



1° ① Arbitrary select a point. here we choose 'a'
② Retrieve all the points density-reachable from 'a' {b.c.d.e.f.g.h.i,j,k,l.m.n,o.q.u.v.w.x}
③ Here, 'a' is a core point. a cluster is formed:
Cluster_1 = {a,b.c.d.e.f.g.h.i,j,k,l.m.n.o.q.u.v.w.x}
④ [unprocessed points] = {p.r.s.t.y,z}

2° ① Next, select a point from [unprocessed points]. here we choose 'p'
② Here, 'p' is a border point, just visit the next point
③ Then visit next point from [unprocessed points]. here we choose 's'
④ Retrieve all the points density-reachable from 's': {p.r.t,y,z}
⑤ Here p is a core point. a cluster is formed:
Cluster_2 = {p,r,s,t,y,z}
⑥ [unprocessed points] = Φ, the algorithm stops here.

Q4 [20 Marks] Fuzzy Cluster

Assume there are 2 clusters in which the data is to be divided, initializing the data point randomly. Each data point lies in both clusters with some membership value which can be assumed anything in the initial state.

The table below represents the values of the data points along with their membership (gamma) in each cluster.

Cluster	(1,3)	(2,5)	(4,8)	(7,9)	(9,12)
1)	0.8	0.7	0.5	0.3	0.1
2)	0.2	0.3	0.5	0.7	0.9

Please work out the centroids, the distance of each point from centroid, and the cluster membership value.

Here, we have:

5 Data Points: $O = \{O_1, O_2, O_3, O_4, O_5\} = \{(1,3),(2,5),(4,8),(7,9),(9,12)\} \Rightarrow O_i = (X_i^{(0)}, Y_i^{(0)})$

2 Clusters: $C = \{C_1, C_2\}$, with their centroids $\{c_1, c_2\} \Rightarrow c_j = (x_j^{(c)}, y_j^{(c)})$

$\Rightarrow$ ① the distance of each point from centroid:

$$dist(O_i, C_j)^2 = (x_i^{(0)} - x_j^{(c)})^2 + (y_i^{(0)} - y_j^{(c)})^2$$

② Cluster membership value

$$w_{ij} = \frac{\frac{1}{dist(O_i, C_j)^2}}{\sum_{j=1}^{2} \frac{1}{dist(O_i, C_j)^2}} \Rightarrow \begin{cases} w_{i1} = \dfrac{dist(O_i, C_2)^2}{dist(O_i, C_1)^2 + dist(O_i, C_2)^2} \\ w_{i2} = \dfrac{dist(O_i, C_1)^2}{dist(O_i, C_1)^2 + dist(O_i, C_2)^2} \end{cases}$$

③ $$SSE(C_j) = \sum_{i=1}^{5} w_{ij}^2 \, dist(O_i, C_j)^2$$

0° We randomly set the centroid: $C_1 = O_1 = (1,3)$, $C_2 = O_2 = (2,5)$

1° In E-Step, we use centroid $C_1 = (x_1^{(c)}, y_1^{(c)})$ & $C_2 = (x_2^{(c)}, y_2^{(c)})$ to calculate σ_{i1}, σ_{i2}.

$$\boxed{\begin{cases} w_{i1} = \dfrac{(x_i^{(0)} - x_2^{(c)})^2 + (y_i^{(0)} - y_2^{(c)})^2}{(x_i^{(0)} - x_1^{(c)})^2 + (y_i^{(0)} - y_1^{(c)})^2 + (x_i^{(0)} - x_2^{(c)})^2 + (y_i^{(0)} - y_2^{(c)})^2} \\ w_{i2} = \dfrac{(x_i^{(0)} - x_1^{(c)})^2 + (y_i^{(0)} - y_1^{(c)})^2}{(x_i^{(0)} - x_1^{(c)})^2 + (y_i^{(0)} - y_1^{(c)})^2 + (x_i^{(0)} - x_2^{(c)})^2 + (y_i^{(0)} - y_2^{(c)})^2} \end{cases}}$$

2° In M-Step, we do optimization on $SSE(C_1)$ & $SSE(C_2)$ for $C_1 = (x_1^{(c)}, y_1^{(c)})$ & $C_2 = (x_2^{(c)}, y_2^{(c)})$, to get new c_1^* & c_2^*.

$$\frac{\partial SSE(C_j)}{\partial (x_j^{(c)}, y_j^{(c)})} = \left[\frac{\partial SSE(C_j)}{\partial x_j^{(c)}}, \frac{\partial SSE(C_j)}{\partial y_j^{(c)}} \right]$$

$$\Rightarrow \begin{cases} \dfrac{\partial SSE(C_j)}{\partial x_j^{(c)}} = -2 \sum_{i=1}^{5} w_{ij}^2 (x_i^{(0)} - x_j^{(c)}) \\ \dfrac{\partial SSE(C_j)}{\partial y_j^{(c)}} = -2 \sum_{i=1}^{5} w_{ij}^2 (y_i^{(0)} - y_j^{(c)}) \end{cases}$$

when $\dfrac{\partial SSE(C_j)}{\partial (x_j^{*(c)}, y_j^{*(c)})} = 0$, it is easy to prove from the second derivative of $SSE(C_j)$ that the stationary point $(x_j^{*(c)}, y_j^{*(c)})$ here is the Minimum Point :

$$\begin{cases} -2 \sum_{i=1}^{5} w_{ij}^2 (x_i^{(0)} - x_j^{(c)}) = 0 \Rightarrow x_j^{*(c)} = \dfrac{\sum_{i=1}^{5} w_{ij}^2 x_i^{(0)}}{\sum_{i=1}^{5} w_{ij}^2} \\ -2 \sum_{i=1}^{5} w_{ij}^2 (y_i^{(0)} - y_j^{(c)}) = 0 \Rightarrow y_j^{*(c)} = \dfrac{\sum_{i=1}^{5} w_{ij}^2 y_i^{(0)}}{\sum_{i=1}^{5} w_{ij}^2} \end{cases}$$

$$\Rightarrow C_j^* = \left(\frac{\sum_{i=1}^{5} w_{ij}^2 x_i^{(0)}}{\sum_{i=1}^{5} w_{ij}^2}, \frac{\sum_{i=1}^{5} w_{ij}^2 y_i^{(0)}}{\sum_{i=1}^{5} w_{ij}^2} \right)$$

so that, centroid will be renewed as below

$$\boxed{\begin{cases} C_1^* = \left(\dfrac{\sum_{i=1}^{5} w_{i1}^2 x_i^{(0)}}{\sum_{i=1}^{5} w_{i1}^2}, \dfrac{\sum_{i=1}^{5} w_{i1}^2 y_i^{(0)}}{\sum_{i=1}^{5} w_{i1}^2} \right) \\ C_1^* = \left(\dfrac{\sum_{i=1}^{5} w_{i2}^2 x_i^{(0)}}{\sum_{i=1}^{5} w_{i2}^2}, \dfrac{\sum_{i=1}^{5} w_{i2}^2 y_i^{(0)}}{\sum_{i=1}^{5} w_{i2}^2} \right) \end{cases}}$$

3° Cycle through E-step & M-step until the change of centroids is sufficiently small.

★ Programming to implement the specific calculationg process of the EM Algorithm above:

```python
import numpy as np

# Distance definition
def calculate_squared_distance(point, centroid):
    return np.sum((point - centroid) ** 2)
```

```python
# Caculate the centroid position with given membership
# Data points
points = np.array([[1.0, 3.0], [2.0, 5.0], [4.0, 8.0], [7.0, 9.0], [9.0, 12.0]])

# Initial membership
membership = np.array([[0.8, 0.7, 0.5, 0.3, 0.1],
                       [0.2, 0.3, 0.5, 0.7, 0.9]]).T

#M-step: Update centroid position
for i in range(2):
    # Centroid that will optimizate SSE
    centroids_numerator = np.sum(membership[:, i][:, None] ** 2 * points, axis=0).astype(float)
    centroids_denominator = np.sum(membership[:, i]** 2).astype(float)
    centroids[i] = centroids_numerator / centroids_denominator

print("Centroids with given membership:")
print(centroids)
```

Centroids with given membership in output:

```
Centroids with given membership:
[[2.25675676 4.93243243]
 [7.10714286 9.94047619]]
```

```python
# Iterative Execution of EM Algorithm
iter_time = 0
while 1:
    iter_time += 1

    # Calculate the square distance from each point to each center point
    squared_distances = np.array([[calculate_squared_distance(point, centroid) \
                                  for centroid in centroids] \
                                 for point in points])

    # E-step: Update membership based on square distance
    new_membership_denominator = np.sum(squared_distances, axis=1)

    new_membership_numerator = squared_distances
    new_membership_numerator[:, [0, 1]] = new_membership_numerator[:, [1, 0]]

    new_membership =  new_membership_numerator/ new_membership_denominator[:, None]

    mean_change_of_membership = np.mean(np.abs(new_membership-membership))
    membership = new_membership

    #M-step: Update centroid position
    for i in range(len(centroids)):
        # New centroid that will optimizate SSE
        centroids_numerator = np.sum(new_membership[:, i][:, None] ** 2 * points, axis=0).astype(float)
        centroids_denominator = np.sum(new_membership[:, i]** 2).astype(float)
        centroids[i] = centroids_numerator / centroids_denominator

    print(f"Iteration {iter_time}:")
    print("Distance:")
    print(squared_distances)
    print("New Membership:")
    print(membership)
    print("New Centroids:")
    print(centroids)
    print("Mean change of Menbership:")
    print(mean_change_of_membership)
    print("----------")

    # Two shutdown criteria
    if mean_change_of_membership < 0.00001:
        break
    if iter_time > 20:
        break
```

The Iteration 1 to 3 in output shows below:

```
Iteration 1:
Distance:
[[8.54674036e+01 5.31373265e+00]
 [5.04912132e+01 7.04894083e-02]
 [1.34197846e+01 1.24488678e+01]
 [8.95975057e-01 3.90434624e+01]
 [7.82454649e+00 9.54218408e+01]]
New Membership:
[[0.94146655 0.05853345]
 [0.99860587 0.00139413]
 [0.51876628 0.48123372]
 [0.02243334 0.97756666]
 [0.07578518 0.92421482]]
New Centroids:
[[ 1.85854048  4.57240718]
 [ 7.48562706 10.12986197]]
Mean change of Menbership:
0.15212403662985943
----------
```

```
Iteration 2:
Distance:
[[ 92.89829012   3.20955612]
 [ 56.40758811   0.20284641]
 [ 16.68590803  16.33424136]
 [  1.51242172  46.03818409]
 [  5.79074164 106.16957904]]
New Membership:
[[0.96660464 0.03339536]
 [0.9964168  0.0035832 ]
 [0.50532503 0.49467497]
 [0.03180657 0.96819343]
 [0.05172137 0.94827863]]
New Centroids:
[[ 1.81711109  4.50607855]
 [ 7.50785987 10.17469156]]
Mean change of Menbership:
0.014841089128195067
----------
```

```
Iteration 3:
Distance:
[[ 93.82843918   2.93594314]
 [ 57.11395318   0.27740675]
 [ 17.0343643   16.97249107]
 [  1.63782192  47.05766741]
 [  5.55823304 107.75275173]]
New Membership:
[[0.96965884 0.03034116]
 [0.9951664  0.0048336 ]
 [0.50090972 0.49909028]
 [0.03363396 0.96636604]
 [0.04905291 0.95094709]]
New Centroids:
[[ 1.80965236  4.49365897]
 [ 7.50554914 10.17718346]]
Mean change of Menbership:
0.0026431543367460945
----------
```