EE 4146 Data Engineering and Learning Systems

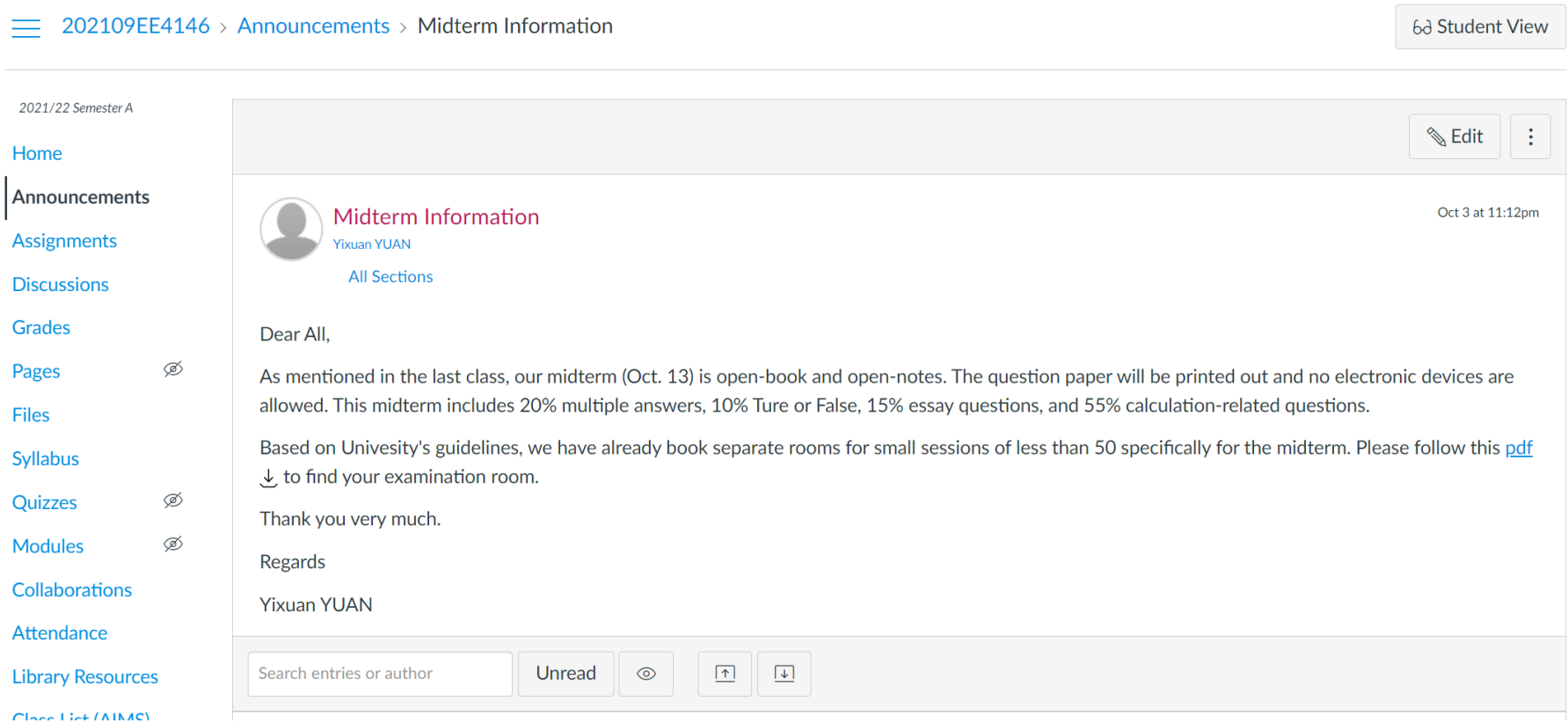
Lecture 6: Clustering II

Semester A, 2021-2022

Schedules

Week	Date	Topics
1	Sep. 1	Introduction
2	Sep. 8	Data exploration
3	Sep. 15	Feature reduction and selection (HW1 out)
4	Sep. 22	Mid-Autumn Festival
5	Sep. 29	Clustering I: Kmeans based models (HW1 due in this weekend)
6	Oct. 6	Clustering II: Hierarchical/density based/fuzzing clustering
7	Oct. 13	Midterm (no tutorials this week)
8	Oct. 20	Linear classifiers
9	Oct. 27	Classification based on decision tree (Tutorial on project) (HW2 out)
10	Nov. 3	Bayes based classifier (Tutorial on codes) (HW2 due in this weekend)
11	Nov. 10	KNN and classifier ensemble
12	Nov. 17	Deep learning based models (Quiz)
13	Nov. 24	Summary

Arrangement of Midterm



Outline

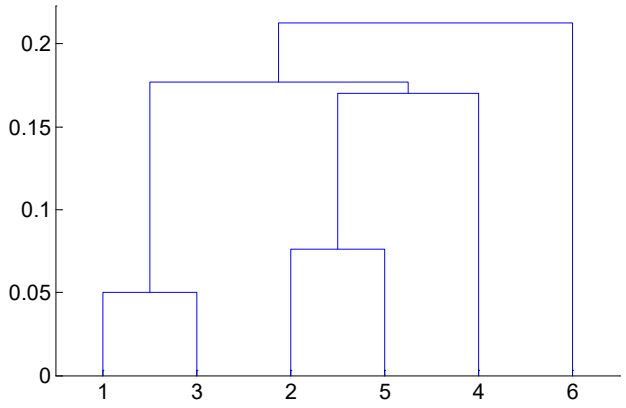
- Hierarchical Clustering
- Clustering based on density
- Fuzzing clustering

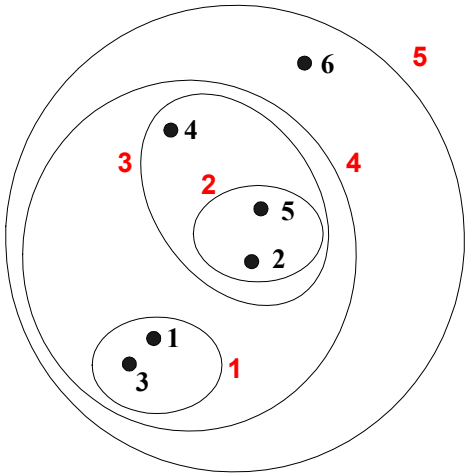
Hierarchical Clustering

- K-means is an objective-based approach that requires us to pre-specify the number of clusters K
- The answer it gives is somewhat random: it depends on the random initialization we started with
- Hierarchical clustering is an alternative approach that does not require a pre-specified choice of K, and which provides a deterministic answer (no randomness)

Hierarchical Clustering

- Produce a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges or splits

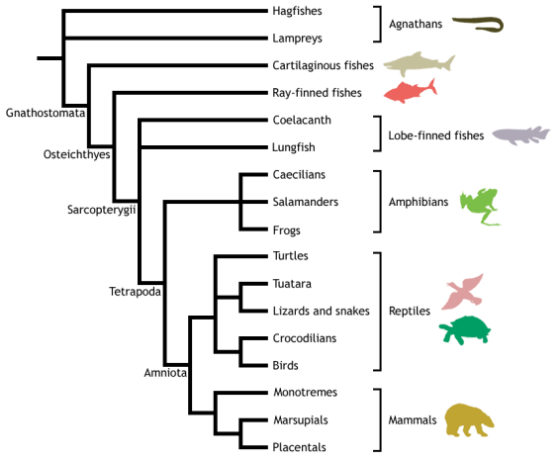




Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by ‘cutting’ the dendrogram at the proper level

- They may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, …)



Hierarchical Clustering

- Two main types of hierarchical clustering
  - Agglomerative:
  - Start with the points as individual clusters
  - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left

  - Divisive:
  - Start with one, all-inclusive cluster
  - At each step, split a cluster until each cluster contains an individual point (or there are k clusters)

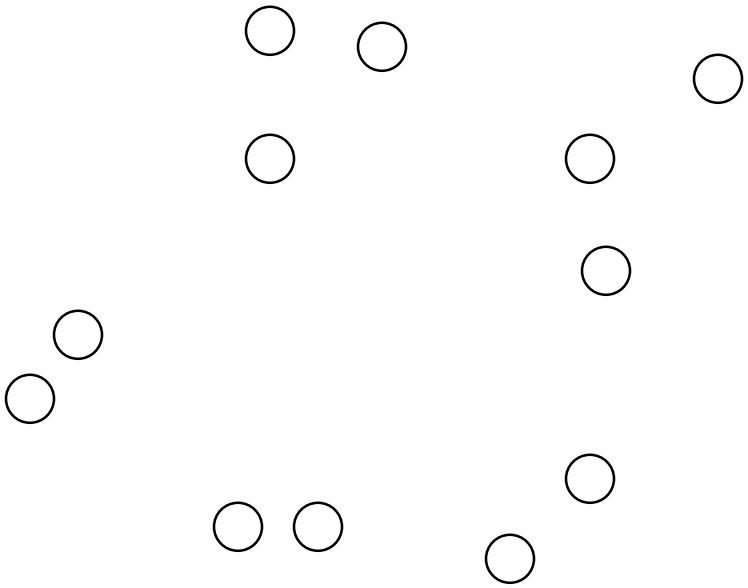
- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time

Agglomerative Clustering Algorithm

- Most popular hierarchical clustering technique
- Basic algorithm is straightforward
  1. Compute the proximity matrix
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the proximity matrix
  6. **Until** only a single cluster remains
- Key operation is the computation of the proximity of two clusters
  - Different approaches to defining the distance between clusters distinguish the different algorithms

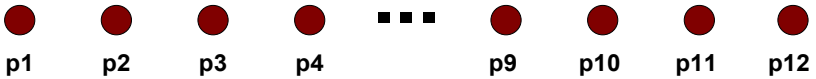
Starting Situation

- Start with clusters of individual points and a proximity matrix



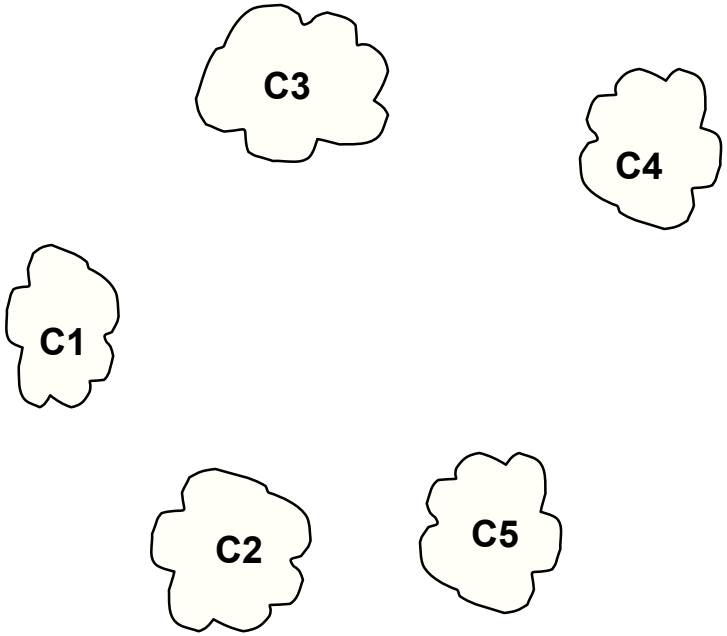
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix



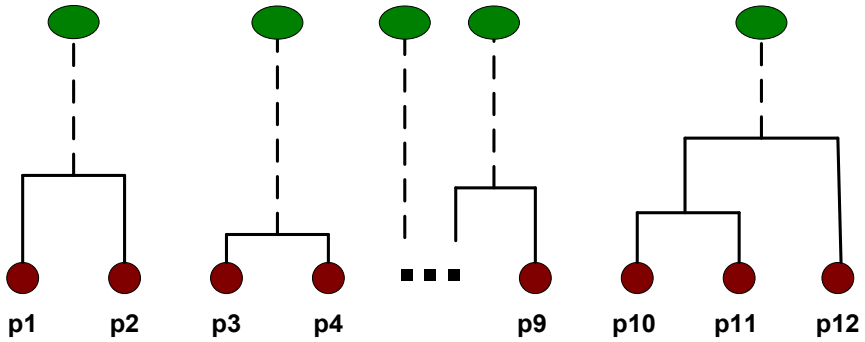
Intermediate Situation

- After some merging steps, we have some clusters



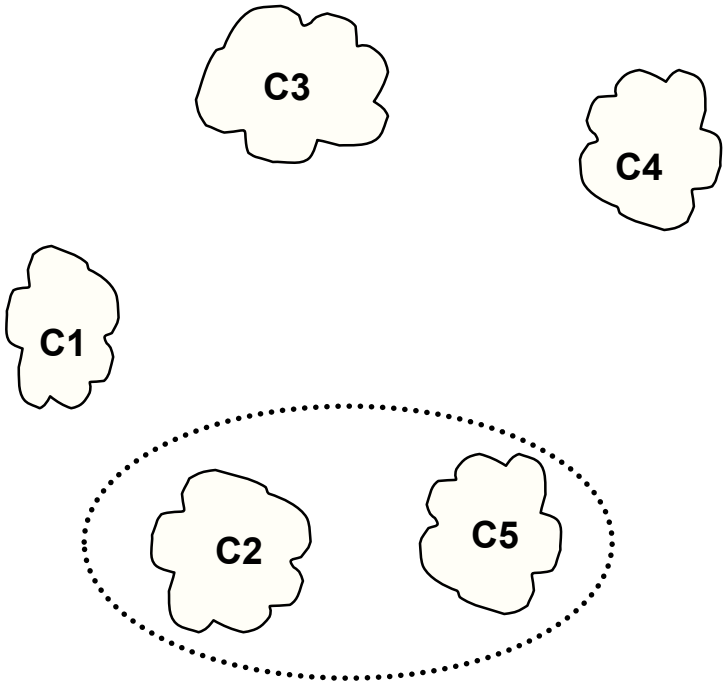
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

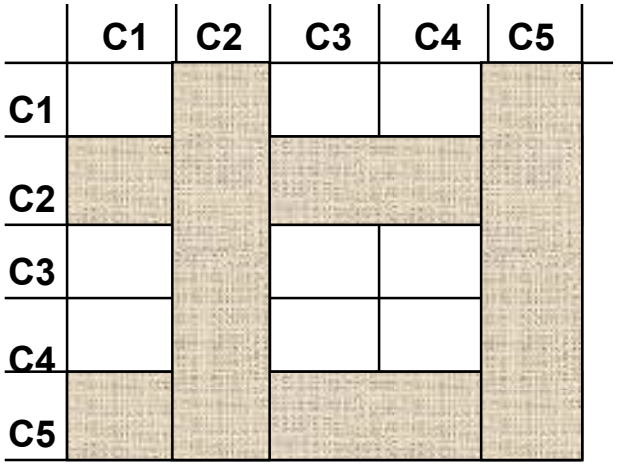
Proximity Matrix



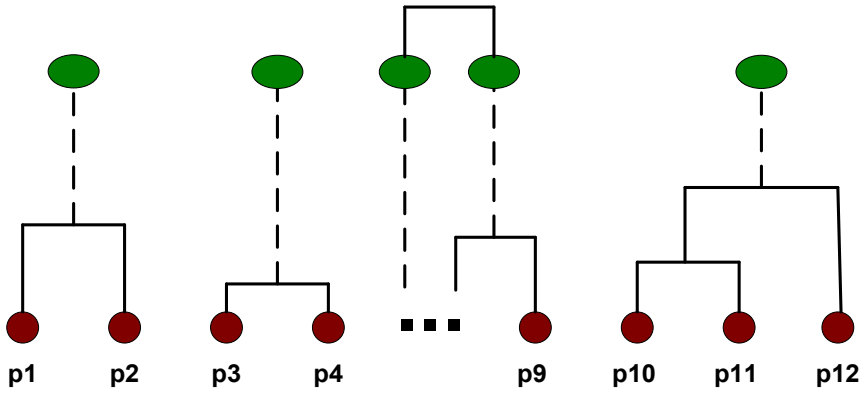
Intermediate Situation

- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.



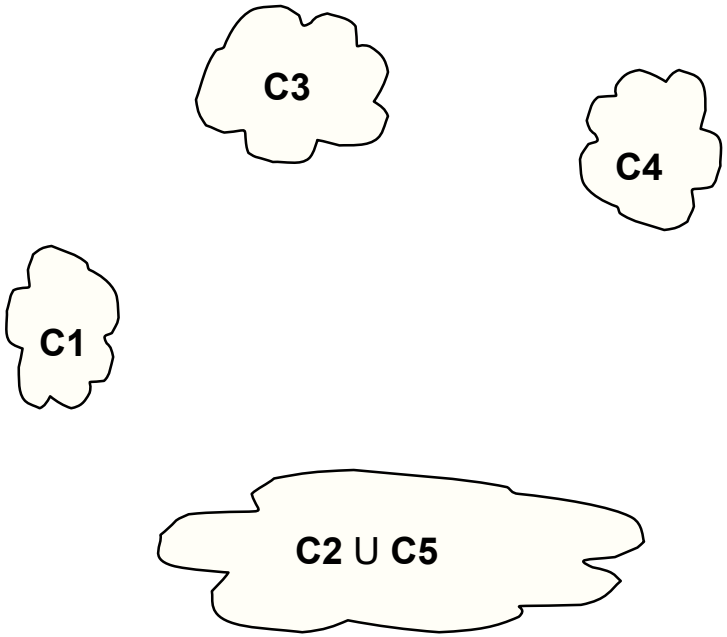


Proximity Matrix



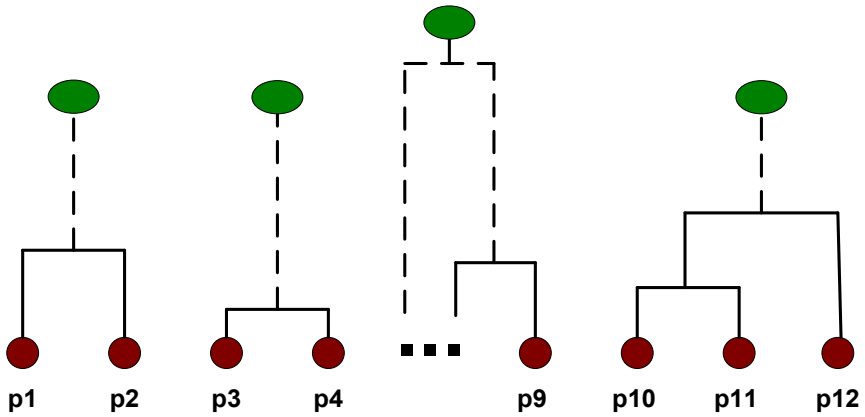
After Merging

- The question is “How do we update the proximity matrix?”

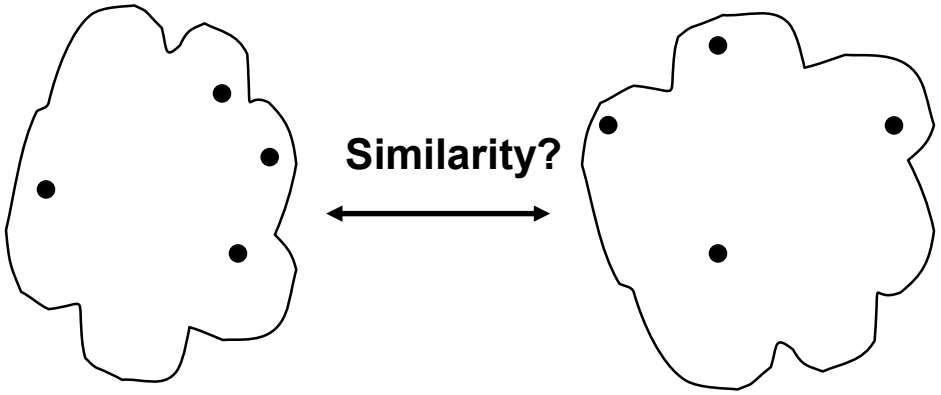


	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



After Merging

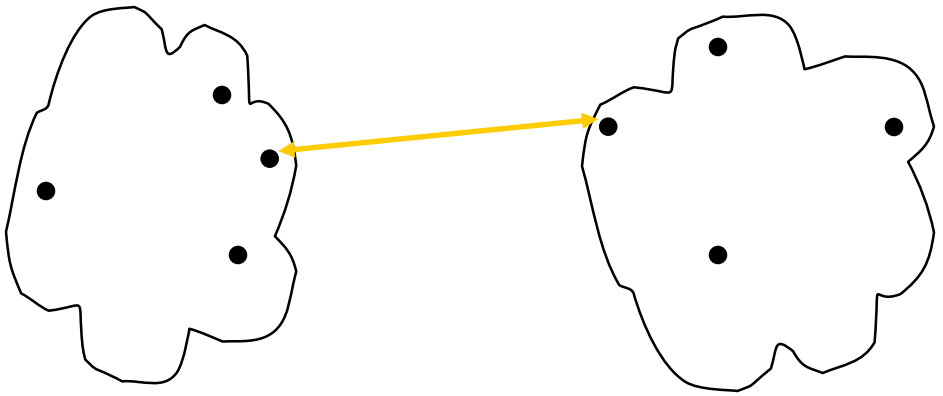


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

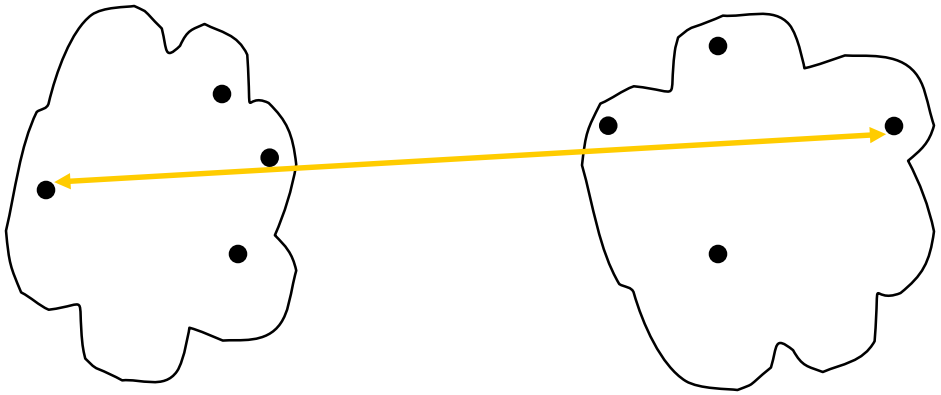


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

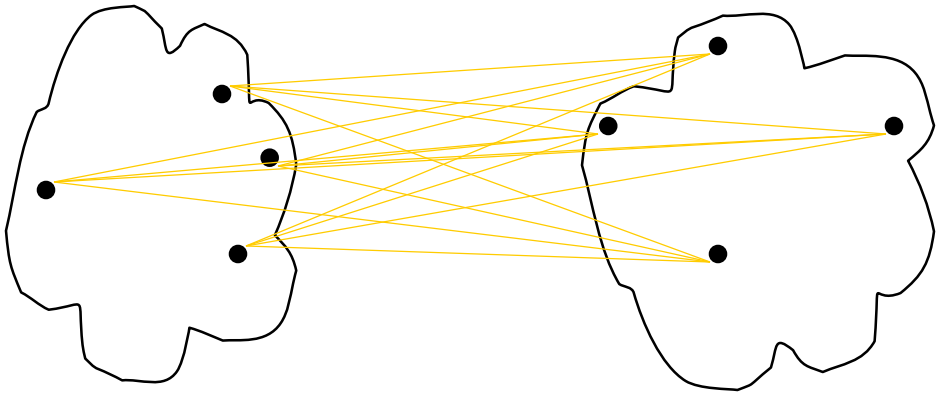


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

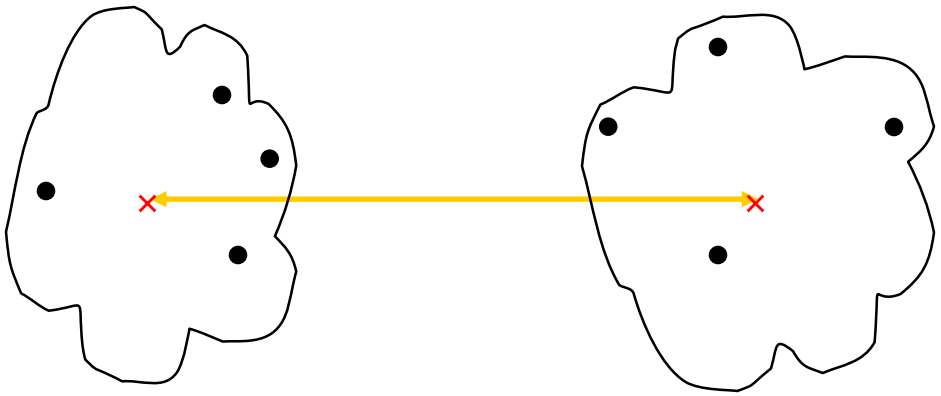


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

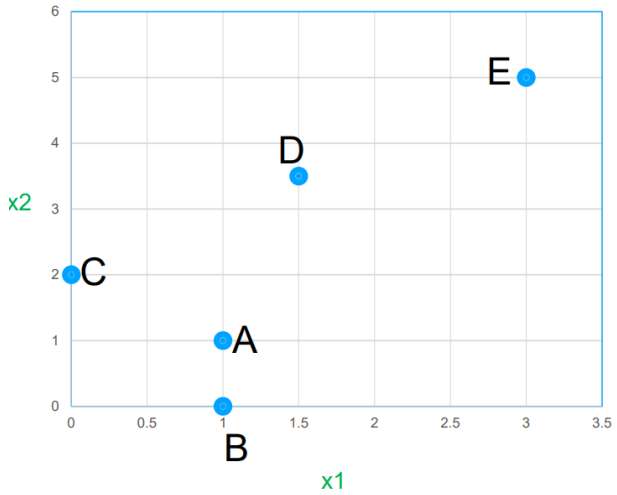
Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

Example 1: MIN or Single Link

- Proximity of two clusters is based on the two closest points in the different clusters
  - Determined by one pair of points, i.e., by one link in the proximity graph

i	X1	X2
A	1	1
B	1	0
C	0	2
D	1.5	3.5
E	3	5



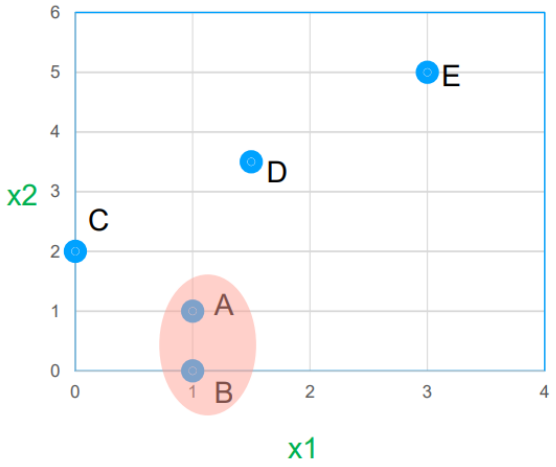
EUCLIDEAN DISTANCE

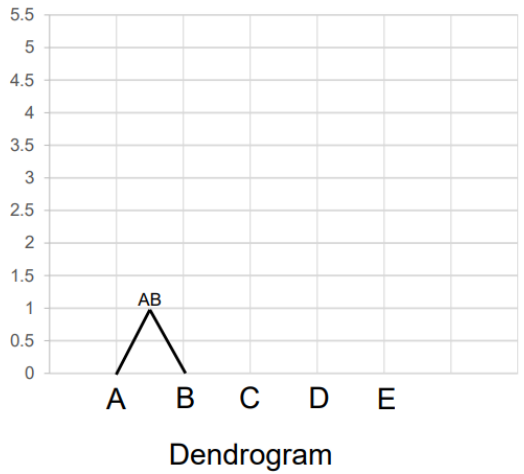
	A	B	C	D	E
A	0	1	1.4	2.55	4.5
B	1	0	2.2	3.53	5.4
C	1.4	2.2	0	2.12	4.2
D	2.55	3.53	2.12	0	2.12
E	4.5	5.4	4.2	2.12	0

Example 1: MIN or Single Link

	A	B	C	D	E
A	0	1	1.4	2.55	4.5
B	1	0	2.2	3.53	5.4
C	1.4	2.2	0	2.12	4.2
D	2.55	3.53	2.12	0	2.12
E	4.5	5.4	4.2	2.12	0

	(A,B)	C	D	E
(A,B)	0	1.4	2.55	4.5
C	1.4	0	2.12	4.2
D	2.55	2.12	0	2.12
E	4.5	4.2	2.12	0

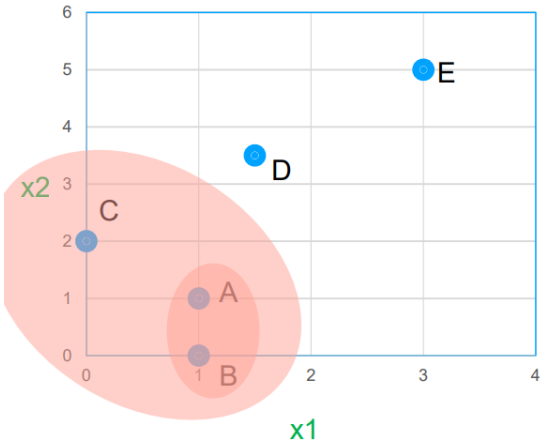


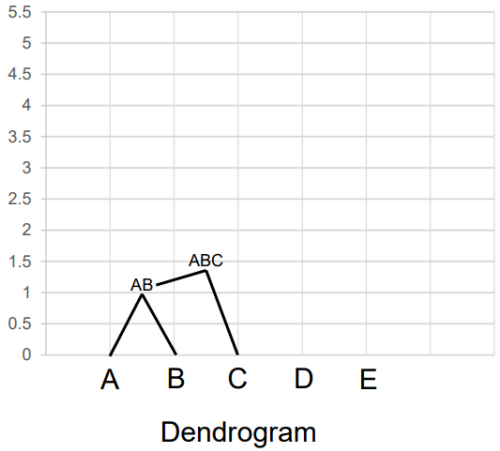


Example 1: MIN or Single Link

	(A,B)	C	D	E
(A,B)	0	1.4	2.55	4.5
C	1.4	0	2.12	4.2
D	2.55	2.12	0	2.12
E	4.5	4.2	2.12	0

	(A,B),C	D	E
(A,B),C	0	2.12	4.2
D	2.12	0	2.12
E	4.2	2.12	0

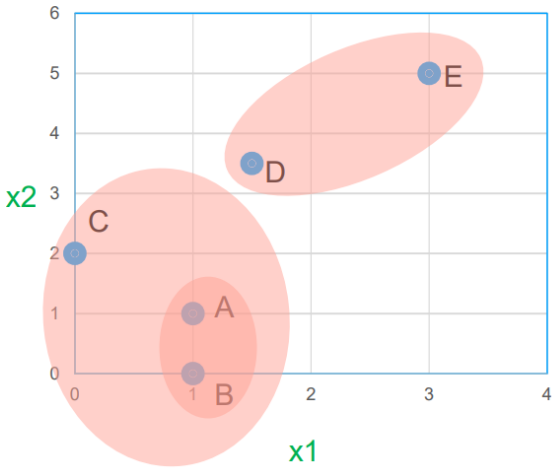


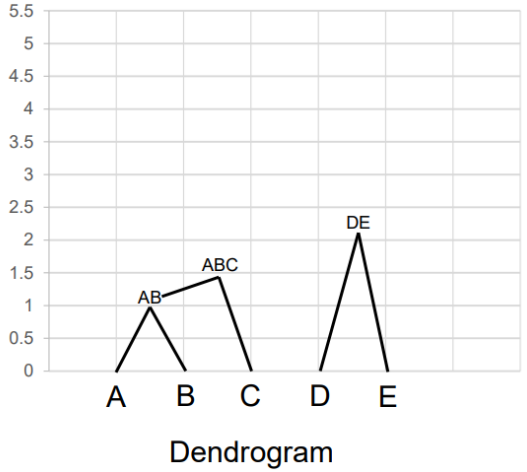


Example 1: MIN or Single Link

	(A,B), C	D	E
(A,B), C	0	2.12	4.2
D	2.12	0	2.12
E	4.2	2.12	0

	((A,B),C)	(D,E)
((A,B),C)	0	2.12
(D,E)	2.12	0

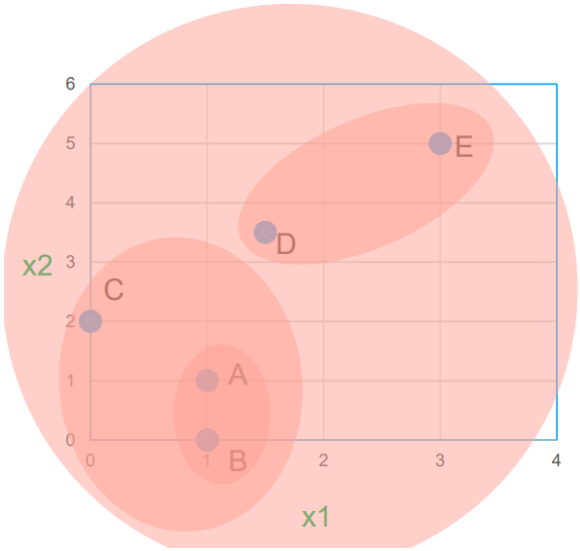


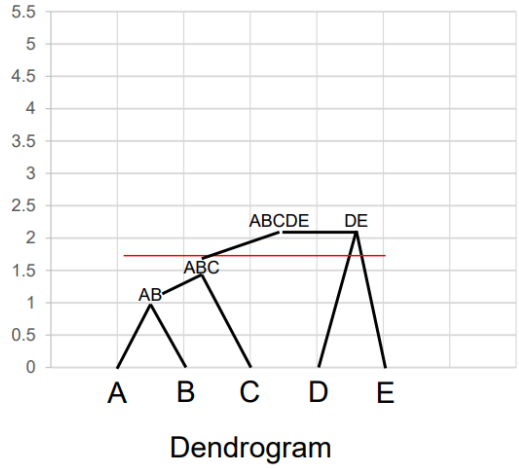


Example 1: MIN or Single Link

	((A,B),C)	(D,E)
((A,B),C)	0	2.12
(D,E)	2.12	0

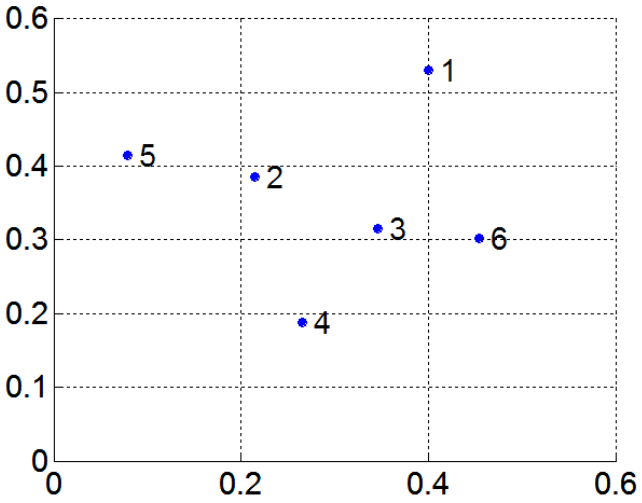
	(((A,B),C),(D,E))
(((A,B),C),(D,E))	0





MIN or Single Link (example 2)

- Proximity of two clusters is based on the two closest points in the different clusters
  - Determined by one pair of points, i.e., by one link in the proximity graph
- Example:



Distance Matrix:

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

MIN or Single Link (example 2)

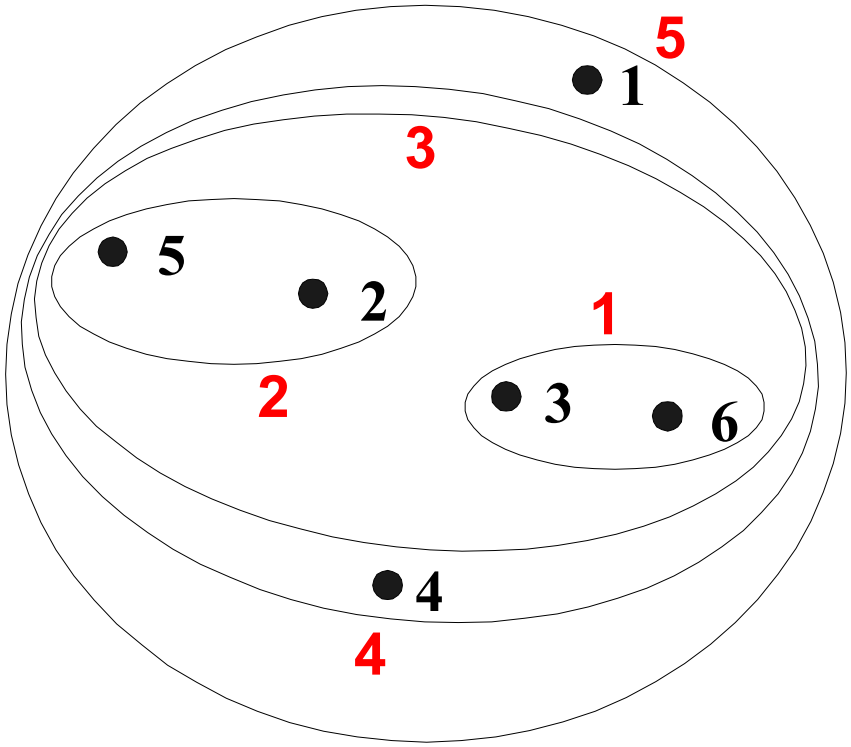
	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

	P1	P2	P3/6	P4	P5
P1	0	0.24	0.22	0.37	0.34
P2	0.24	0	0.15	0.2	0.14
P3/6	0.22	0.15	0	0.15	0.28
P4	0.37	0.2	0.15	0	0.29
P5	0.34	0.14	0.28	0.29	0

	P1	P2/5	P3/6	P4
P1	0	0.24	0.22	0.37
P2/5	0.24	0	0.15	0.2
P3/6	0.22	0.15	0	0.15
P4	0.37	0.2	0.15	0

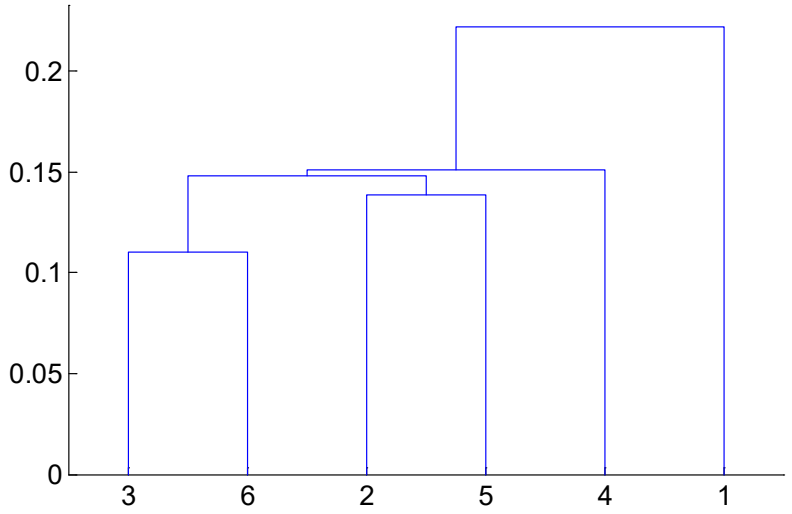
	P1	P2/5/3/6	P4
P1	0	0.22	0.37
P2/5/3/6	0.22	0	0.15
P4	0.37	0.15	0

MIN or Single Link (example 2)



Nested Clusters

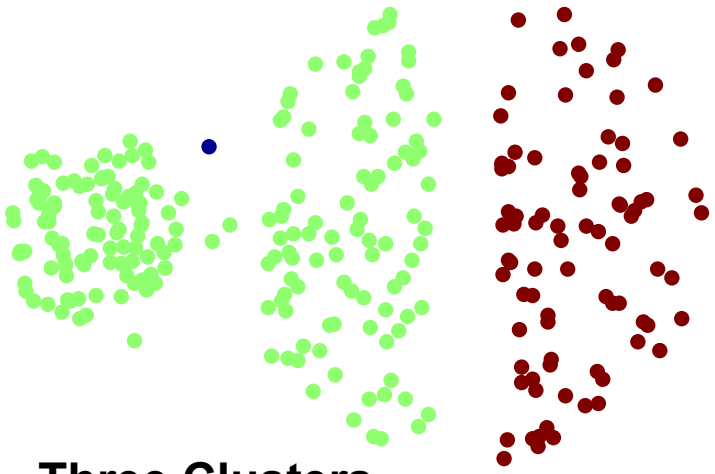
	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00



Dendrogram

Limitations of MIN

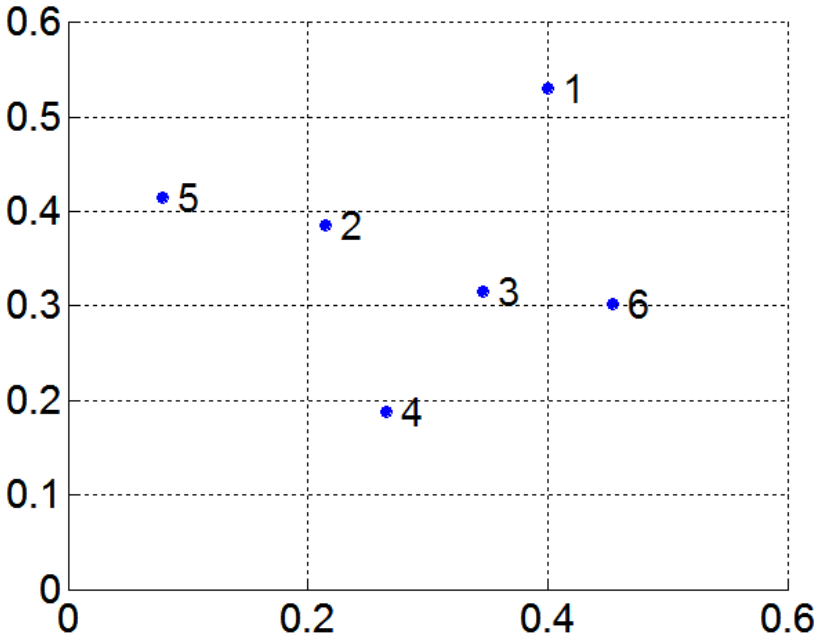
- Sensitive to noise and outliers
- This calculation method is easy to cause an effect called Chaining.
- Two clusters are far away from each other in the "big picture", but some points are close to each other, so the Chaining effect will be further expanded after the merging, and finally a relatively loose cluster will be formed.



Three Clusters

MAX or Complete Linkage

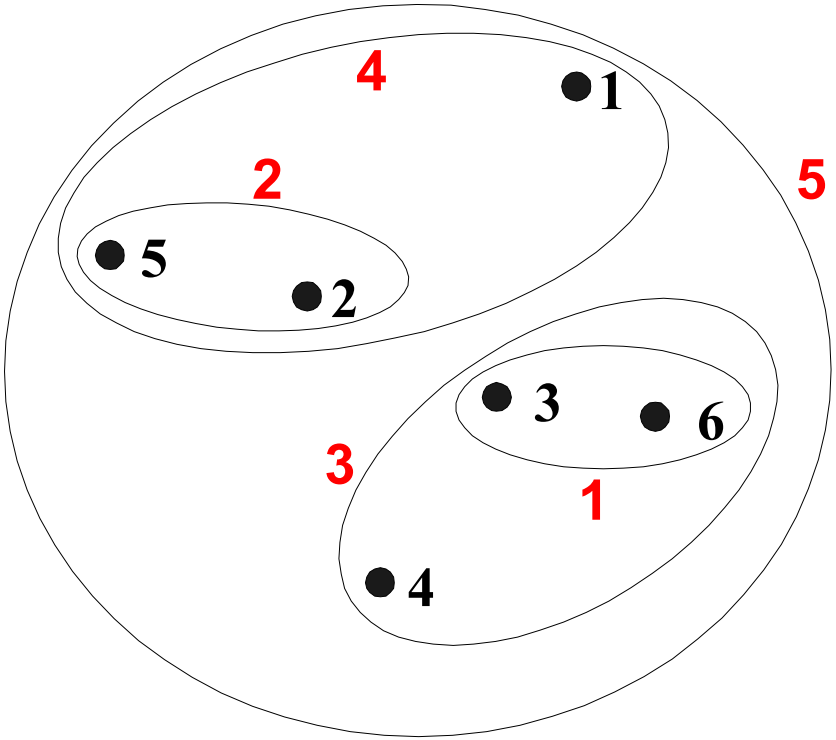
- Proximity of two clusters is based on the two most distant points in the different clusters
  - Determined by all pairs of points in the two clusters



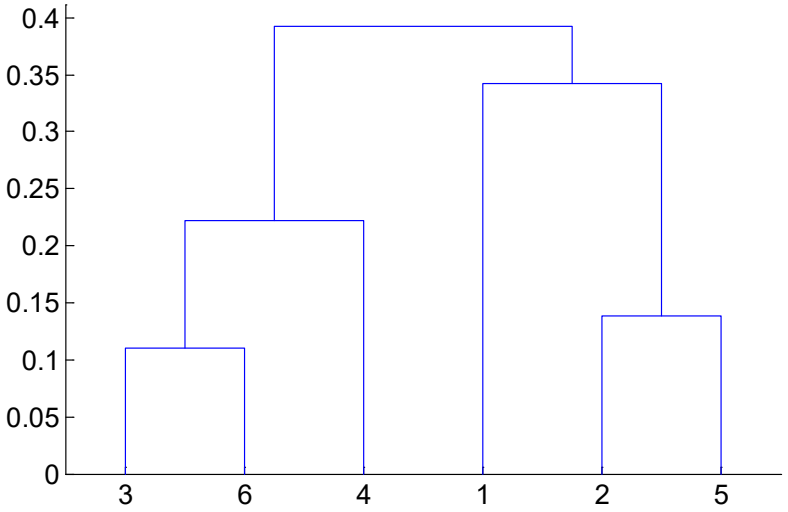
Distance Matrix:

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Hierarchical Clustering: MAX



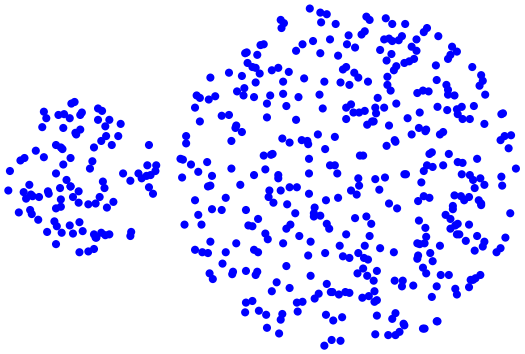
Nested Clusters



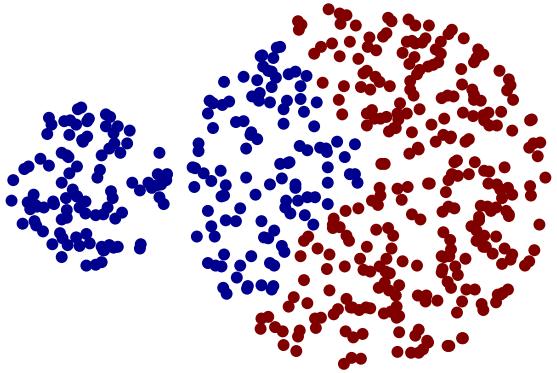
Dendrogram

Limitations of MAX

- Tends to break large clusters
- Even though the two clusters are close to each other, as long as there are special points, they will not merge with each other.
- Ignore the overall characteristic of the data within the class.



Original Points



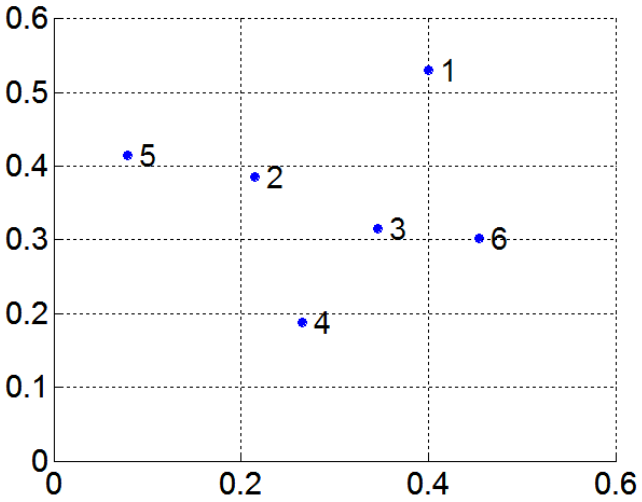
Two Clusters

Group Average

- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum_{\substack{p_i \in \text{Cluster}_i \\ p_j \in \text{Cluster}_j}} \text{proximity}(p_i, p_j)}{|\text{Cluster}_i| \times |\text{Cluster}_j|}$$

- Need to use average connectivity for scalability since total proximity favors large clusters



Distance Matrix:

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Hierarchical Clustering: Group Average

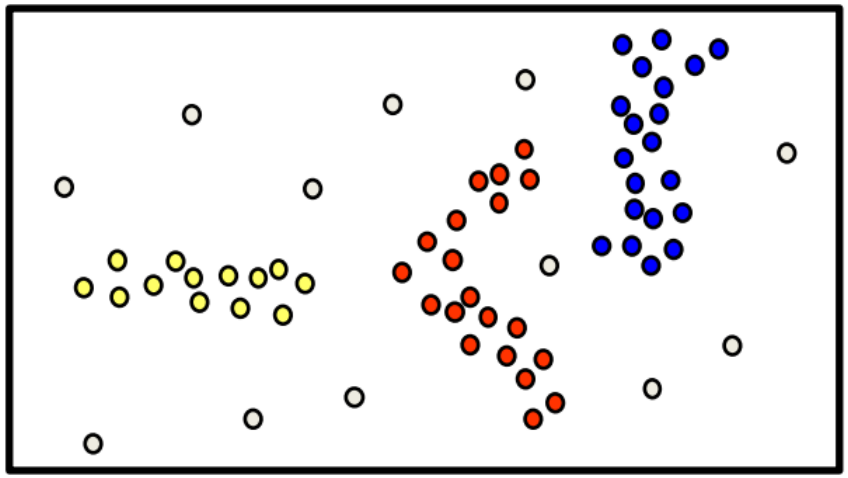
- Compromise between Single and Complete Link
- Strengths
  - Less susceptible to noise and outliers
- Limitations
  - Biased towards globular clusters

Outline

- Hierarchical Clustering
- Clustering based on density
- Fuzzing clustering

Density-based Clustering

- Basic idea
  - Clusters are dense regions in the data space, separated by regions of lower object density
  - A cluster is defined as a maximal set of densityconnected points
  - Discovers clusters of arbitrary shape
- Method
  - DBSCAN: Density-Based Spatial Clustering of Applications with Noise

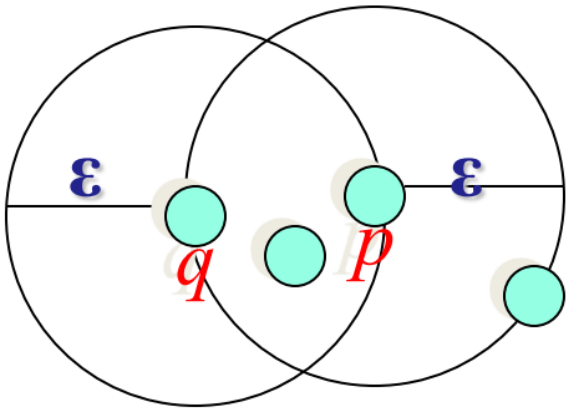


Density Definition

- ε -Neighborhood – Objects within a radius of ε from an object.

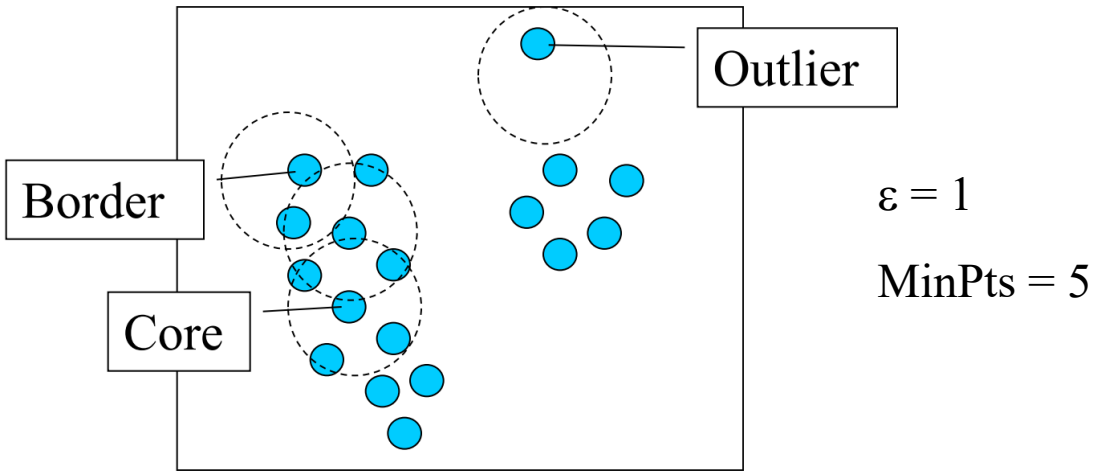
$$N_{\varepsilon}(p) : \{q \mid d(p,q) \leq \varepsilon\}$$

- "High density" -ε-Neighborhood of an object contains at least MinPts of objects.
  - ε-Neighborhood of p
  - ε-Neighborhood of q
  - Density of p is "high" (MinPts = 4)
  - Density of q is "low" (MinPts = 4)



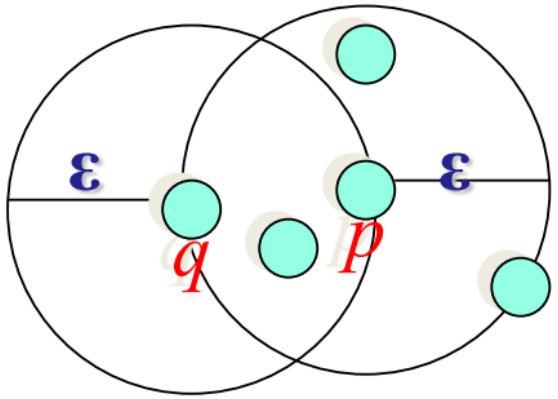
DBSCAN: Core, Border & Outlier

- According to ε-neighborhood of point p and MinPts, we classify all points into three types
  - **Core points**: Given a point p and a non-negative integer MinPts, if the size of N(p) is at least MinPts, then p is said to be a **core point**.
  - **Border points:** Given a point p, p is said to be a **border point** if it is not a core point but N(p) contains at least one core point.
  - **Noise points**： Given a point p, p is said to be a **noise point** if it is neither a core point nor a border point.



DBSCAN: Density-reachability

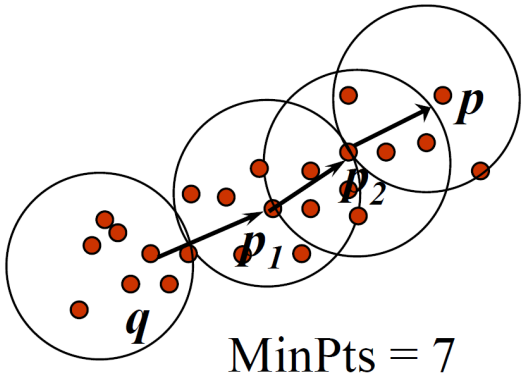
- Directly density-reachable
  - An object q is directly density-reachable from object p if p is a core object and q is in p's ε-neighborhood.
  - q is directly density-reachable from p
  - p is not directly density-reachable from q
  - Density-reachability is asymmetric



MinPts = 4

DBSCAN: Density-reachability

- Density-Reachable (directly and indirectly):
  - A point p is directly density-reachable from p2
  - p2 is directly density-reachable from p1
  - p1 is directly density-reachable from q
  - p <- p2 <- p1 <-q form a chain



- p is (indirectly) density-reachable from q
- q is not density-reachable from p

DBSCAN: The Algorithm

- Steps
  - Arbitrary select a point *p*
  - Retrieve all points density-reachable from *p* w.r.t. *Eps* and *MinPts*
  - If *p* is a core point, a cluster is formed
  - If *p* is a border point, no points are density-reachable from *p* and DBSCAN visits the next point of the database
  - Continue the process until all of the points have been processed

Example

- Consider the following 9 two-dimensional data points:

  x1(0,0), x2(1,0), x3(1,1), x4(2,2), x5(3,1), x6(3,0),

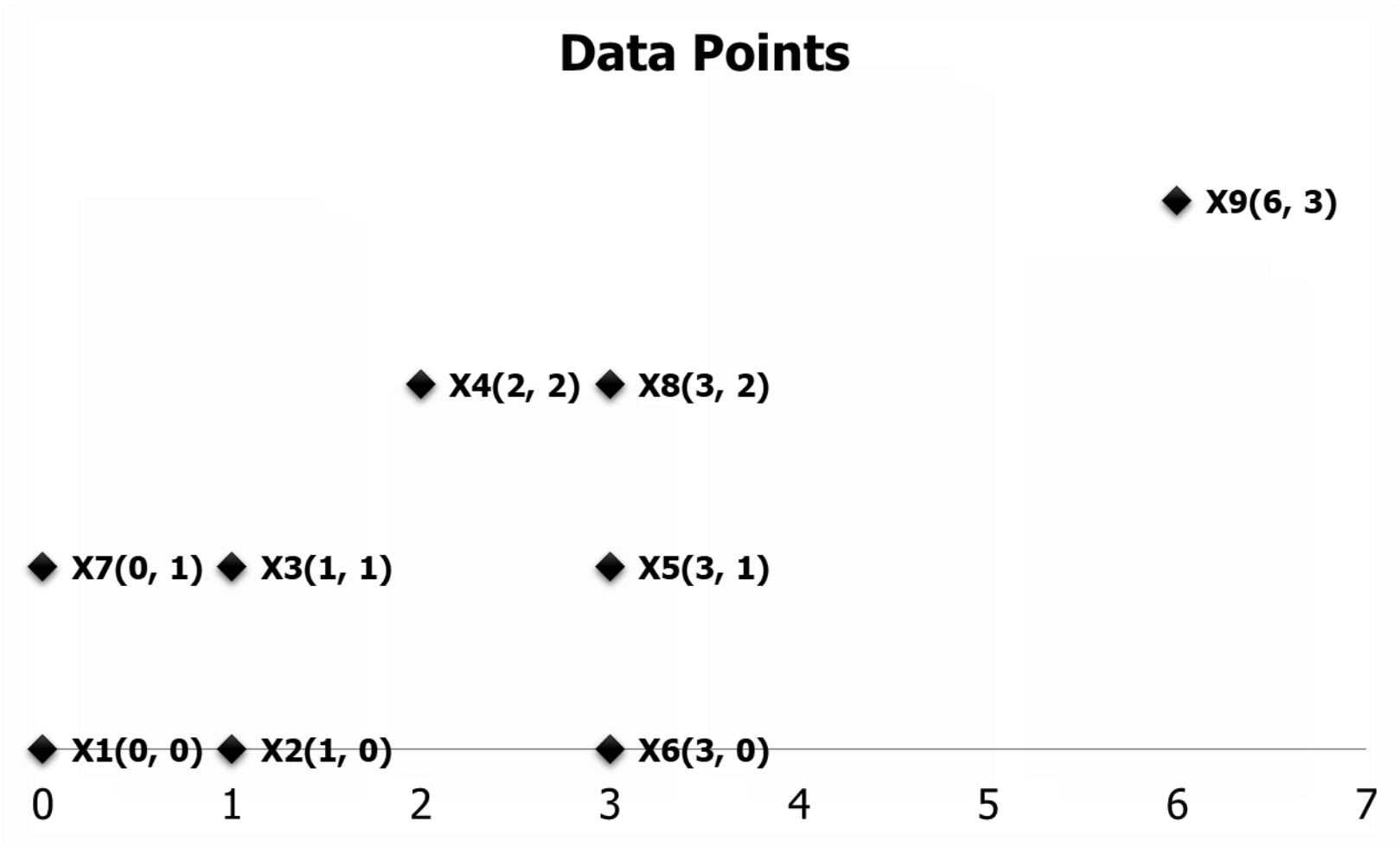
  x7(0,1), x8(3,2), x9(6,3)

Use the Euclidean Distance with ε =1 and MinPts = 3

(Eps is short for epsilon: ε)

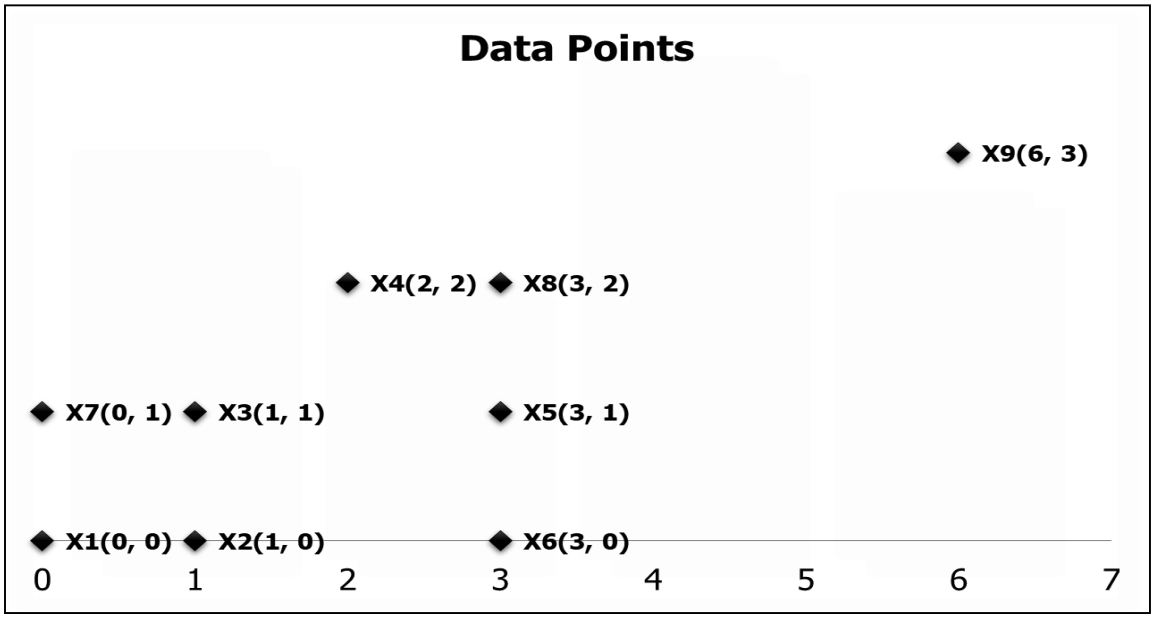
Find all core points, border points and noise points, and show the final clusters using DBCSAN algorithm.

Example


Data Points
X9(6, 3)
X4(2, 2)
X8(3, 2)
X7(0, 1)
X3(1, 1)
X5(3, 1)
X1(0, 0)
X2(1, 0)
X6(3, 0)
0
1
2
3
4
5
6
7

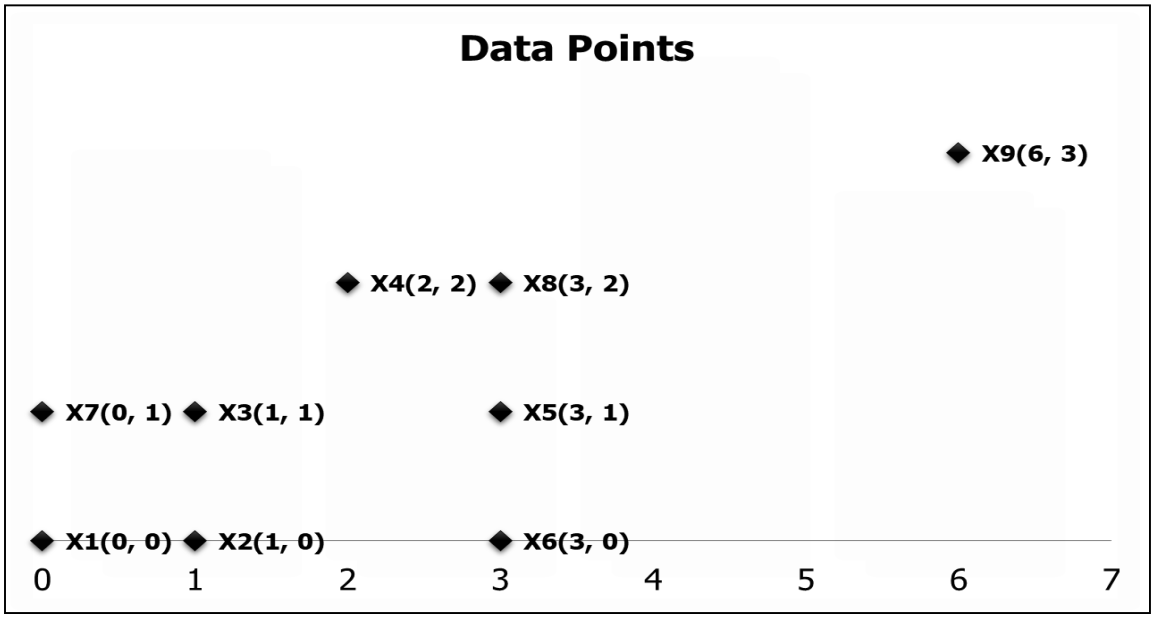
Calculate the N(p), ε-neighborhood of point p

- N(x1) = {x1, x2, x7}
- N(x2) = {x2, x1, x3}
- N(x3) = {x3, x2, x7}
- N(x4) = {x4, x8}
- N(x5) = {x5, x6, x8}
- N(x6) = {x6, x5}
- N(x7) = {x7, x1, x3}
- N(x8) = {x8, x4, x5}
- N(x9) = {x9}



Find all core points according to N(p)

- If the size of N(p) is at least MinPts, then p is said to be a **core point**.
- Here the given MinPts is 3, thus the size of N(p) is at least 3.
- We can find:
- N(x1) = {x1, x2, x7}
- N(x2) = {x2, x1, x3}
- N(x3) = {x3, x2, x7}
- N(x5) = {x5, x6, x8}
- N(x7) = {x7, x1, x3}
- N(x8) = {x8, x4, x5}



Find all core points according to N(p)

- Thus core points are:

  {x1, x2, x3, x5, x7, x8}

- Then according to the definition of border points: given a point p, p is said to be a **border point** if it is not a core point but N(p) contains at least one core point.

  N(x4) = {x4, x8}

  N(x6) = {x6, x5} ,

here **x8 and x5** are **core points**, So both x4 and x6 are border points.

Obviously, **x9** is a **noise** point.

Core points, Border points, Noise points

- **Core Points** are:

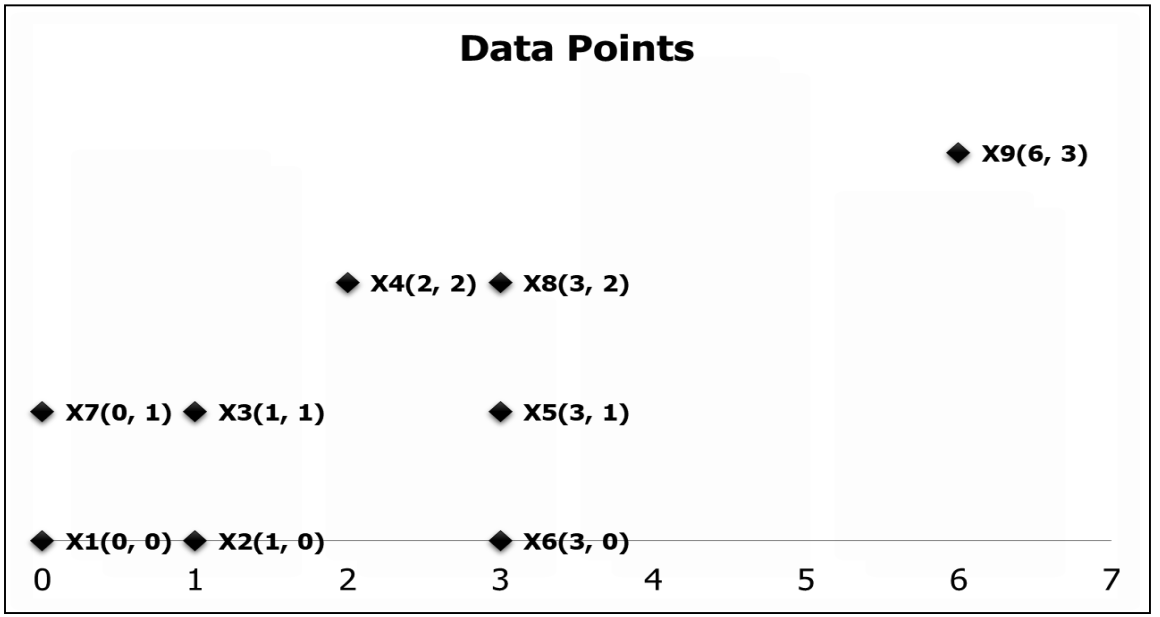
  {x1, x2, x3, x5, x7, x8}

- **Border Points**:

  {x4, x6}

- **Noise Points**:

  {x9}



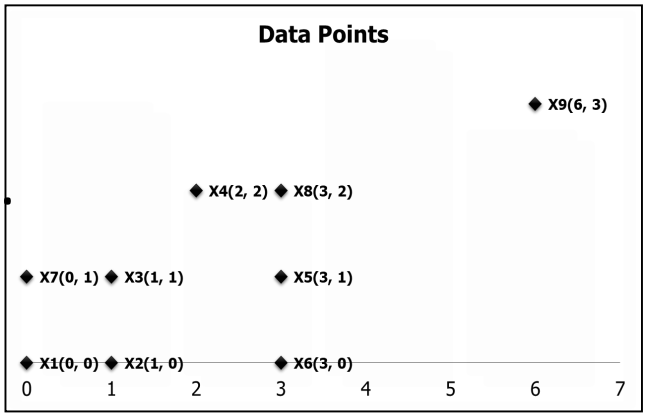
DBSCAN: The Algorithm

- Steps
  - Arbitrary select a point *p*
  - Retrieve all points density-reachable from *p* w.r.t. *Eps* and *MinPts*
  - If *p* is a core point, a cluster is formed
  - If *p* is a border point, no points are density-reachable from *p* and DBSCAN visits the next point of the database
  - Continue the process until all of the points have been processed

DBSCAN: Example step by step

- **Arbitrary** select a point *p, **now we choose x1***
- Retrieve all points density-reachable from x1*:*

  {x2, x3, x7}



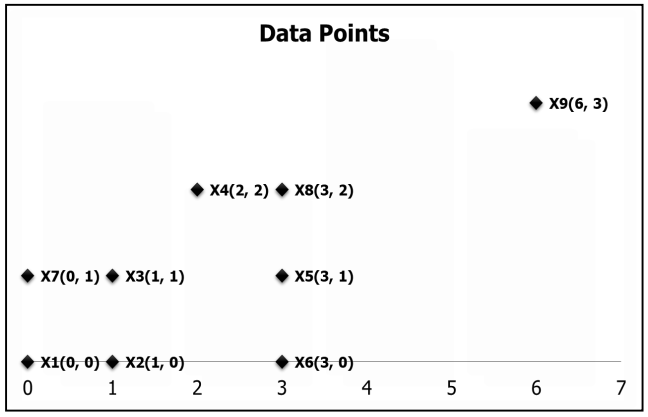
- Here x1 is a **core** point, a **cluster** is formed.
- So we have **Cluster_1: {x1, x2, x3, x7}**
- Next we choose **x5**,
- Retrieve all points density-reachable from x5*:*

  {x8, x4, x6}

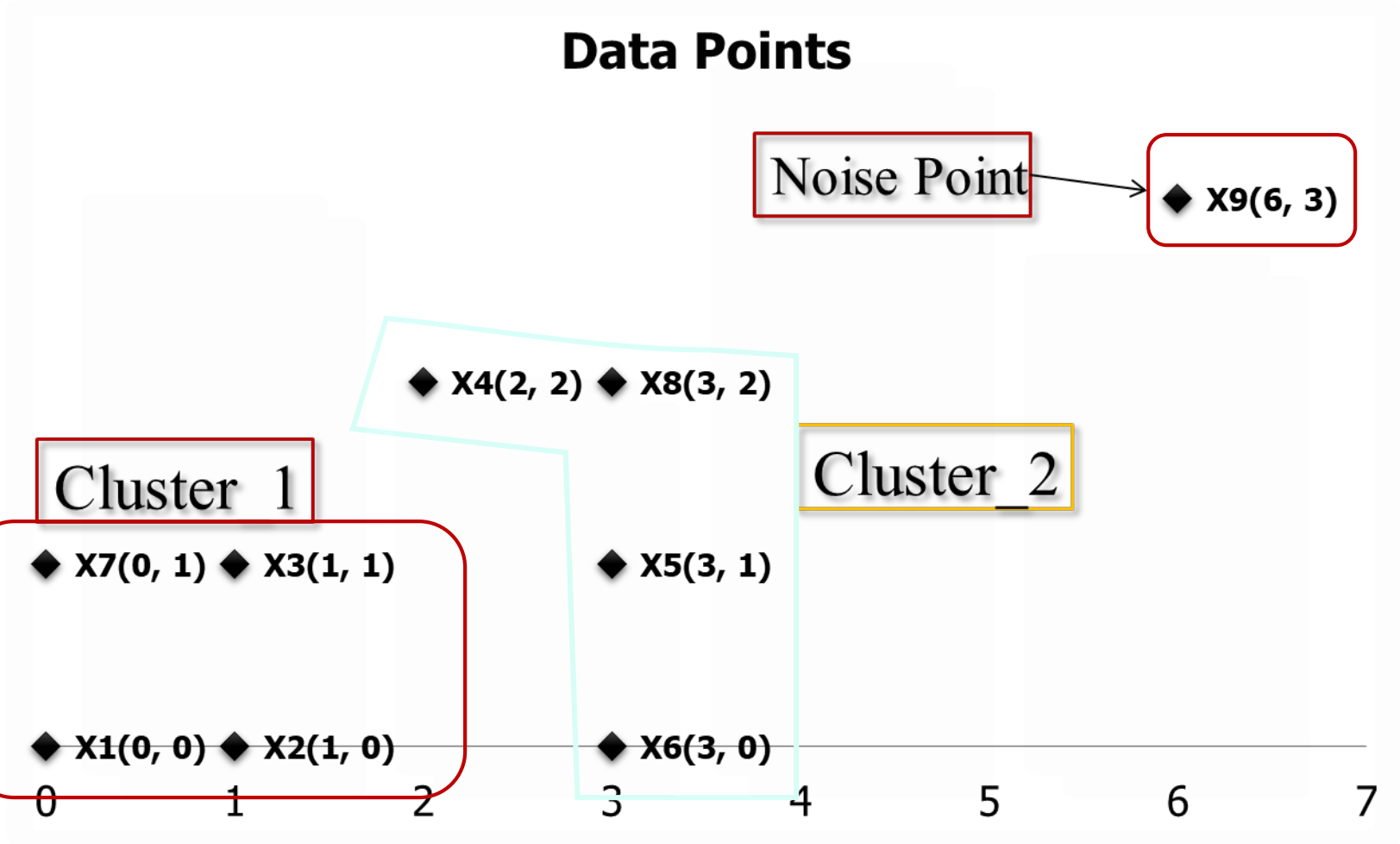
- Directly density-reachable
  - An object q is directly density-reachable from object p if p is a core object and q is in p's ε-neighborhood.

DBSCAN: Example step by step

- Here x5 is a **core** point, a **cluster** is formed.
- So we have **Cluster_2: {x5, x4, x8, x6}**
- Next we choose **x9**, x9 is a noise point, noise points do NOT belong to any clusters.
- Thus the algorithm stops here.



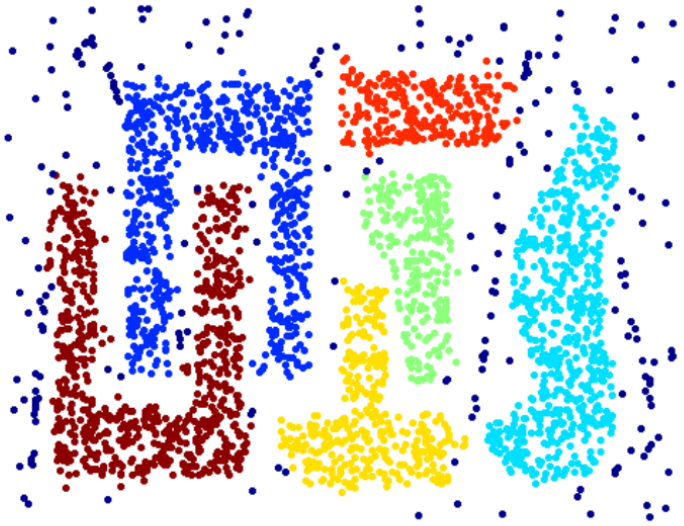
Final Clusters using DBSCAN



When DBSCAN Works Well

- Resistant to Noise
- Can handle clusters of different shapes and sizes

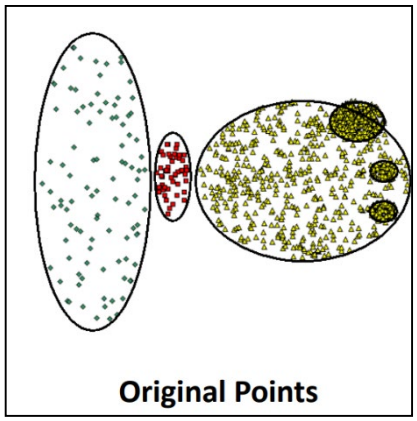
Original Points

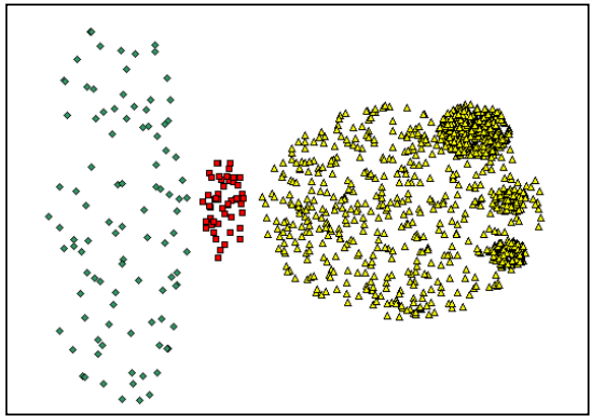


Clusters

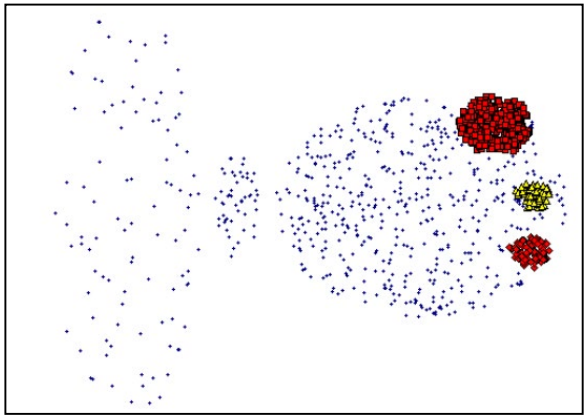
When DBSCAN Does NOT Work Well

- Cannot handle varying densities
- Sensitive to parameters—hard to determine the correct set of parameters





MinPts=4, Eps=9.92



MinPts=4, Eps=9.75

Summary

- The basic idea of density-based clustering
- The two important parameters and the definitions of neighborhood and density in DBSCAN
- Core, border and outlier points
- DBSCAN algorithm
- DBSCAN's pros and cons

Outline

- Hierarchical Clustering
- Clustering based on density
- Fuzzing clustering

Fuzzy Set and Fuzzy Cluster

- Clustering methods discussed so far
  - Every data object is assigned to exactly one cluster
- Some applications may need for fuzzy or soft cluster assignment
  - Ex. An e-game could belong to both entertainment and software
- Methods: fuzzy clusters and probabilistic model-based clusters
- Fuzzy cluster: A fuzzy set *S:* $F_S : X \rightarrow [0, 1]$ (value between 0-1)
- Example: Popularity of cameras is defined as a fuzzy mapping

Camera	Sales (units)
A	50
B	1320
C	860
D	270

$$\text{Pop}(o) = \begin{cases} 1 & \text{if } 1,000 \text{ or more units of } o \text{ are sold} \\ \frac{i}{1000} & \text{if } i \ (i < 1000) \text{ units of } o \text{ are sold} \end{cases}$$

- Then, *A*(0.05), *B*(1), *C*(0.86), *D*(0.27)

Hard (Crisp) vs Soft (Fuzzy) Clustering

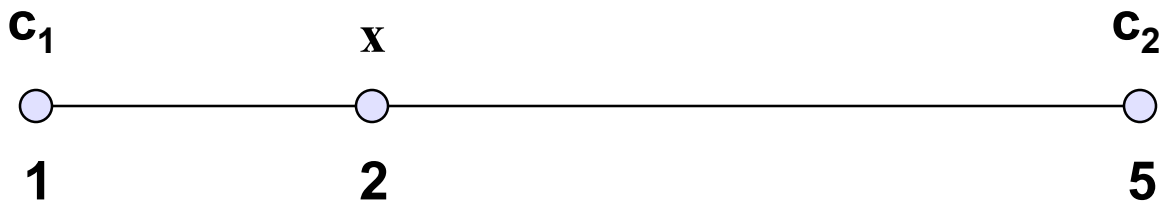
- Hard (Crisp) vs. Soft (Fuzzy) clustering
  - For soft clustering allow point to belong to more than one cluster
  - For K-means, generalize objective function

$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{m} w_{ij}\, dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2 \qquad \sum_{j=1}^{k} w_{ij} = 1$$

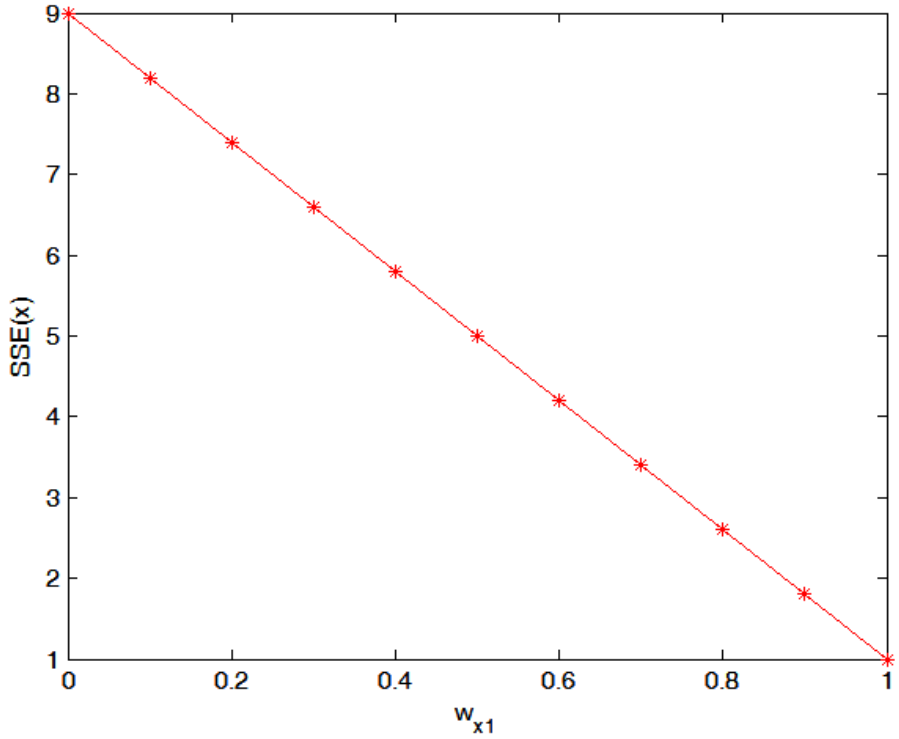
w_{ij} : weight with which object x_i belongs to cluster $\boldsymbol{c_j}$

  - To minimize SSE, repeat the following steps:
    - Fix $\boldsymbol{c_j}$ and determine w_{ij} (cluster assignment)
    - Fix w_{ij} and recompute $\boldsymbol{c_j}$
  - Hard clustering: $w_{ij} \in \{0,1\}$

Soft (Fuzzy) Clustering: Estimating Weights



$$SSE(x) = w_{x1}(2-1)^2 + w_{x2}(5-2)^2$$
$$= w_{x1} + 9w_{x2}$$



SSE(x) is minimized when w_{x1} = 1, w_{x2} = 0

Fuzzy C-means

- Objective function

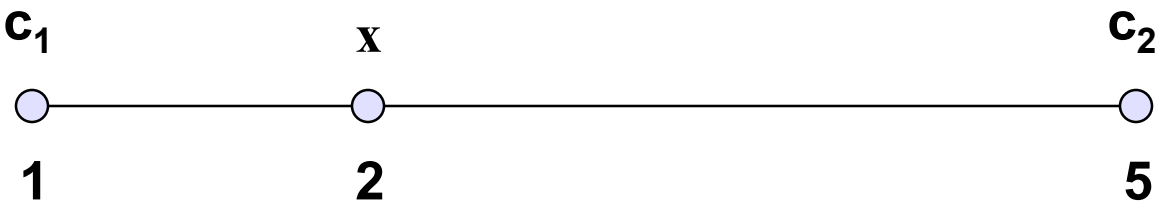
p: fuzzifier (p > 1)

$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{m} w_{ij}^{p} \, dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2$$

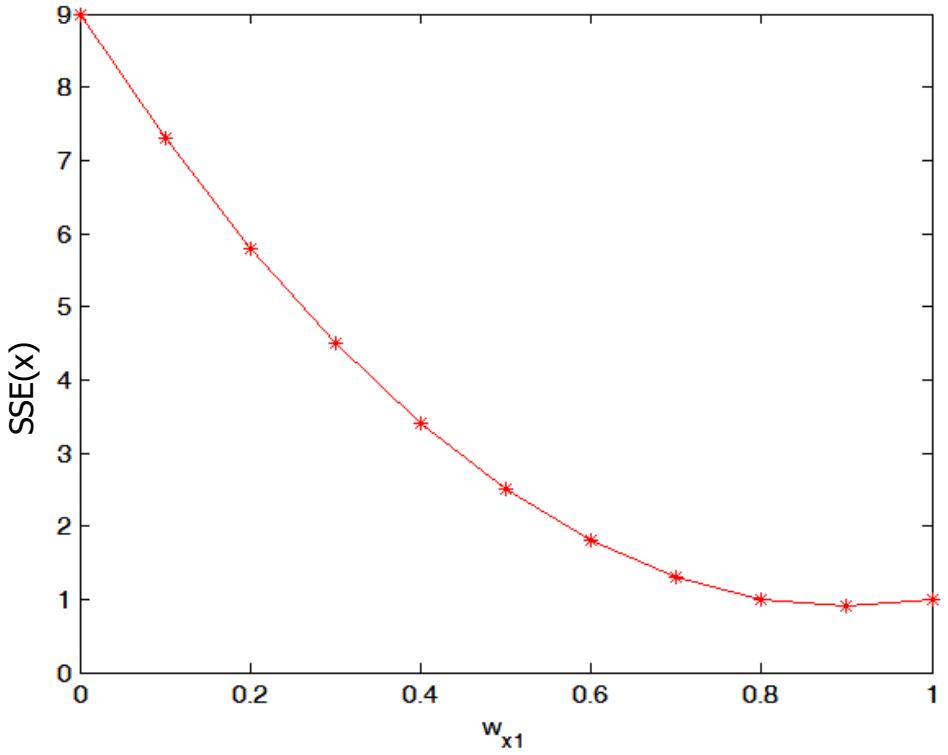
  - w_{ij}: weight with which object $\boldsymbol{x}_i$ belongs to cluster $\boldsymbol{c}_j$
  - p: a power for the weight not a superscript and controls how "fuzzy" the clustering is $\sum_{j=1}^{k} w_{ij} = 1$

- To minimize objective function, repeat the following:
  - Fix $\boldsymbol{c}_j$ and determine w_{ij}
  - Fix w_{ij} and recompute $\boldsymbol{c}$
- Fuzzy c-means clustering: $w_{ij} \in [0,1]$

Fuzzy C-means



$$SSE(x) = w_{x1}^2(2-1)^2 + w_{x2}^2(5-2)^2$$
$$= w_{x1}^2 + 9w_{x2}^2$$



SSE(x) is minimized when w_{x1} = 0.9, w_{x2} = 0.1

Fuzzy C-means

- Objective function:

$$SSE = \sum_{j=1}^{k}\sum_{i=1}^{m} w_{ij}^{p}\, dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2 \qquad \sum_{j=1}^{k} w_{ij} = 1$$

- Initialization: choose the weights w_{ij} randomly
- Repeat:
  - Update centroids: $\boldsymbol{c_j} = \sum_{i=1}^{m} w_{ij}^{p}\boldsymbol{x}_i \,/\, \sum_{i=1}^{m} w_{ij}^{p}$
  - Update weights:

$$w_{ij} = (1/dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2)^{\frac{1}{p-1}} / \sum_{j=1}^{k} (1/dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2)^{\frac{1}{p-1}}$$

Fuzzy (Soft) Clustering

- Example: Let cluster features be
  - C_1 :“digital camera” and “lens”
  - C_2: “computer“
- Fuzzy clustering
  - k fuzzy clusters C_1, …,C_k ,represented as a partition matrix M = [w_{ij}]
  - For each object o_i and cluster C_j, $0 \le w_{ij} \le 1$ (fuzzy set)
  - For each object o_i $\sum_{i=1}^{k} w_{ij} = 1$, participation in the clustering
  - For each cluster C_j , $0 < \sum_{i=1}^{n} w_{ij} < n$ ensures there is no empty cluster

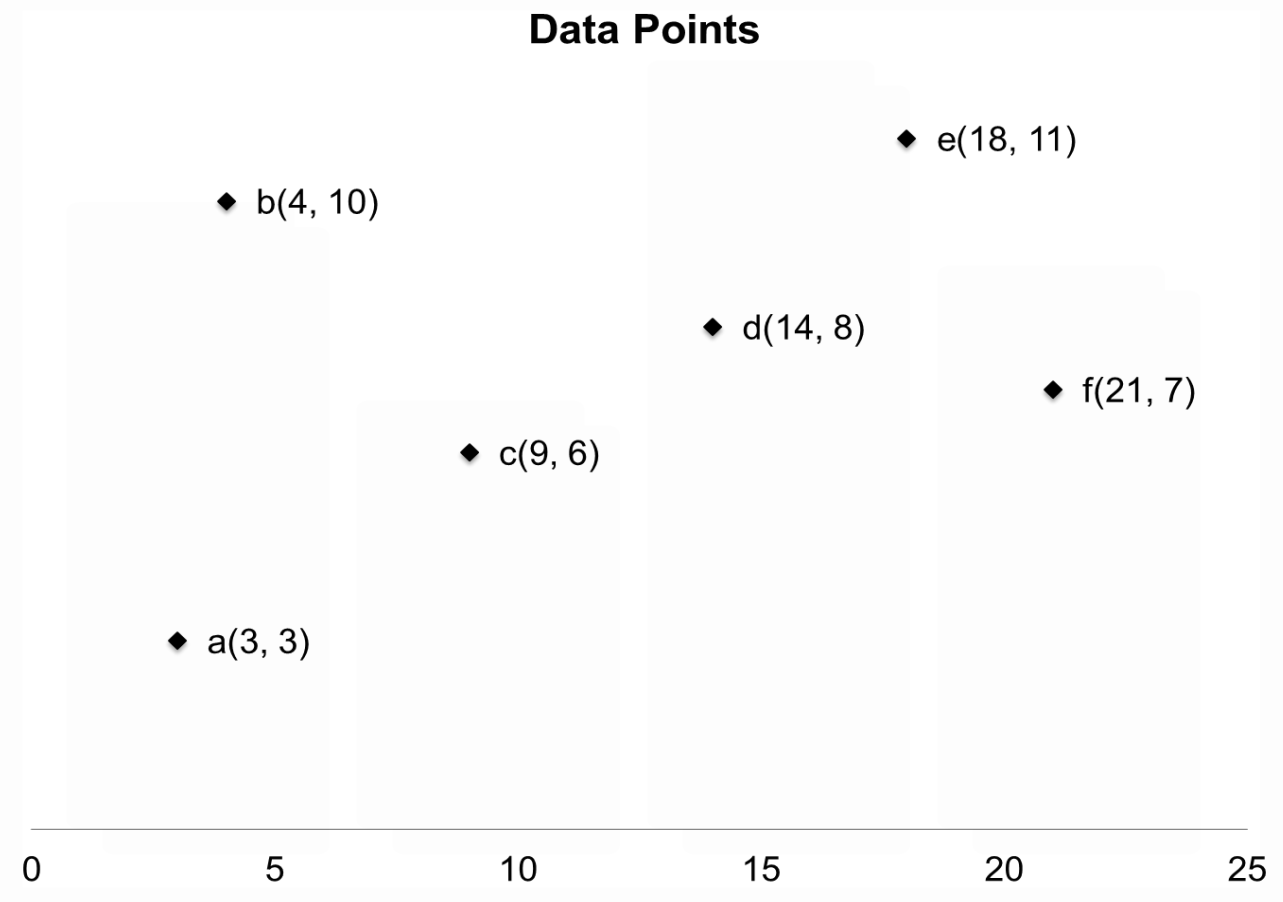
Review-id	Keywords
R_1	digital camera, lens
R_2	digital camera
R_3	lens
R_4	digital camera, lens, computer
R_5	computer, CPU
R_6	computer, computer game

$$M = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ \frac{2}{3} & \frac{1}{3} \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

The EM (Expectation Maximization) Algorithm

- The k-means algorithm has two steps at each iteration:
  - **Expectation Step** (E-step): Given the current cluster centers, each object is assigned to the cluster whose center is closest to the object: An object is *expected to belong to the closest cluster*
  - **Maximization Step** (M-step): Given the cluster assignment, for each cluster, the algorithm *adjusts the center* so that *the sum of distance* from the objects assigned to this cluster and the new center is minimized
- **The (EM) algorithm:** A framework to approach maximum likelihood or maximum a posteriori estimates of parameters in statistical models.
  - **E-step** assigns objects to clusters according to the current fuzzy clustering or parameters of probabilistic clusters
  - **M-step** finds the new clustering or parameters that maximize the sum of squared error (SSE) or the expected likelihood

Fuzzy Clustering Using the EM



Fuzzy Clustering Using the EM

- Initially, let c_1= a and c_2 = b
- 1st E-step: assign objects to clusters: c_1 and c_2
- Calculate the weight for each object for each cluster: w_{ij} means the weight of object i in cluster j. **Below is a specific formula for this question only since there are only two clusters here.**
- w_{i1} **means the weight of object i in cluster 1 (c_1).**

$$w_{i1} = \frac{\frac{1}{dist(o_i, c_1)^2}}{\frac{1}{dist(o_i, c_1)^2} + \frac{1}{dist(o_i, c_2)^2}} = \frac{dist(o_i, c_2)^2}{dist(o_i, c_2)^2 + dist(o_i, c_1)^2}$$

Repeat:

- Update centroids: $\boldsymbol{c_j} = \sum_{i=1}^{m} w_{ij}^p \boldsymbol{x}_i \,/ \sum_{i=1}^{m} w_{ij}^p$
- Update weights:

$$w_{ij} = (1/dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2)^{\frac{1}{p-1}} / \sum_{j=1}^{k} (1/dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2)^{\frac{1}{p-1}}$$

Fuzzy Clustering Using the EM

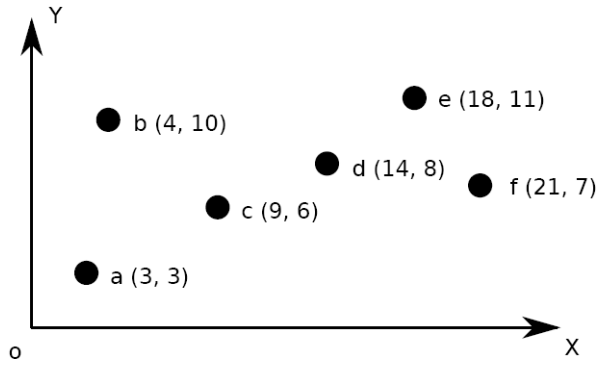
- Calculate w_{i2} : **the weight of object i in cluster 2 (c_2).**

$$w_{i2} = \frac{\frac{1}{dist(o_i, c_2)^2}}{\frac{1}{dist(o_i, c_1)^2} + \frac{1}{dist(o_i, c_2)^2}} = \frac{dist(o_i, c_1)^2}{dist(o_i, c_2)^2 + dist(o_i, c_1)^2}$$

*For **this case particularly**, we can use a simple way to calculate* w_{i2}

$\boldsymbol{w_{i2} = 1 - w_{i1}}$.

This is because there are only two clusters in this case, and it also obeys the rule $\sum_1^k \boldsymbol{w_{ij}} = \mathbf{1}$



E-step in the 1st Iteration

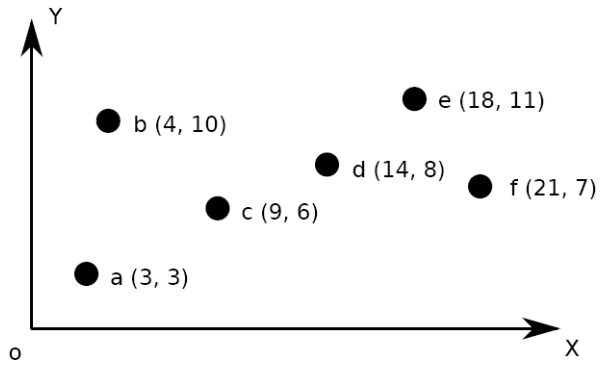
- With this formula, we can calculate the weight for each object in c_1
- **Next will calculate object c , because a is c_1 and b is c_2, so**

$\boldsymbol{w_{a1} = 1, w_{a2} = 0, w_{b1} = 0, w_{b2} = 1,}$

$$\boldsymbol{w_{c1}} = \frac{dist(c, c_2)^2}{dist(c, c_2)^2 + dist(c, c_1)^2} = \frac{(9-4)^2 + (6-10)^2}{(9-4)^2 + (6-10)^2 + (9-3)^2 + (6-3)^2}$$

$$= \frac{41}{41+45} = \mathbf{0.48}$$

Then use the simple method to calculate $\boldsymbol{w_{c2}}$ = 1- 0.48 = **0.52**



E-step in the 1st Iteration

- **Take point d for another example:**

$$w_{d1}=\frac{dist(d,c_2)^2}{dist(d,c_2)^2+dist(d,c_1)^2}=\frac{(14-4)^2+(8-10)^2}{(14-4)^2+(8-10)^2+(14-3)^2+(8-3)^2}$$

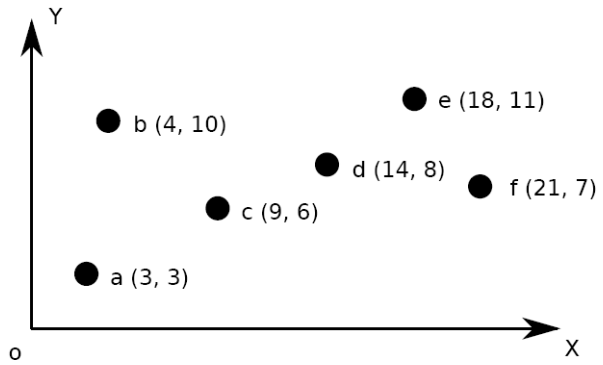
$$=\frac{104}{104+146}=\mathbf{0.42}$$

Then use the simple method to calculate w_{d2} = 1- 0.42 = **0.58**

Similarly, we can calculate the other weights:

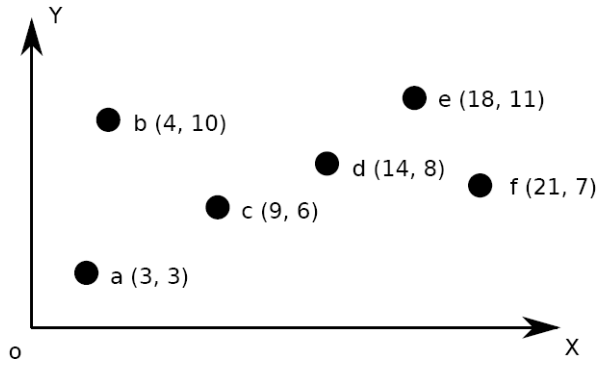
w_{e1} = **0.41**, w_{e2} = 1- 0.41 = **0.59**

w_{f1} = **0.47**, w_{f2} = 1- 0.47 = **0.53**



Partition Matrix in the 1st Iteration

- **Now we can draw the partition Matrix:**
- $$\mathbf{M} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0.48 & 0.52 \\ 0.42 & 0.58 \\ 0.41 & 0.59 \\ 0.47 & 0.53 \end{bmatrix}$$



- Each row in Partition Matrix represents an Object(a Point in this case)
- Each column represents a Cluster.

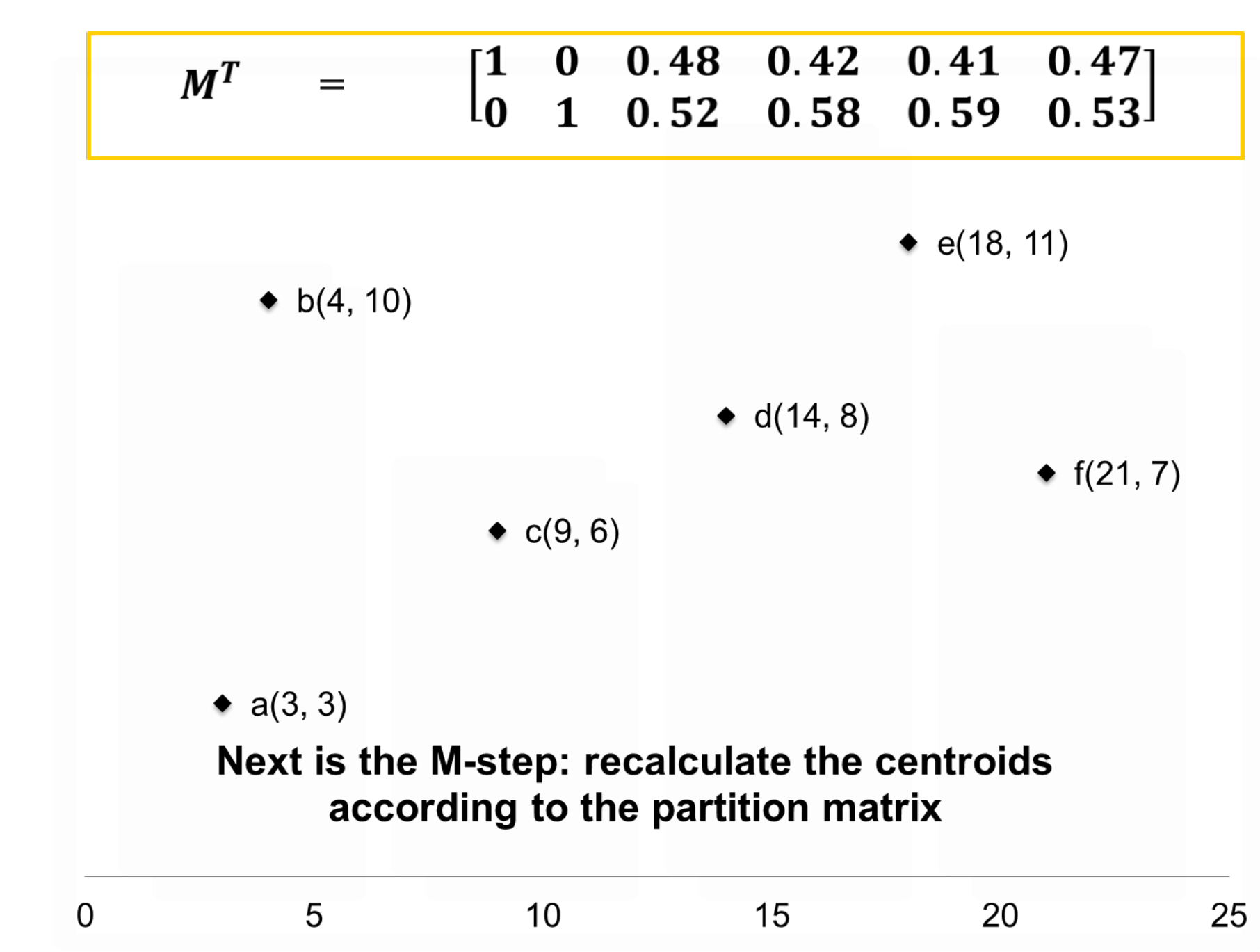
Transposition Matrix in the 1st Iteration

- M^T =

	a	b	c	d	e	f	
	1	0	0.48	0.42	0.41	0.47	c_1
	0	1	0.52	0.58	0.59	0.53	c_2

- For the **transposition** of the Matrix:
- Each **column** in Transposition Matrix represents an **Object**(a Point in this case)
- Each **row** represents a **Cluster**.

$$M^T = \begin{bmatrix} 1 & 0 & 0.48 & 0.42 & 0.41 & 0.47 \\ 0 & 1 & 0.52 & 0.58 & 0.59 & 0.53 \end{bmatrix}$$



M-step in the 1st Iteration

- 1st M-step: **recalculate the centroids** according to the partition matrix.

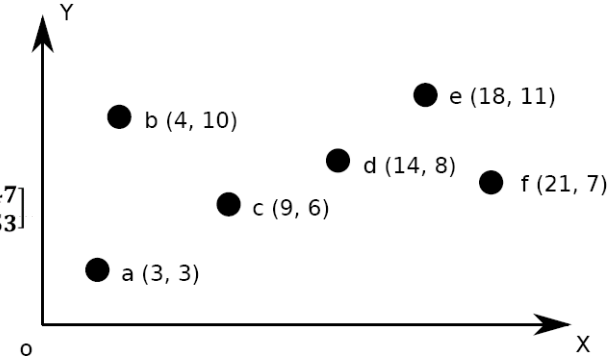
- $c_j = \frac{\sum_{each\ point\ o}(w^2_{o,c_j} * o)}{\sum_{each\ point\ o} w^2_{o,c_j}}$, **for example, calculate c_1 :**

- $$c_1 = \left(\frac{\sum_{each\ point\ o}(w^2_{o,c_1} * o_x)}{\sum_{each\ point\ o} w^2_{o,c_1}}, \frac{\sum_{each\ point\ o}(w^2_{o,c_1} * o_y)}{\sum_{each\ point\ o} w^2_{o,c_1}}\right)$$

$$c_1 = \left(\frac{1^2*3+0^2*4+0.48^2*9+0.42^2*14+0.41^2*18+0.47^2*21}{1^2+0^2+0.48^2+0.42^2+0.41^2+0.47^2}, \frac{1^2*3+0^2*10+0.48^2*6+0.42^2*8+0.41^2*11+0.47^2*7}{1^2+0^2+0.48^2+0.42^2+0.41^2+0.47^2}\right)$$

$$= (8.47, 5.12)$$

$$M^T = \begin{bmatrix} 1 & 0 & 0.48 & 0.42 & 0.41 & 0.47 \\ 0 & 1 & 0.52 & 0.58 & 0.59 & 0.53 \end{bmatrix}$$



Repeat:

- Update centroids: $c_j = \sum_{i=1}^{m} w_{ij}^p x_i / \sum_{i=1}^{m} w_{ij}^p$
- Update weights:

$$w_{ij} = (1/dist(x_i, c_j)^2)^{\frac{1}{p-1}} / \sum_{j=1}^{k} (1/dist(x_i, c_j)^2)^{\frac{1}{p-1}}$$

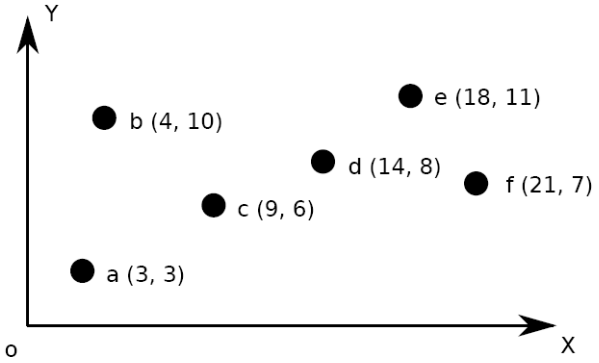
M-step in the 1st Iteration

- 1st M-step: **recalculate the centroids for cluster 2.**

- $$c_2 = \left(\frac{\sum_{each\ point\ o}(w^2_{o,c_2} * o_x)}{\sum_{each\ point\ o} w^2_{o,c_2}}, \frac{\sum_{each\ point\ o}(w^2_{o,c_2} * o_y)}{\sum_{each\ point\ o} w^2_{o,c_2}}\right)$$

$$c_2 = \left(\frac{0^2*3+1^2*4+0.52^2*9+0.58^2*14+0.59^2*18+0.53^2*21}{0^2+1^2+0.52^2+0.58^2+0.59^2+0.53^2}, \frac{0^2*3+1^2*10+0.52^2*6+0.58^2*8+0.59^2*11+0.53^2*7}{0^2+1^2+0.52^2+0.58^2+0.59^2+0.53^2}\right)$$

$$= \mathbf{(10.42, 8.99)}$$



The 1st Iteration Result

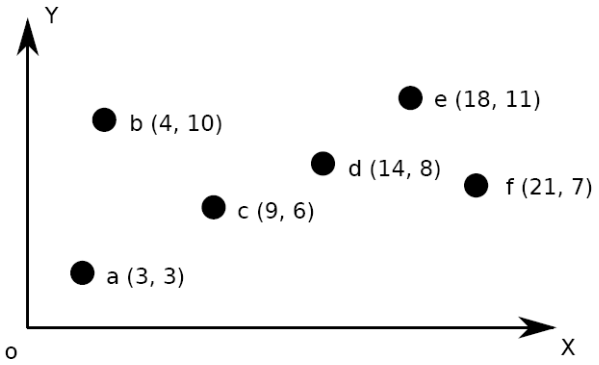
Iteration	E-Step	M-Step
1	$M^T = \begin{bmatrix} 1 & 0 & 0.48 & 0.42 & 0.41 & 0.47 \\ 0 & 1 & 0.52 & 0.58 & 0.59 & 0.53 \end{bmatrix}$	$c_1 = (8.47, 5.12)$ $c_2 = (10.42, 8.99)$

Now the first iteration is over, we should **repeat** the same process.

Go to the 2nd E-step: assign objects to new clusters: c_1 and c_2

Now is E-step in the 2nd Iteration

- New cluster centers: $\boldsymbol{c_1 = (8.47, 5.12)}$ and $\boldsymbol{c_2 = (10.42, 8.99)}$
- **2nd E-step**: assign objects to new clusters: c_1 and c_2
- Calculate the weight for each object for each cluster: w_{ij}
- For example, calculate weight of **a**:
- $\boldsymbol{w_{a1}} = \frac{dist(a, c_2)^2}{dist(a, c_2)^2 + dist(a, c_1)^2} = \frac{(3-10.42)^2+(3-8.99)^2}{(3-10.42)^2+(3-8.99)^2+(3-8.47)^2+(3-5.12)^2} = \frac{90.9365}{90.9365+34.4153} = \frac{90.9365}{125.3518} = \mathbf{0.73}$
- $\boldsymbol{w_{a2}} = 1 - w_{a1} = 1 - 0.73 = \mathbf{0.27}$
- **Similarly, we can calculate the other points' weight.**

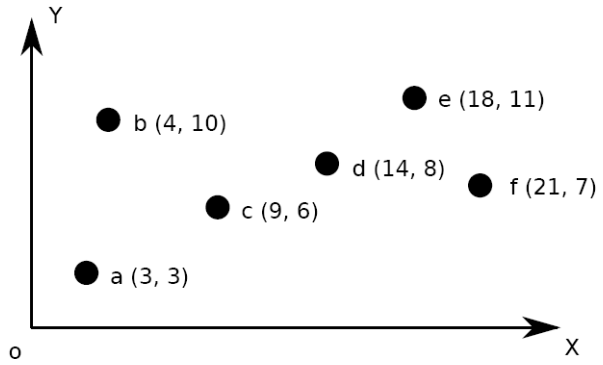


M-step in the 2nd Iteration

- After all weights are calculated, we now get the Matrix again.
- $M^T = \begin{bmatrix} 0.73 & 0.49 & 0.91 & 0.26 & 0.33 & 0.42 \\ 0.27 & 0.51 & 0.09 & 0.74 & 0.67 & 0.58 \end{bmatrix}$
- **The 2nd E-step is over, continue the 2nd M-step:**
- **For example, calculate new c_1 :**
- $$c_1 = \left(\frac{\sum_{each\ point\ o}(w^2_{o,c_1} * o_x)}{\sum_{each\ point\ o} w^2_{o,c_1}}, \frac{\sum_{each\ point\ o}(w^2_{o,c_1} * o_y)}{\sum_{each\ point\ o} w^2_{o,c_1}}\right)$$

$$c_1 = \left(\frac{0.73^2*3+0.49^2*4+0.91^2*9+0.26^2*14+0.33^2*18+0.42^2*21}{0.73^2+0.49^2+0.91^2+0.26^2+0.33^2+0.42^2}, \frac{0.73^2*3+0.49^2*10+0.91^2*6+0.26^2*8+0.33^2*11+0.42^2*7}{0.73^2+0.49^2+0.91^2+0.26^2+0.33^2+0.42^2}\right)$$

$$= (8.51, 6.11)$$



M-step in the 2nd Iteration

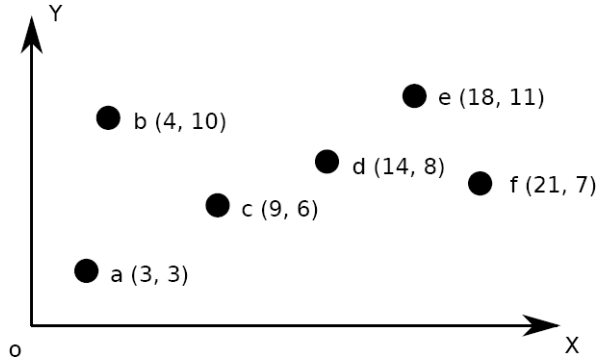
- 2nd M-step: **recalculate the centroids for cluster 2.**

- $$c_2 = \left(\frac{\sum_{each\ point\ o}(w^2_{o,c_2} * o_x)}{\sum_{each\ point\ o} w^2_{o,c_2}}, \frac{\sum_{each\ point\ o}(w^2_{o,c_2} * o_y)}{\sum_{each\ point\ o} w^2_{o,c_2}}\right)$$

$$c_2 = \left(\frac{0.27^2*3+0.51^2*4+0.09^2*9+0.74^2*14+0.67^2*18+0.58^2*21}{0.27^2+0.51^2+0.09^2+0.74^2+0.67^2+0.58^2}, \frac{0^2*3+1^2*10+0.52^2*6+0.58^2*8+0.59^2*11+0.53^2*7}{0.27^2+0.51^2+0.09^2+0.74^2+0.67^2+0.58^2}\right)$$

$$= (14.42, 8.69)$$

$$M^T = \begin{bmatrix} 0.73 & 0.49 & 0.91 & 0.26 & 0.33 & 0.42 \\ 0.27 & 0.51 & 0.09 & 0.74 & 0.67 & 0.58 \end{bmatrix}$$



The 2nd Iteration Result

Iteration	E-Step	M-Step
2	$M^T = \begin{bmatrix} 0.73 & 0.49 & 0.91 & 0.26 & 0.33 & 0.42 \\ 0.27 & 0.51 & 0.09 & 0.74 & 0.67 & 0.58 \end{bmatrix}$	$c_1 = (8.51, 6.11)$ $c_2 = (14.42, 8.69)$

Now the second iteration is over, but the centers do not converge, so we need to **repeat** the same process.

Go to the 3rd E-step: assign objects to new clusters: c_1 and c_2

The 3rd Iteration Result

Iteration	E-Step	M-Step
3	$M^T = \begin{bmatrix} 0.80 & 0.76 & 0.99 & 0.02 & 0.14 & 0.23 \\ 0.20 & 0.24 & 0.01 & 0.98 & 0.86 & 0.77 \end{bmatrix}$	$c_1 = (6.40, 6.24)$ $c_2 = (16.55, 8.64)$

Now the third iteration is over.
But it looks like the result is still not good because the cluster centers do not converge.
So we need to continue repeating again, **until the cluster centers converge or the change is small enough.**

Summary

- Provides degree of cluster membership
- Similar strengths and weakness as K-means
- More computational expensive