Clustering

- **Review of K-means algorithm**

Algorithm: *k*-means. The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects from *D* as the initial cluster centers;
(2) **repeat**
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) update the cluster means, that is, calculate the mean value of the objects for each cluster;
(5) **until** no change;

Chair of Process and Data Science

Clustering

- **An example implementation of k-means algorithm**

(17.936710, 15.784810) ← Cluster 1

Centroid of cluster 1

Initialization

(4.693684, 12.342170) ← Cluster 2

Centroid of cluster 2

	x1	x2
0	2.072345	-3.241693
$\boxed{1}$	17.936710	15.784810
2	1.083576	7.319176
3	11.120670	14.406780
4	23.711550	2.557729
5	24.169930	32.024780
6	21.665780	4.892855
$\boxed{7}$	4.693684	12.342170
8	19.211910	-1.121366
9	4.230391	-4.441536

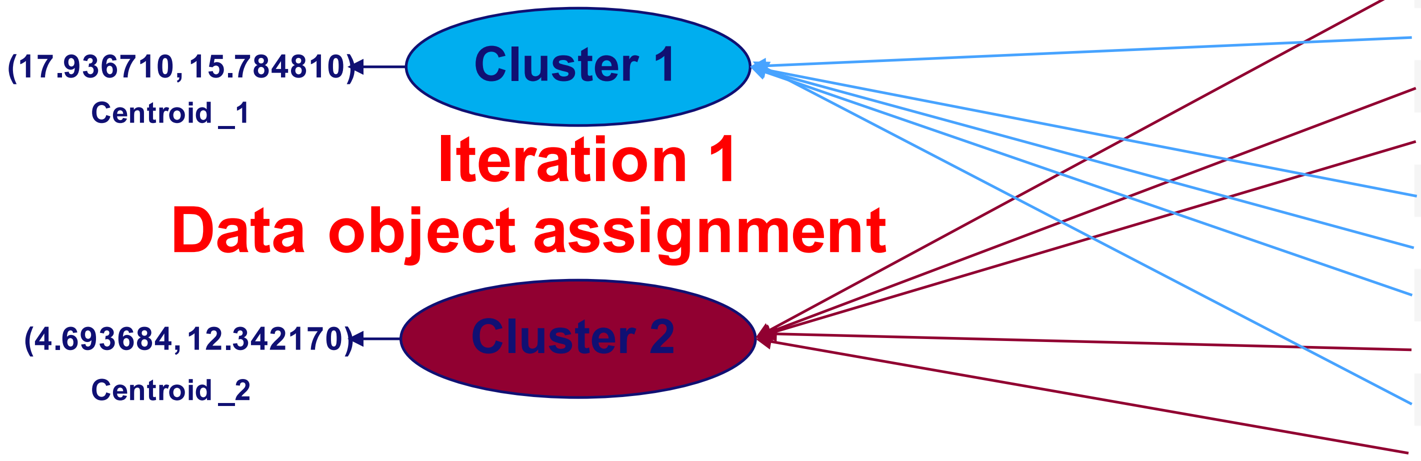
Example data set

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{ip} - x_{jp})^2}$$

Euclidean distance

Clustering

- **An example implementation of k-means algorithm**



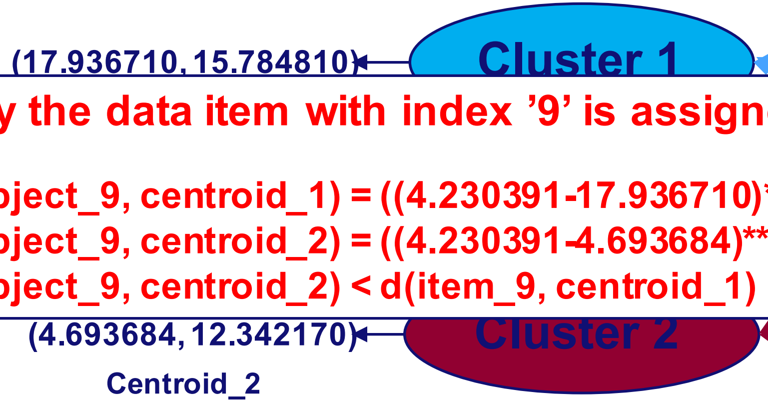
	x1	x2
0	2.072345	-3.241693
1	17.936710	15.784810
2	1.083576	7.319176
3	11.120670	14.406780
4	23.711550	2.557729
5	24.169930	32.024780
6	21.665780	4.892855
7	4.693684	12.342170
8	19.211910	-1.121366
9	4.230391	-4.441536

Example data set

Chair of Process and Data Science

Clustering

- **An example implementation of k-means algorithm**



Why the data item with index '9' is assigned to cluster 2 ???

d(object_9, centroid_1) = ((4.230391-17.936710)2 + (-4.441536-15.784810)**2)**0.5 = 24.43293378**
d(object_9, centroid_2) = ((4.230391-4.693684)2 + (-4.441536-12.342170)**2)**0.5 = 16.79009909**
d(object_9, centroid_2) < d(item_9, centroid_1) => item with index 9 is assigned to cluster 2

	x1	x2
0	2.072345	-3.241693
1	17.936710	15.784810
	4.693684	12.342170
	9.211910	-1.121366
9	4.230391	-4.441536

Example data set

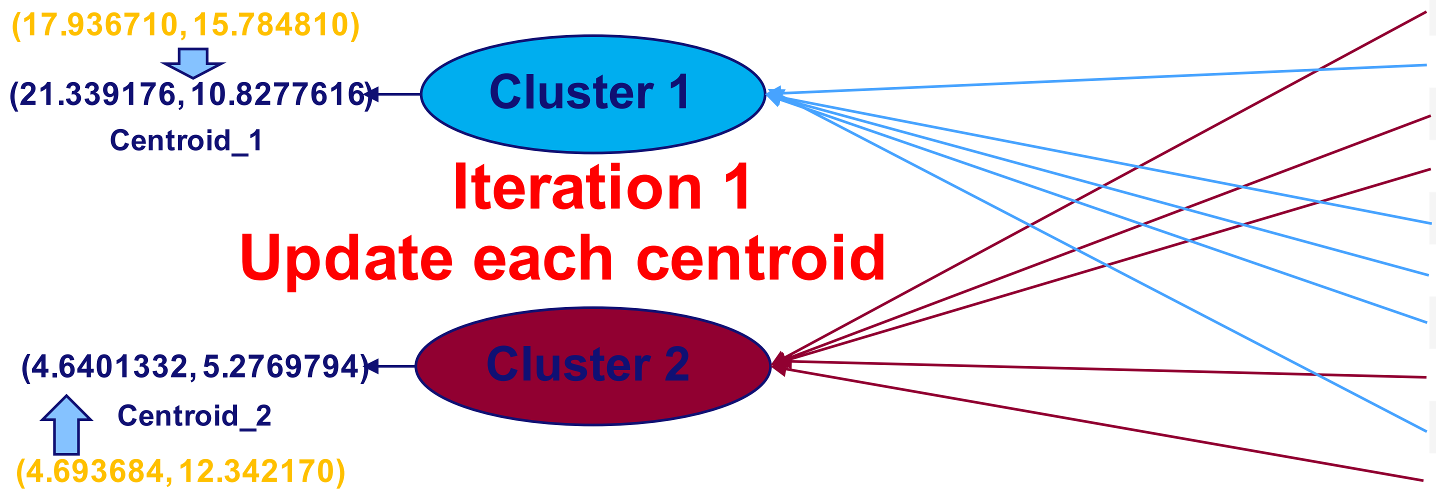
$$d(i, j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{ip} - x_{jp})^2}$$

Euclidean distance

Chair of Process and Data Science

Clustering

- **An example implementation of k-means algorithm**

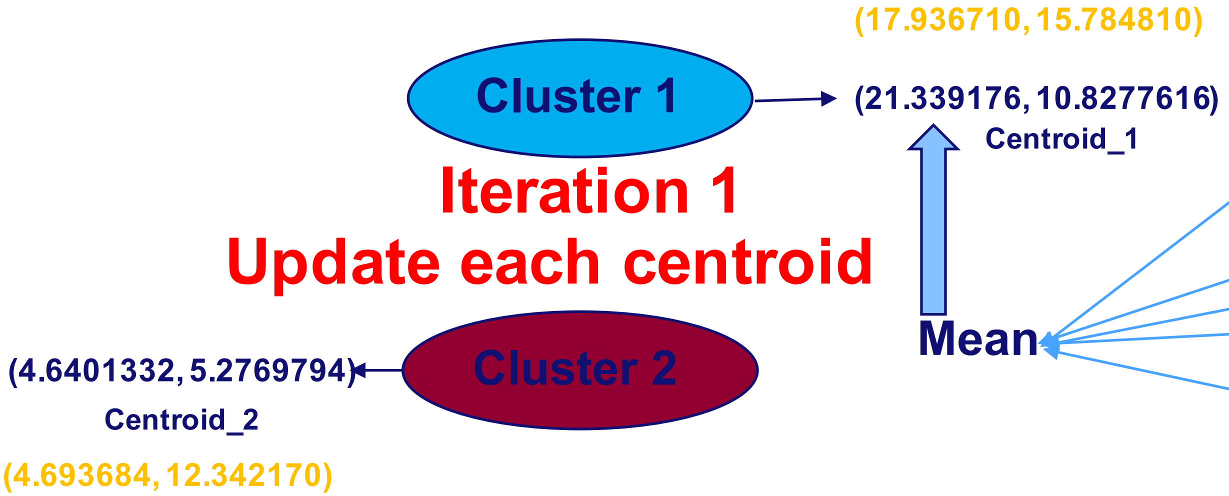


	x1	x2
0	2.072345	-3.241693
1	17.936710	15.784810
2	1.083576	7.319176
3	11.120670	14.406780
4	23.711550	2.557729
5	24.169930	32.024780
6	21.665780	4.892855
7	4.693684	12.342170
8	19.211910	-1.121366
9	4.230391	-4.441536

Example data set

Clustering

- An example implementation of k-means algorithm



	x1	x2
0	2.072345	-3.241693
1	17.936710	15.784810
2	1.083576	7.319176
3	11.120670	14.406780
4	23.711550	2.557729
5	24.169930	32.024780
6	21.665780	4.892855
7	4.693684	12.342170
8	19.211910	-1.121366
9	4.230391	-4.441536

Example data set

Chair of Process and Data Science

Clustering

- **An example implementation of k-means algorithm**

(17.936710, 15.784810)

Cluster 1 → (21.339176, 10.8277616)
Centroid_1

Iteration 1
Update each centroid

Mean

(4.6401332, 5.2769794) ← **Cluster 2**
Centroid_2

(4.693684, 12.342170)

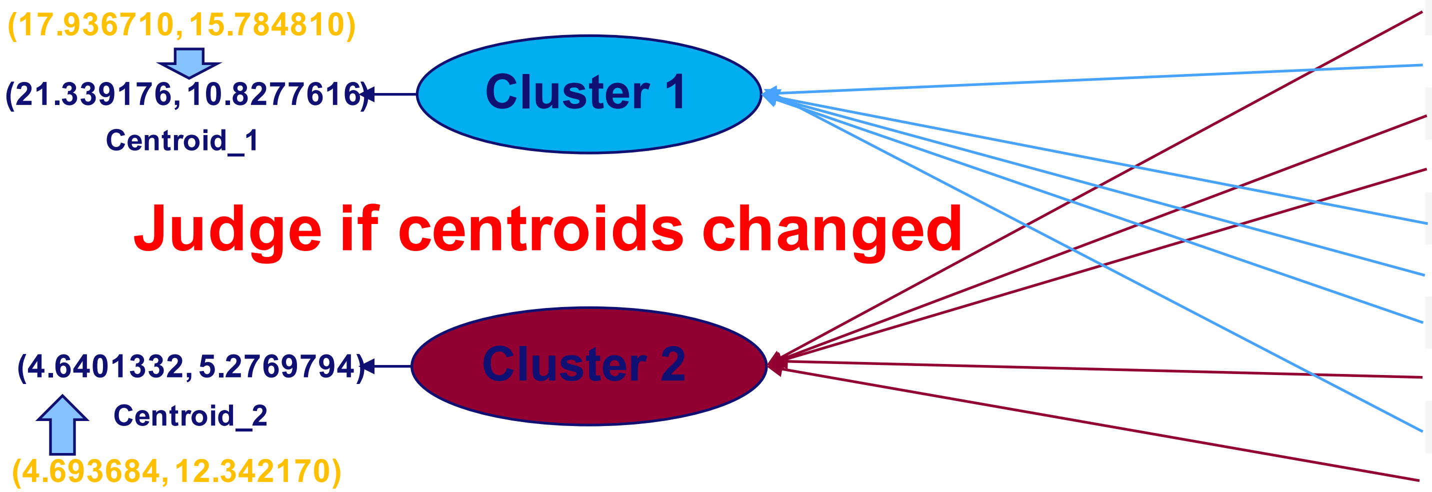
	x1	x2
0	2.072345	-3.241693
1	17.936710	15.784810
2	1.083576	7.319176
3	11.120670	14.406780
4	23.711550	2.557729
5	24.169930	32.024780
6	21.665780	4.892855
7	4.693684	12.342170
8	19.211910	-1.121366
9	4.230391	-4.441536

Example data set

Chair of Process and Data Science

Clustering

- An example implementation of k-means algorithm

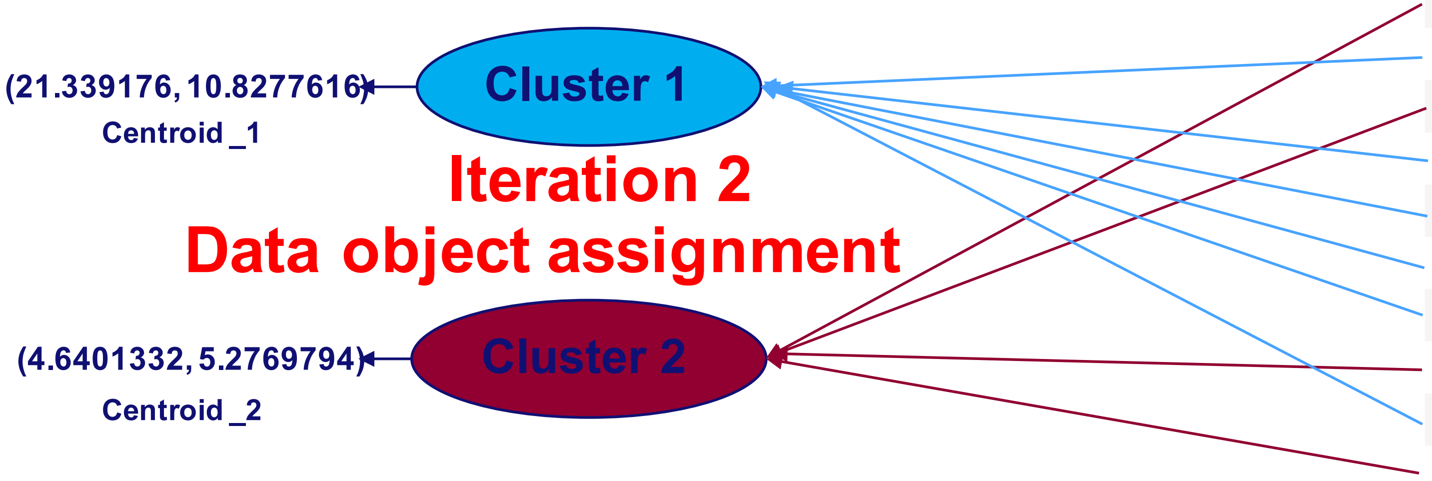


	x1	x2
0	2.072345	-3.241693
1	17.936710	15.784810
2	1.083576	7.319176
3	11.120670	14.406780
4	23.711550	2.557729
5	24.169930	32.024780
6	21.665780	4.892855
7	4.693684	12.342170
8	19.211910	-1.121366
9	4.230391	-4.441536

Example data set

Clustering

- An example implementation of k-means algorithm



	x1	x2
0	2.072345	-3.241693
1	17.936710	15.784810
2	1.083576	7.319176
3	11.120670	14.406780
4	23.711550	2.557729
5	24.169930	32.024780
6	21.665780	4.892855
7	4.693684	12.342170
8	19.211910	-1.121366
9	4.230391	-4.441536

Example data set

Chair of Process and Data Science

Clustering

- An example implementation of k-means algorithm

(21.339176, 10.8277616)

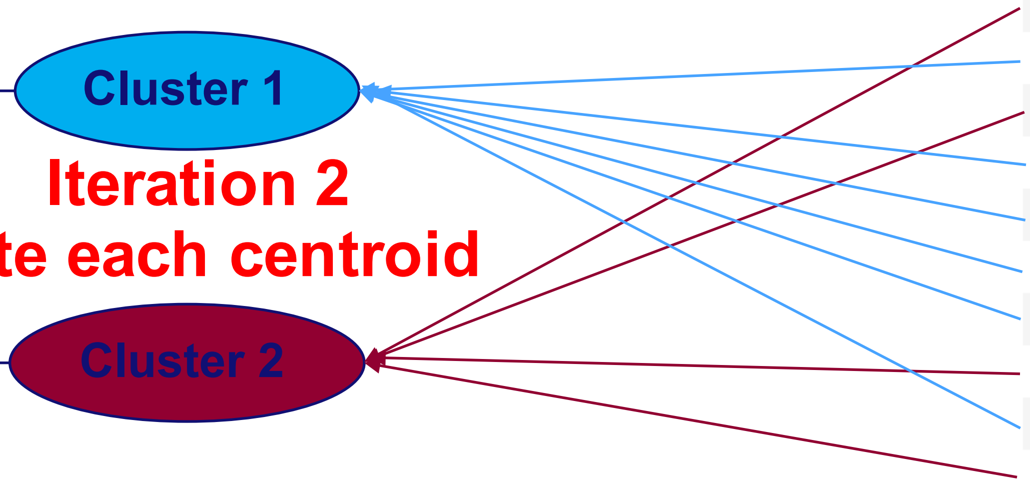
(19.636092, 11.4242647)

Centroid _1

(4.6401332, 5.2769794)

(3.019999, 2.99452925)

Centroid _2



	x1	x2
0	2.072345	-3.241693
1	17.936710	15.784810
2	1.083576	7.319176
3	11.120670	14.406780
4	23.711550	2.557729
5	24.169930	32.024780
6	21.665780	4.892855
7	4.693684	12.342170
8	19.211910	-1.121366
9	4.230391	-4.441536

Example data set

Chair of Process and Data Science

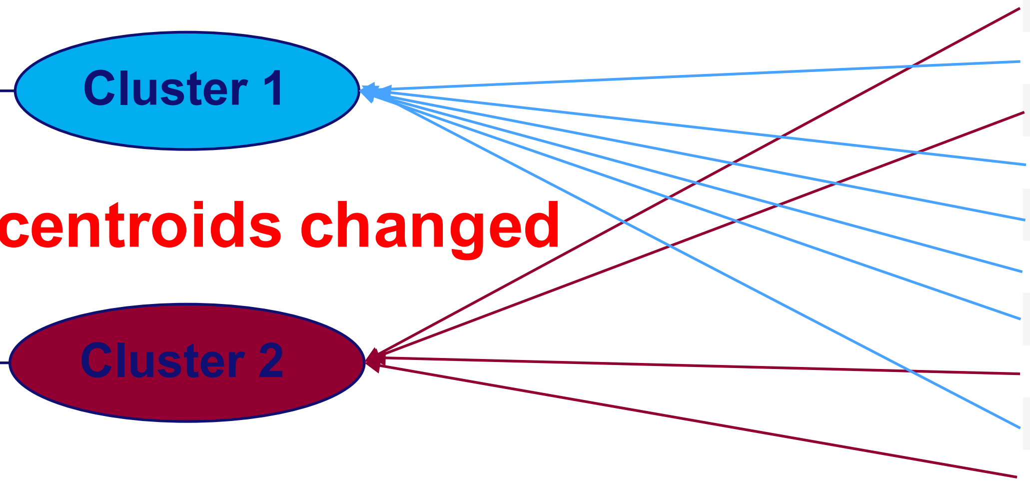
Clustering

- **An example implementation of k-means algorithm**

(21.339176, 10.8277616)

(19.636092, 11.4242647)

Centroid _1



Judge if centroids changed

(4.6401332, 5.2769794)

(3.019999, 2.99452925)

Centroid _2

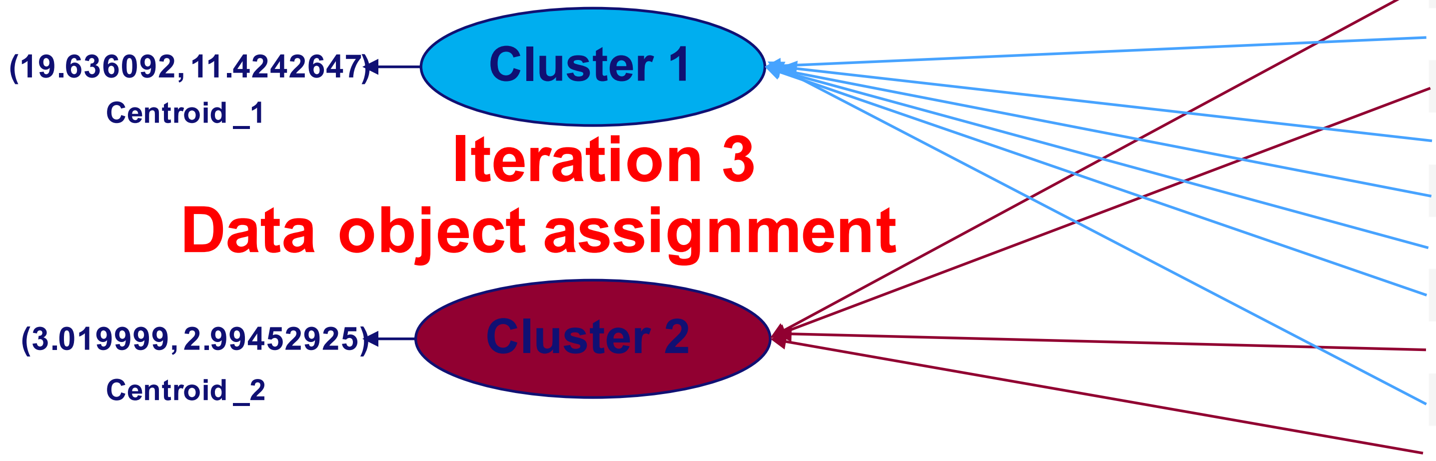
	x1	x2
0	2.072345	-3.241693
1	17.936710	15.784810
2	1.083576	7.319176
3	11.120670	14.406780
4	23.711550	2.557729
5	24.169930	32.024780
6	21.665780	4.892855
7	4.693684	12.342170
8	19.211910	-1.121366
9	4.230391	-4.441536

Example data set

Chair of Process and Data Science

Clustering

- An example implementation of k-means algorithm



	x1	x2
0	2.072345	-3.241693
1	17.936710	15.784810
2	1.083576	7.319176
3	11.120670	14.406780
4	23.711550	2.557729
5	24.169930	32.024780
6	21.665780	4.892855
7	4.693684	12.342170
8	19.211910	-1.121366
9	4.230391	-4.441536

Example data set

Clustering

- An example implementation of k-means algorithm

(19.636092, 11.4242647)

(19.636092, 11.4242647)

Centroid _1

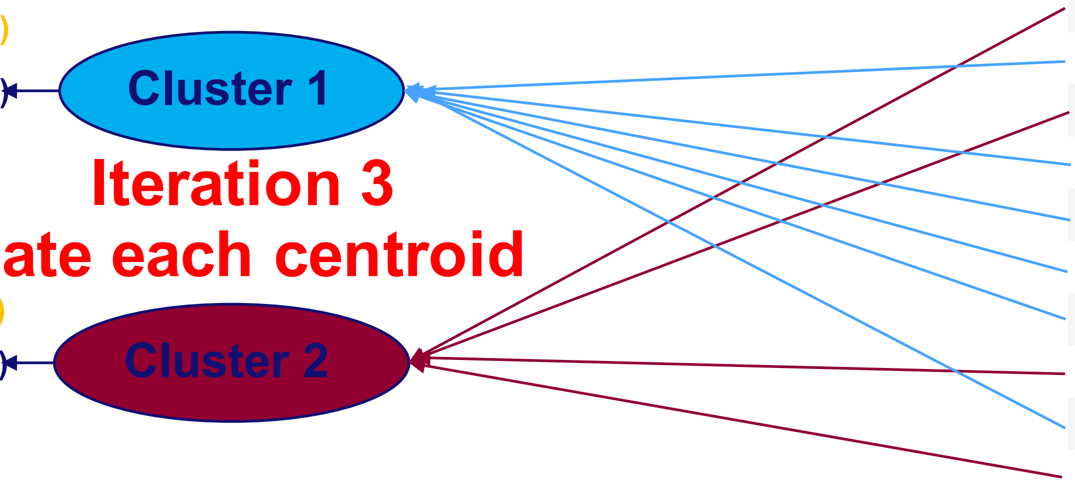
Iteration 3

Update each centroid

(3.019999, 2.99452925)

(3.019999, 2.99452925)

Centroid _2



	x1	x2
0	2.072345	-3.241693
1	17.936710	15.784810
2	1.083576	7.319176
3	11.120670	14.406780
4	23.711550	2.557729
5	24.169930	32.024780
6	21.665780	4.892855
7	4.693684	12.342170
8	19.211910	-1.121366
9	4.230391	-4.441536

Example data set

Chair of Process and Data Science

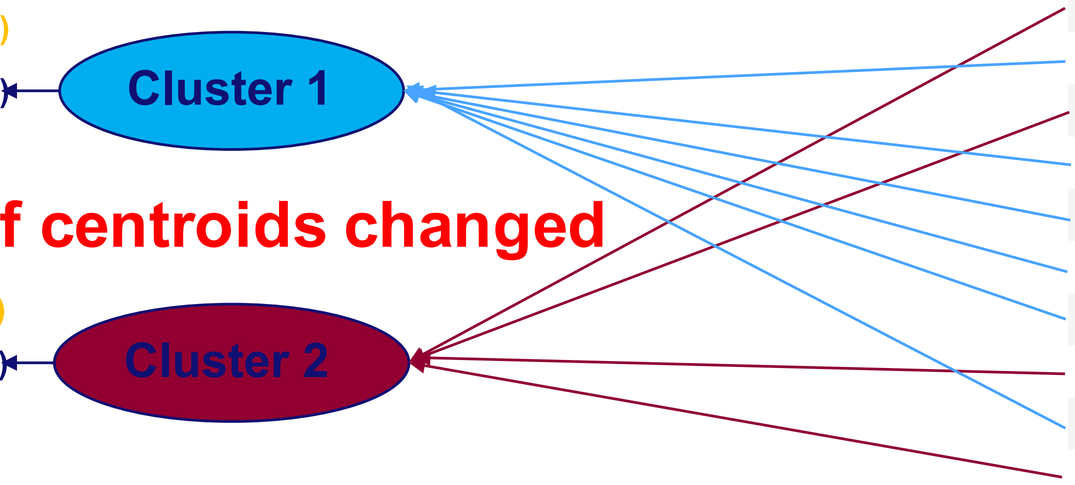
Clustering

- An example implementation of k-means algorithm

(19.636092, 11.4242647)

(19.636092, 11.4242647)

Centroid _1



Judge if centroids changed

(3.019999, 2.99452925)

(3.019999, 2.99452925)

Centroid _2

	x1	x2
0	2.072345	-3.241693
1	17.936710	15.784810
2	1.083576	7.319176
3	11.120670	14.406780
4	23.711550	2.557729
5	24.169930	32.024780
6	21.665780	4.892855
7	4.693684	12.342170
8	19.211910	-1.121366
9	4.230391	-4.441536

Example data set

Clustering

- Exercise 1: cluster data objects from data set 1 manually
- Conditions
  - K = 2 :– cluster 1 and cluster 2
  - Data object with index ‘4’ as initial centroid for cluster 1
  - Data object with index ’8’ as initial centroid for cluster 2
  - Use Euclidean distance
- You should calculate and output
  - Centroids of the found clusters
  - Data objects for each of the two final clusters

	x1	x2
0	0.407503	15.297050
1	7.314846	3.309312
2	-3.438403	-12.025270
3	17.639350	-3.212345
4	4.415292	22.815550
5	11.941220	8.122487
6	0.725853	1.806819
7	8.185273	28.132600
8	-5.773587	1.024800
9	18.769430	24.169460

Data set 1

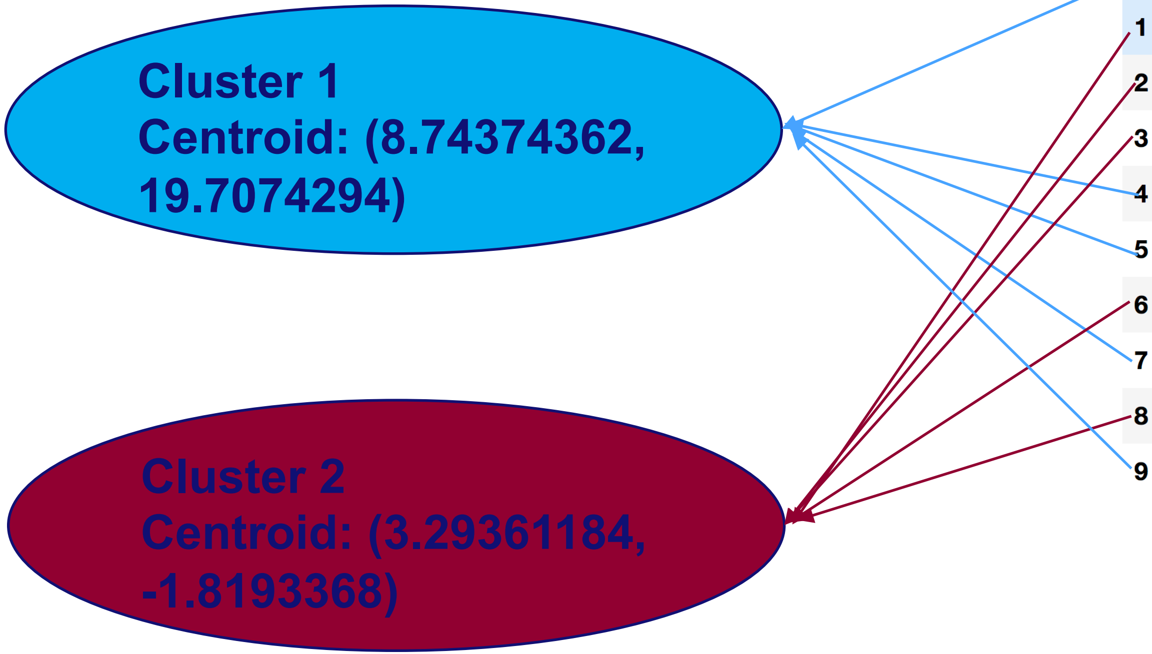
$$d(i,j) = \sqrt{(x_{i1}-x_{j1})^2 + (x_{i2}-x_{j2})^2 + \cdots + (x_{ip}-x_{jp})^2}$$

Euclidean distance

Chair of Process and Data Science

Clustering

- **Solution 1: cluster data items from data set 1 manually**



	x1	x2
0	0.407503	15.297050
1	7.314846	3.309312
2	-3.438403	-12.025270
3	17.639350	-3.212345
4	4.415292	22.815550
5	11.941220	8.122487
6	0.725853	1.806819
7	8.185273	28.132600
8	-5.773587	1.024800
9	18.769430	24.169460

Data set 1

Clustering

- Some weaknesses of k-means algorithm
  - Number of clusters needs to be decided beforehand
  - Can only discover spherical clusters (compare to density-based methods)
  - Sensitive to outliers (show this to you later)

Frequent Itemsets

TID	Set of items
0	bread, meat, wine
1	bread, meat
2	pizza, wine
3	bread, meat, pizza, wine

Set of transactions D

Minimum support$_{count}$ = 2

Item set	Support
{bread}	3
{meat}	3
{pizza}	2
{wine}	3
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2
{bread, meat, wine}	2

Frequent itemsets

Chair of Process and Data Science

Frequent Itemsets

- **Basic ideas of Apriori algorithm**
  - **Apriori rule: all the non-empty sub-itemsets of frequent itemsets must be frequent.**

Frequent Itemsets

TID	Set of items
0	bread, meat, wine
1	bread, meat
2	pizza, wine
3	bread, meat, pizza, wine

Set of transactions D

Minimum support$_{count}$ = 2

Item set	Support
{bread}	3
{meat}	3
{pizza}	2
{wine}	3
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2
{bread, meat, wine}	2

Frequent itemsets

Itemset {bread, meat, wine} is frequent

Sub-item set	Support	
{bread}	3	frequent
{meat}	3	frequent
{wine}	3	frequent
{bread, meat}	3	frequent
{bread, wine}	2	frequent
{meat, wine}	2	frequent

Chair of Process and Data Science

Frequent Itemsets

- Basic ideas of Apriori algorithm
  - **Apriori rule**: all the non-empty **sub-itemsets** of **frequent itemsets** should **be frequent**.
  - Use the set L_k of frequent itemsets with **length k** to search for both candidate set C_{k+1} of itemsets and set L_{k+1} of frequent itemsets with **length k+1**

Frequent Itemsets

Item set	Support
{bread}	3
{meat}	3
{pizza}	2
{wine}	3
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2
{bread, meat, wine}	2

L_1

Frequent itemsets

First, Scan D to generate L_1

Chair of Process and Data Science

Frequent Itemsets

Item set	Support
{bread}	3
{meat}	3
{pizza}	2
{wine}	3
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2
{bread, meat, wine}	2

L_1

Frequent itemsets

First, Scan D to generate L_1

TID	Set of items
0	bread, meat, wine
1	bread, meat
2	pizza, wine
3	bread, meat, pizza, wine

Set of transactions D

min_sup represents minimum support$_{count}$

Scan, compare support to min_sup

remove infrequent ones

Item set	Support
{bread}	3
{meat}	3
{pizza}	2
{wine}	3

L_1

Chair of Process and Data Science

Frequent Itemsets

- **Exercise 2: manually generate L_1 from set S**
  - **Set minimum support$_{count}$ to 2**

Item set	Support
?	

L_1

TID	Data items
1	A, B, E
2	C, A, D
3	C, B, D
4	C, A, B, E

Example data set S

Chair of Process and Data Science

Frequent Itemsets

- **Solution 2:**

Item set	Support
{A}	3
{B}	3
{C}	3
{D}	2
{E}	2

L_1

Frequent Itemsets

Item set	Support
{bread}	3
{meat}	3
{pizza}	2
{wine}	3
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2
{bread, meat, wine}	2

L_2

L_3

Frequent itemsets

How to generate C_k from L_{k-1}, when $k >= 2$?

Chair of Process and Data Science

Frequent Itemsets

Item set	Support
{bread}	3
{meat}	3
{pizza}	2
{wine}	3
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2
{bread, meat, wine}	2

L_2

L_3

Frequent itemsets

How to generate C_k from L_{k-1}, when $k >= 2$?

1. Keep items in itemsets in L_{k-1} in an ascending order according to their dictionary order

Chair of Process and Data Science

Frequent Itemsets

Item set	Support
{bread}	3
{meat}	3
{pizza}	2
{wine}	3
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2
{bread, meat, wine}	2

L_2

L_3

Frequent itemsets

How to generate C_k from L_{k-1}, when $k >= 2$?

1. Keep items in itemsets in L_{k-1} in an ascending order according to their dictionary order

Set of items	Support
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2

L_2

For generating C_3

bread < meat

bread < wine

meat < wine

pizza < wine

Frequent Itemsets

Item set	Support
{bread}	3
{meat}	3
{pizza}	2
{wine}	3
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2
{bread, meat, wine}	2

L_2

L_3

Frequent itemsets

How to generate C_k from L_{k-1}, when $k >= 2$?

2. Merge each possible pair of itemsets from L_{k-1}

Let $(\{p_1, p_2, \ldots, p_{k-2}, p_{k-1}\}, \{q_1, q_2, \ldots, q_{k-2}, q_{k-1}\})$ be a pair of two itemsets from L_{k-1}.

-- if: $\{p_1, p_2, \ldots, p_{k-2}\} = \{q_1, q_2, \ldots, q_{k-2}\}$ and $p_{k-1} < q_{k-1}$

-- then merge them into: $\{p_1, p_2, \ldots, p_{k-2}, p_{k-1}, q_{k-1}\}$

-- else: not merge them

Chair of Process and Data Science

Frequent Itemsets

Item set	Support
{bread}	3
{meat}	3
{pizza}	2
{wine}	3
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2
{bread, meat, wine}	2

L_2

L_3

Frequent itemsets

How to generate C_k from L_{k-1}, where $k >= 2$?

2. Merge each possible pair of items from L_{k-1}

Set of items	Support
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2

L_2

({bread, meat}, {bread, wine}) => {bread, meat, wine}

Because {bread} = {bread} and meat < wine

Frequent Itemsets

Item set	Support
{bread}	3
{meat}	3
{pizza}	2
{wine}	3
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2
{bread, meat, wine}	2

L_2

L_3

Frequent itemsets

How to generate C_k from L_{k-1}, where $k >= 2$?

2. Merge each possible pair of items from L_{k-1}

Set of items	Support
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2

L_2

({bread, wine}, {bread, meat}) ≠> {bread, meat, wine}

Because wine > meat

Frequent Itemsets

Item set	Support
{bread}	3
{meat}	3
{pizza}	2
{wine}	3
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2
{bread, meat, wine}	2

L_2

L_3

Frequent itemsets

How to generate C_k from L_{k-1}, where $k >= 2$?

2. Merge each possible pair of items from L_{k-1}

Set of items	Support
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2

L_2

({bread, wine}, {meat, wine}) ≠> {bread, meat, wine}

Because {bread} ≠ {meat}

Chair of Process and Data Science

Frequent Itemsets

Item set	Support
{bread}	3
{meat}	3
{pizza}	2
{wine}	3
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2
{bread, meat, wine}	2

L_2

L_3

Frequent itemsets

How to generate C_k from L_{k-1}, where $k >= 2$?

2. Merge each possible pair of items from L_{k-1}

Set of items	Support
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2

L_2

merge itemsets in L_2

Item set
{bread, meat, wine}

C_3

Chair of Process and Data Science

Frequent Itemsets

Item set	Support
{bread}	3
{meat}	3
{pizza}	2
{wine}	3
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2
{bread, meat, wine}	2

L_2

L_3

Frequent itemsets

Pre-pruning for C_k when k >= 2 before scanning D

Frequent Itemsets

Item set	Support
{bread}	3
{meat}	3
{pizza}	2
{wine}	3
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2
{bread, meat, wine}	2

L_2

L_3

Frequent itemsets

Pre-pruning for C_k when k >= 2 before scanning D

Pruning rule: given an itemset $i_k \subseteq C_k$, if not all sub-itemsets of length k-1 of i_k are contained in L_{k-1}, then remove i_k from C_k

Chair of Process and Data Science

Frequent Itemsets

Item set	Support
{bread}	3
{meat}	3
{pizza}	2
{wine}	3
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2
{bread, meat, wine}	2

L_2

L_3

Frequent itemsets

Pre-pruning for C_k when k >= 2 before scanning D

Set of items	Support
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2

L_2

Item set
{bread, meat, wine}

C_3

The sub-itemsets of length 2 for {bread, meat, wine} include: {bread, meat}, {bread, wine}, {meat, wine}, which are all Included in L_2. So keep {bread, meat, wine}.

Frequent Itemsets

Item set	Support
{bread}	3
{meat}	3
{pizza}	2
{wine}	3
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2
{bread, meat, wine}	2

L_2

L_3

Frequent itemsets

Pre-pruning for C_k when k >= 2 before scanning D

Item set
{bread, meat, wine}

C_3

Pre-pruning

Item set
{bread, meat, wine}

C_3

Frequent Itemsets

Item set	Support
{bread}	3
{meat}	3
{pizza}	2
{wine}	3
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2
{bread, meat, wine}	2

L_2

L_3

Frequent itemsets

Scan D to add supports for itemsets in C_k

Item set
{bread, meat, wine}

C_3

scan

item set	support
{bread, meat, wine}	2

C_3

Chair of Process and Data Science

Frequent Itemsets

Item set	Support
{bread}	3
{meat}	3
{pizza}	2
{wine}	3
{bread, meat}	3
{bread, wine}	2
{meat, wine}	2
{pizza, wine}	2
{bread, meat, wine}	2

L_2

L_3

Frequent itemsets

Remove infrequent itemsets and generate L_k

item set	support
{bread, meat, wine}	2

C_3

compare support to min_sup

remove infrequent ones

item set	support
{bread, meat, wine}	2

L_3

Chair of Process and Data Science

Frequent Itemsets

- Exercise 3: given L_2 for set S, show the three C_3 and L_3
  - Set minimum support_{count} to 2

Item set	Support
{A, B}	2
{A, C}	2
{A, E}	2
{B, C}	2
{B, E}	2
{C, D}	2

L_2

merge itemsets in L_2 →

Item set	Support
?	

C_3

pruning →

Item set	Support
?	

C_3

Scan S →

Item set	Support
?	

C_3

TID	Data items
1	A, B, E
2	C, A, D
3	C, B, D
4	C, A, B, E

Data set S

compare support to min_sup →
remove infrequent ones

Item set	Support
?	

L_3

Chair of Process and Data Science

Frequent Itemsets

- **Solution 3:**

Item set	Support
{A, B}	2
{A, C}	2
{A, E}	2
{B, C}	2
{B, E}	2
{C, D}	2

L_2

merge itemsets in L_2

Item set
{A, B, C}
{A, B, E}
{A, C, E}
{B, C, E}

C_3

pruning

Item set
{A, B, C}
{A, B, E}

C_3

Scan S

Item set	Support
{A, B, C}	1
{A, B, E}	2

C_3

compare support to min_sup

remove infrequent ones

item set	Support
{A, B, E}	2

L_3

Chair of Process and Data Science

Association Rules

- **T** is a set of transactions
- **I** is the set of all possible itemsets composed by items in **T**
- **A** $\subseteq$ **I** and **B** $\subseteq$ **I** are two itemsets/sub-itemsets from T
- **A => B** is an association rule

Association Rules

- **Usually, we would like to discover the association rule A => B of which the support and confidence are above certain levels.**

Association Rules

- $support(A \Rightarrow B) = support(A \cup B) = \frac{support_{count}(A \cup B)}{|T|}$
- $confidence(A \Rightarrow B) = \frac{support(A \cup B)}{support(A)} = \frac{support_{count}(A \cup B)}{support_{count}(A)}$

- min_sup represents minimum support and min_conf represents minimum confidence

- A => B is a desired association rule if:
- $support(A \Rightarrow B)$ >= min_sup and $confidence(A \Rightarrow B)$ >= min_conf

Chair of Process and Data Science

Association Rules

- Set min_sup to 0.5, min_conf to 0.7, is {bread} => {meat} from D a desired association rule?

TID	Set of items
0	bread, meat, wine
1	bread, meat
2	pizza, wine
3	bread, meat, pizza, wine

Set of transactions D

Chair of Process and Data Science

Association Rules

- Set min_sup to 0.5, min_conf to 0.7, is {bread} => {meat} from D a desired association rule?

TID	Set of items
0	bread, meat, wine
1	bread, meat
2	pizza, wine
3	bread, meat, pizza, wine

Set of transactions D

$$support(\{\text{bread}\} \Rightarrow \{\text{meat}\}) = \frac{support_{count}(\{\text{bread,meat}\})}{|D|} = \frac{3}{4} = 0.75 > \text{min\_sup}$$

$$confidence(\{\text{bread}\} \Rightarrow \{\text{meat}\}) = \frac{support_{count}(\{\text{bread,meat}\})}{support_{count}(\{\text{bread}\})} = \frac{3}{3} = 1 > \text{min\_conf}$$

{bread} => {meat} is a desired association rule

Chair of Process and Data Science

Association Rules

- **Usually, we use lift to evaluate the quality of the discovered association rule A => B**

$$lift(A \Rightarrow B) = \frac{support(A \cup B)}{support(A) \cdot support(B)} = \frac{P(A \cup B)}{P(A) \cdot P(B)}$$

If $lift(A \Rightarrow B) \approx 1$ **then** A **and** B **are independent**

If $lift(A \Rightarrow B) \ll 1$ **then** A **and** B **are negatively correlated**

If $lift(A \Rightarrow B) \gg 1$ **then** A **and** B **are positively correlated**

Chair of Process and Data Science

Association Rules

- **Evaluate the quality of the association rule {bread} => {meat} by using lift**

TID	Set of items
0	bread, meat, wine
1	bread, meat
2	pizza, wine
3	bread, meat, pizza, wine

Set of transactions D

$$lift(\{bread\} \Rightarrow \{meat\}) = \frac{support(\{bread, meat\})}{support(\{bread\}) \cdot support(\{meat\})} = \frac{(3/4)}{(3/4) \cdot (3/4)} = 1.33$$

Chair of Process and Data Science

Association Rules

- **Exercise 3: judge if {A, B} => {E}, {A} => {B} and {A} => {C} are the desired association rules under minimum support 0.5 and minimum confidence 0.75? Also evaluate the quality of the desired rules.**

TID	Data items
1	A, B, E
2	C, A, D
3	C, B, D
4	C, A, B, E

Example data set S

Association Rules

- Solution 3: judge if {A, B} => {E}, {A} => {B} and {A} => {C}
- support({A, B} => {E}) = 0.5, confidence({A, B} => {E}) = 1, lift({A, B} => {E}) = 2, it is desired association rule, and lift is larger than 1
- support({A} => {B}) = 0.5, confidence({A} => {B}) = 0.67, lift({A} => {B}) = 0.89, it is not a desired association rule
- support({A} => {C}) = 0.5, confidence({A} => {C}) = 0.67, lift({A} => {C}) = 0.89, it is not a desired association rule