Clustering

Tran Giang Son, tran-giang.son@usth.edu.vn

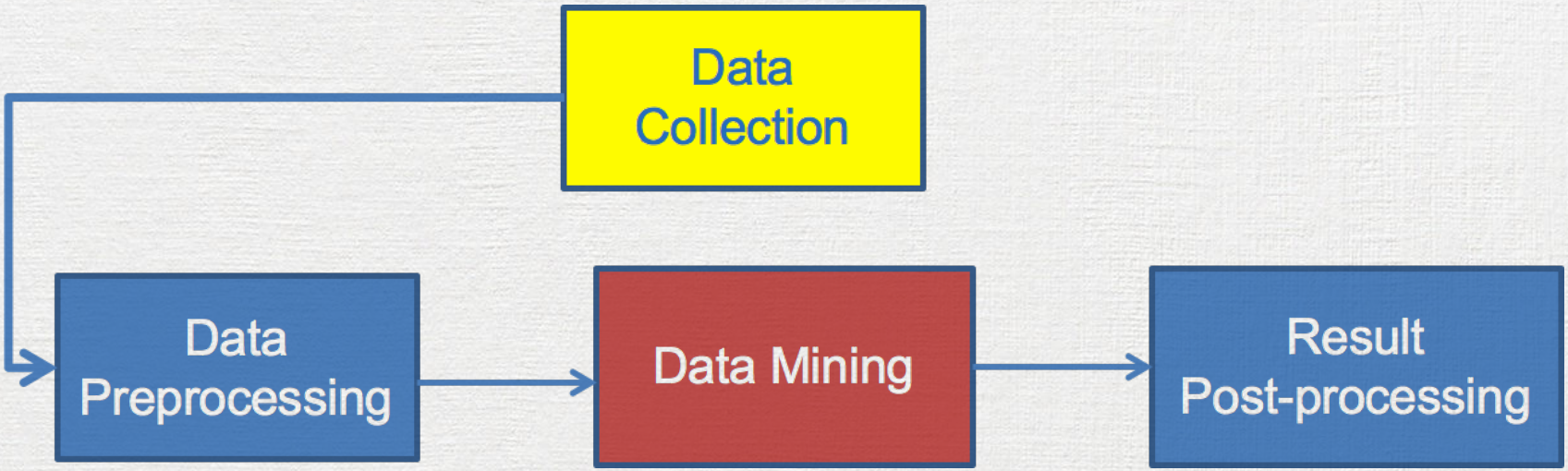
ICT Department, USTH

Clustering

Data Mining

- What: Extract information from data
- Why: A lot of data. Data is $$$

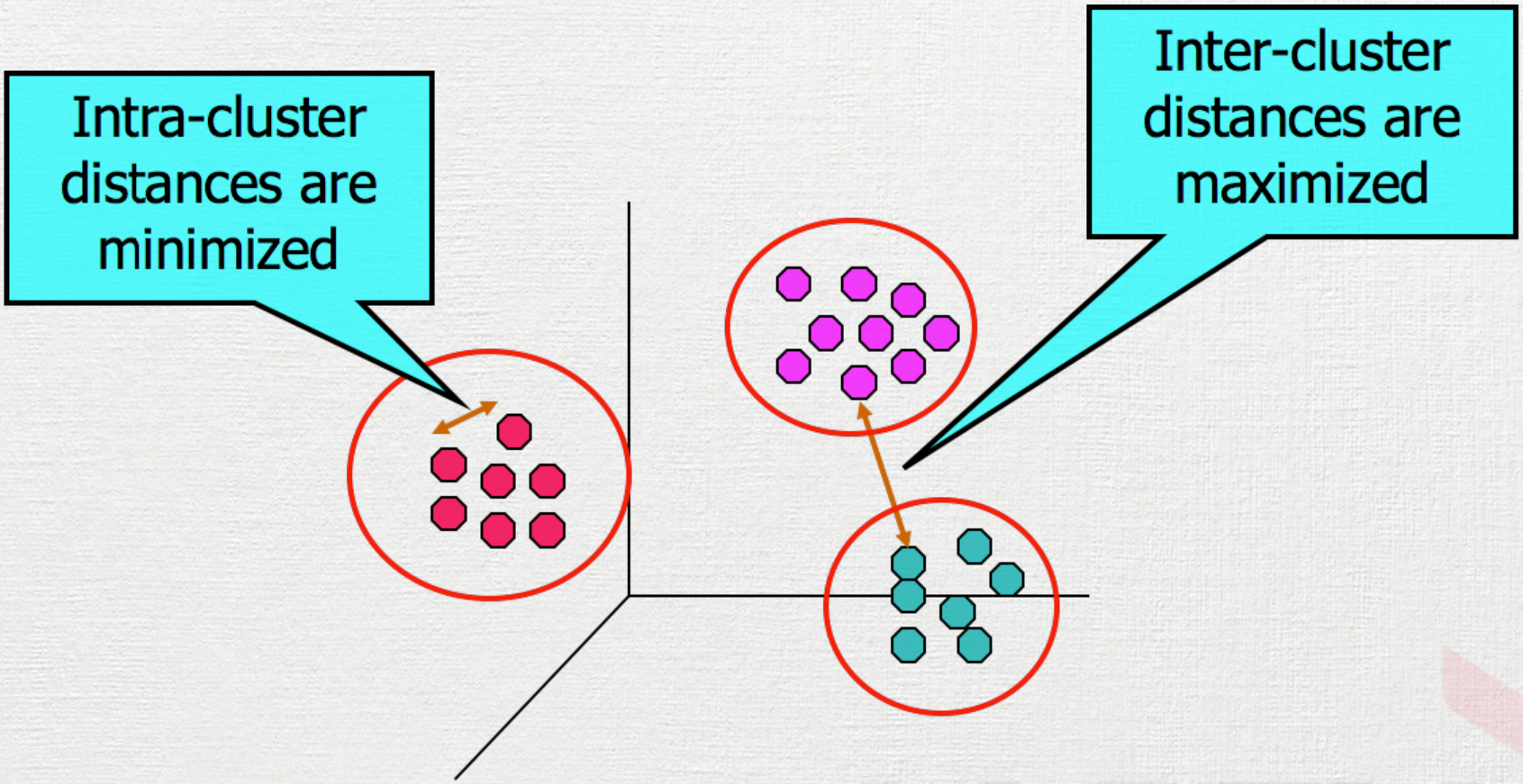
Data Mining


Data Collection
Data Preprocessing
Data Mining
Result Post-processing

Clustering

- Unsupervised learning
- Input: objects
- Output: groups of objects
  - Objects in the same group are similar
  - Objects in different group are unrelated
  - Each group is called a cluster

Clustering


Intra-cluster distances are minimized
Inter-cluster distances are maximized

Clustering

- Interpretability
- Attribute shape
- Different types of attributes
- Noisy data

Clustering

- Applications
  - Image processing: segmentation of objects
  - Biology, bioinformatics: grouping of species
  - Mobile communication: grouping of users
  - Medicine: medical imaging
  - Economics: market research, shopping items

Clustering

- Hierarchical clustering
- K-means and its variants
- DBSCAN

Hierarchical Clustering

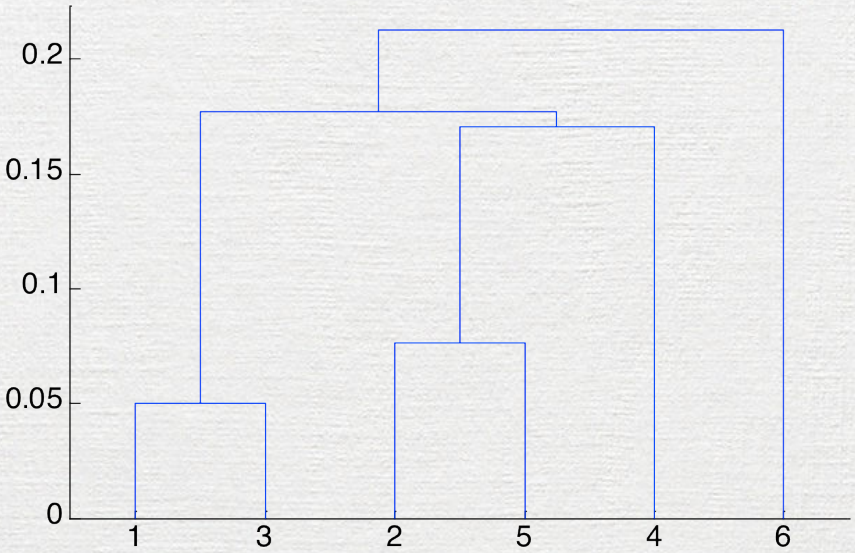
Hierarchical Clustering

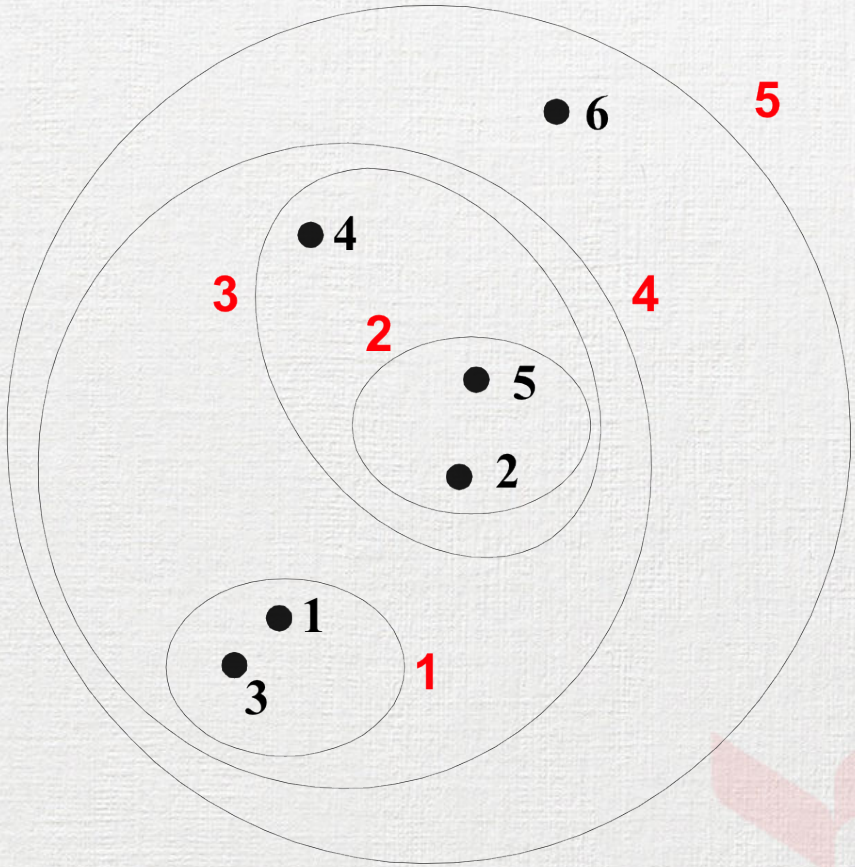
- Two main types of hierarchical clustering
  - Agglomerative: bottom up
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
  - Divisive: top down
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are k clusters)

Hierarchical Clustering

- Traditional hierarchical clustering
  - Similarity or distance matrix
  - Merge or split one cluster at a time
- Produces a set of nested clusters
  - A hierarchical tree
  - Visualized as a dendrogram
    - A tree like diagram that records the sequences of merges or splits

Hierarchical Clustering





Hierarchical Clustering

- No assumption of number of clusters
  - Simply cutting the dendogram at the proper level for a specific number of clusters
- Meaningful taxonomies
  - Biological sciences: animal kingdom
  - Image analysis: finger, palm, hand, limb, human

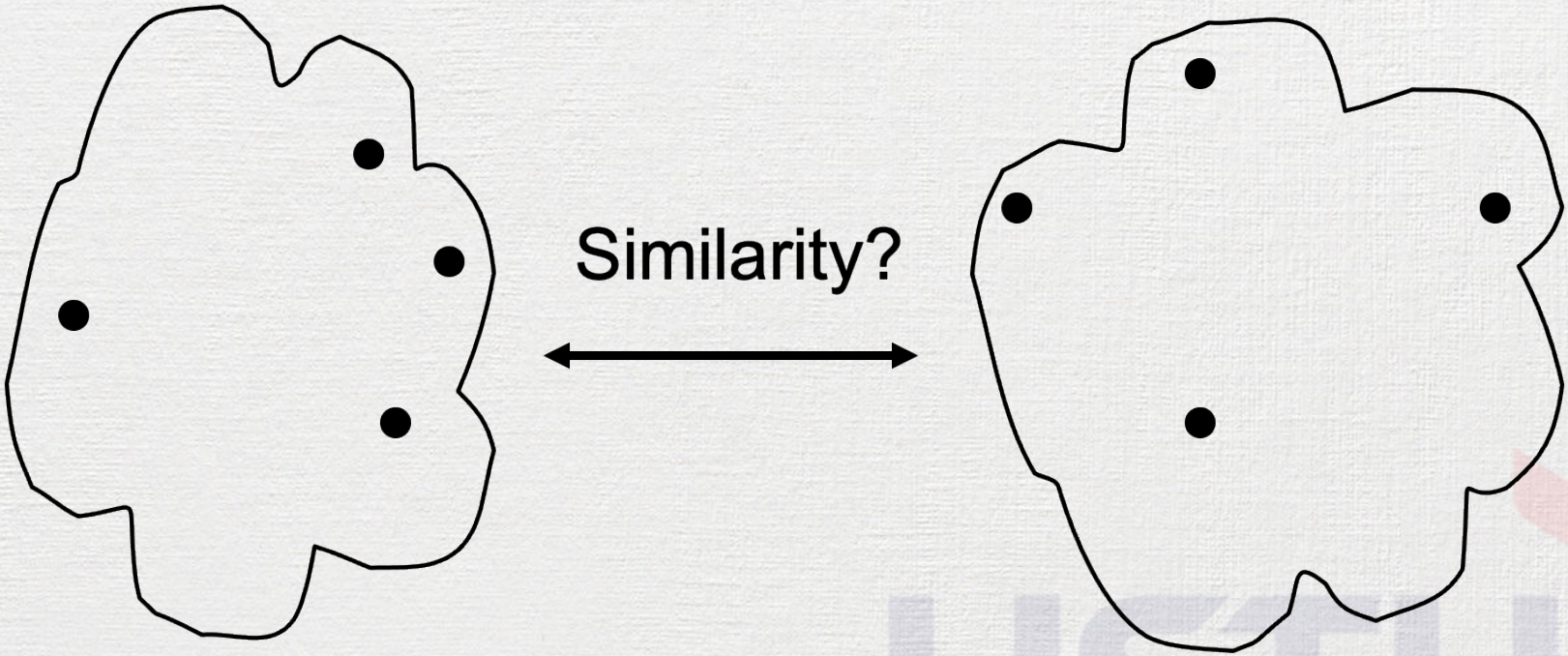
Agglomerative Hierarchical Clustering

- More popular than divisive methods
- Basic algorithm

```
Compute the proximity matrix
Let each data point be a cluster
Repeat
    Calculate distance between clusters
    Merge the two closest clusters
Until only a single cluster remains
```

Agglomerative Hierarchical Clustering: Proximity

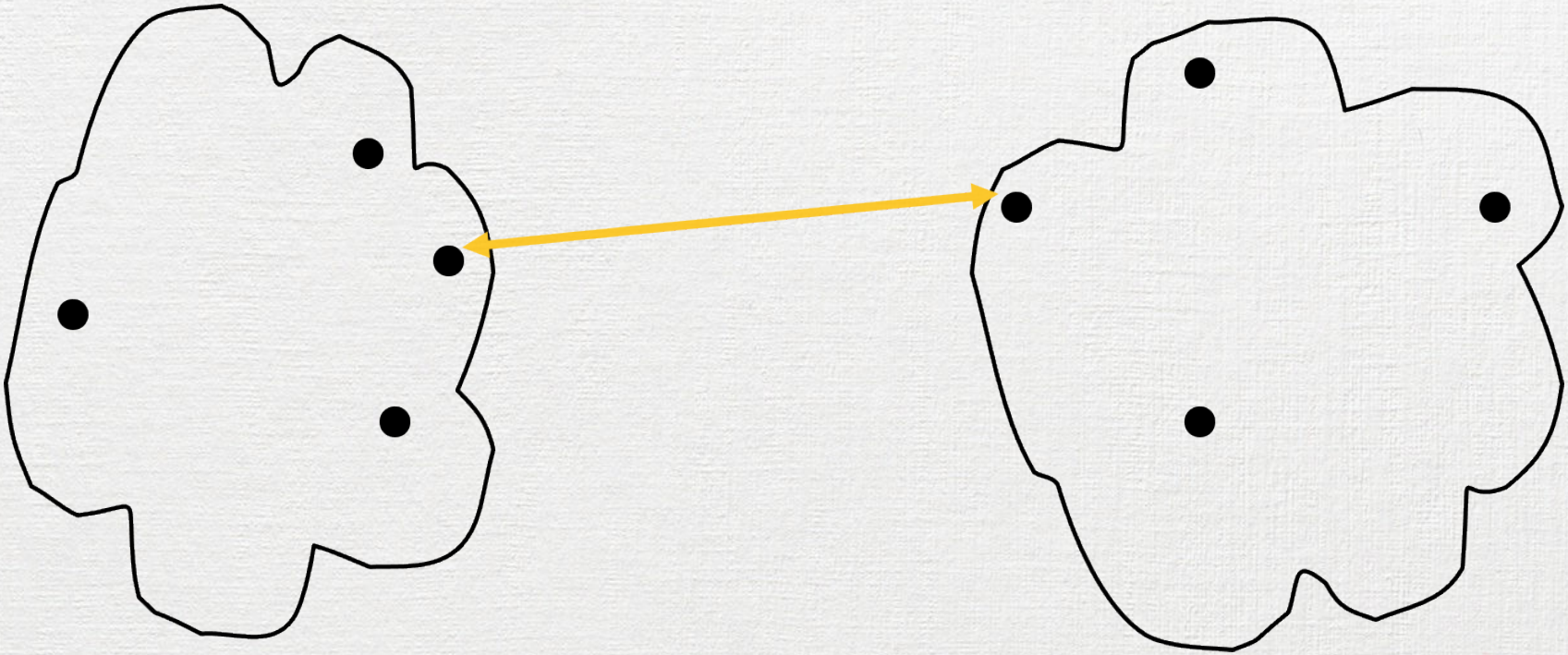
- Proximity matrix: distance between pairs of points.
- Proximity of two clusters?



Agglomerative Hierarchical Clustering: Proximity

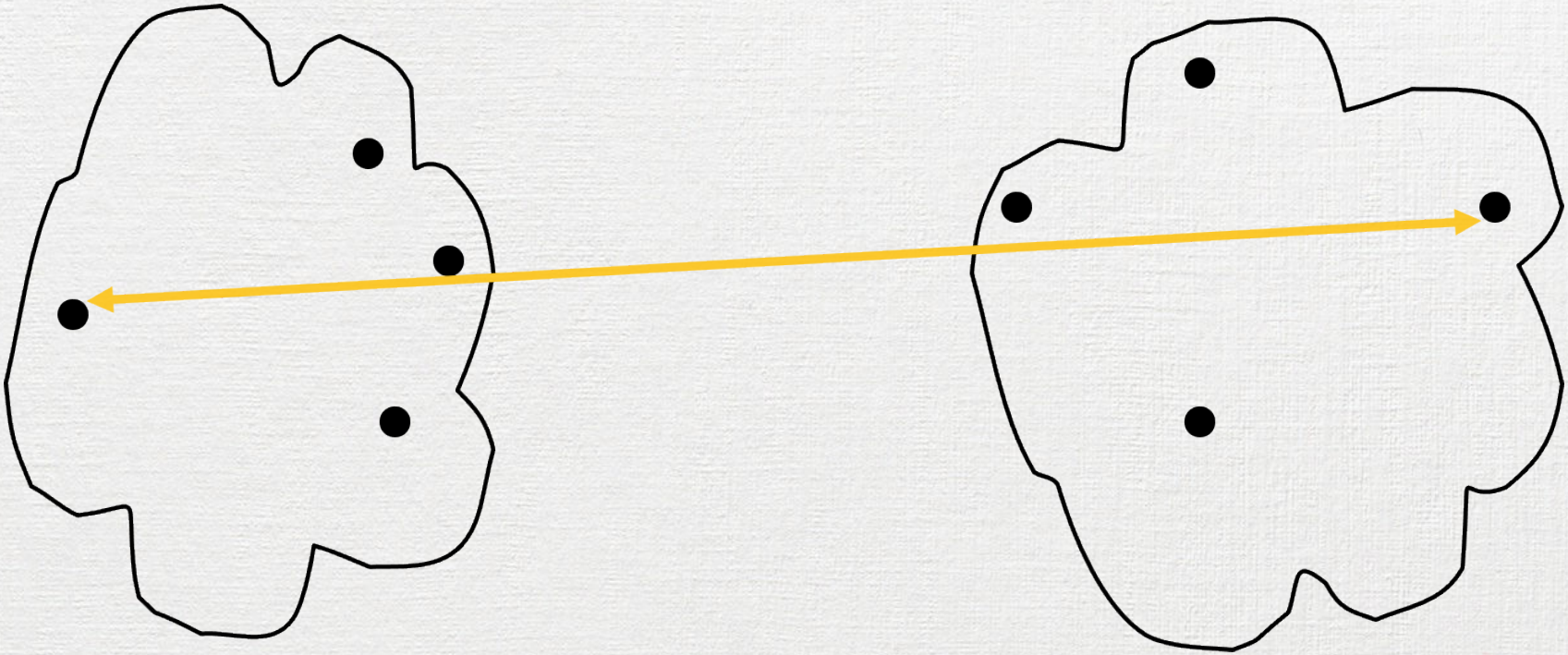
- Proximity of two clusters?
  - Min
  - Max
  - Average
  - Centroid

Agglomerative Hierarchical Clustering



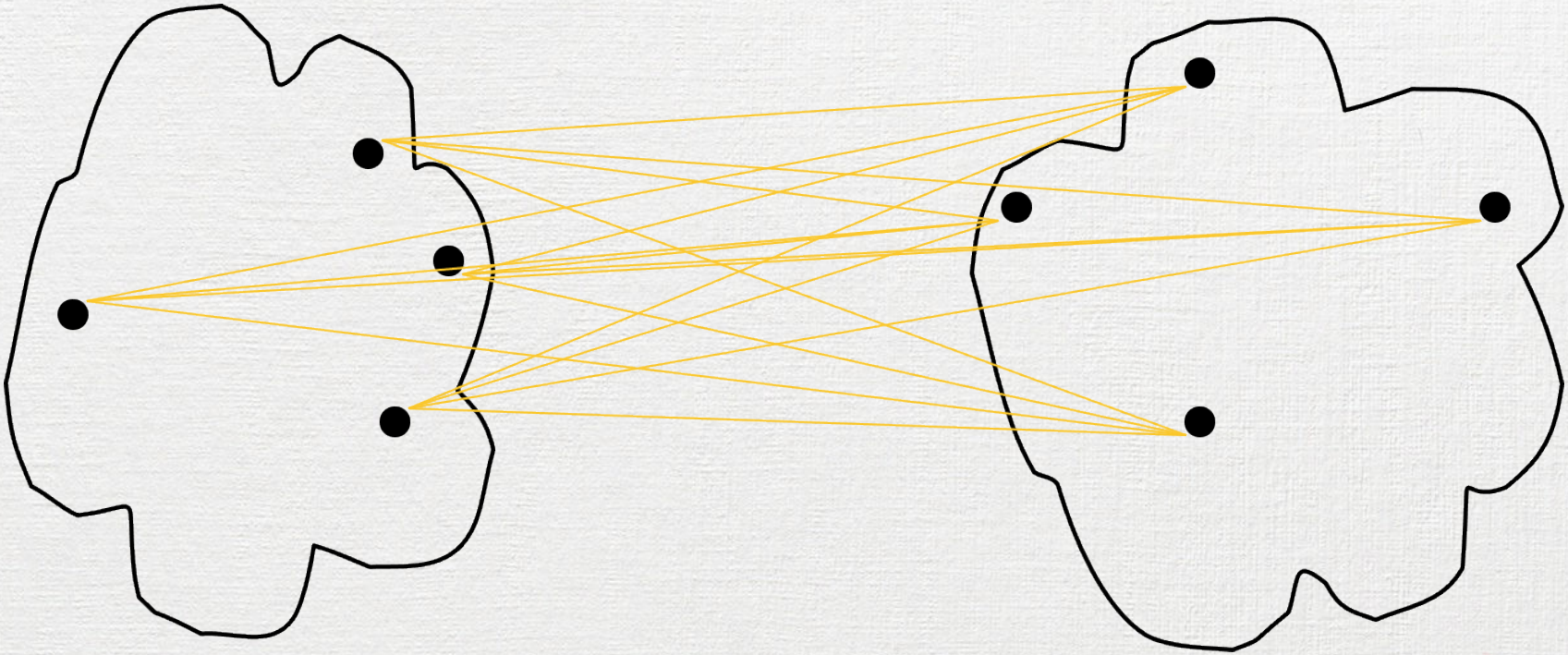
Min item distance

Agglomerative Hierarchical Clustering



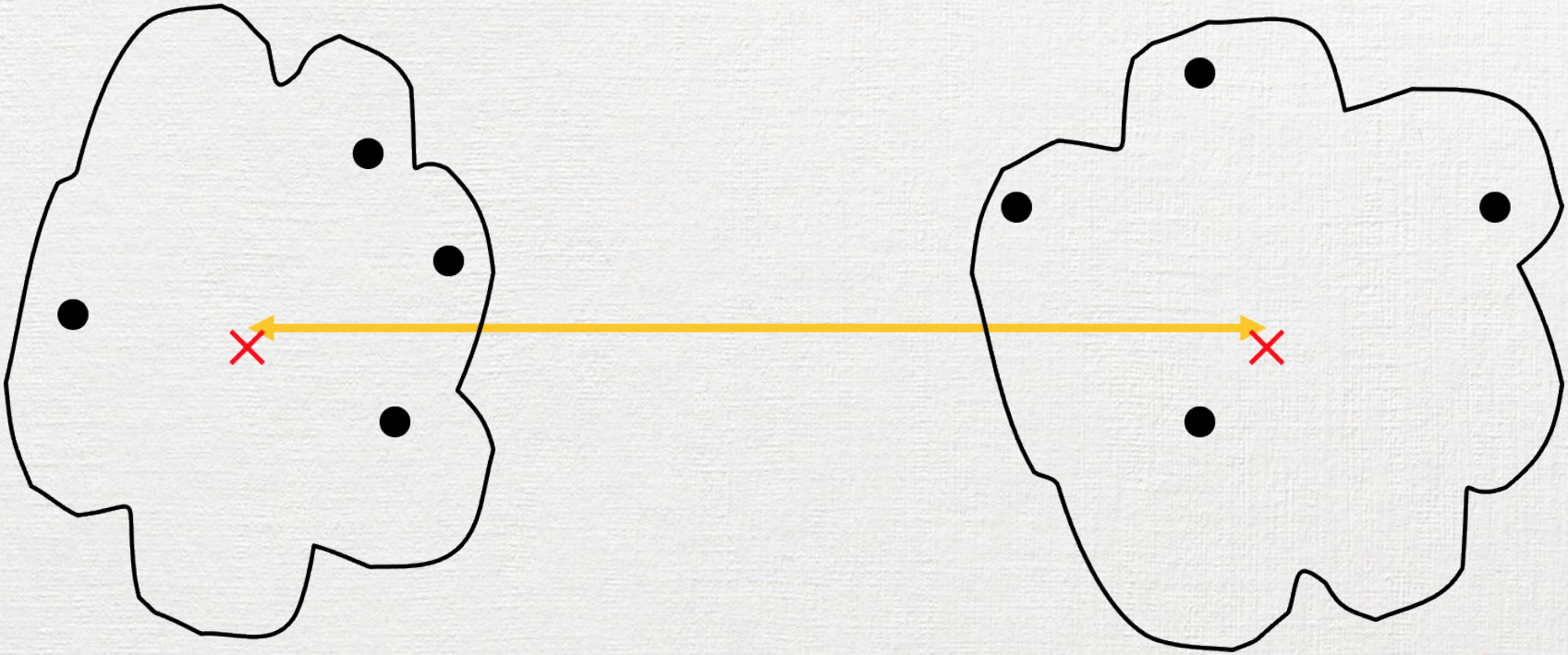
Max item distance

Agglomerative Hierarchical Clustering



Average item distance

Agglomerative Hierarchical Clustering

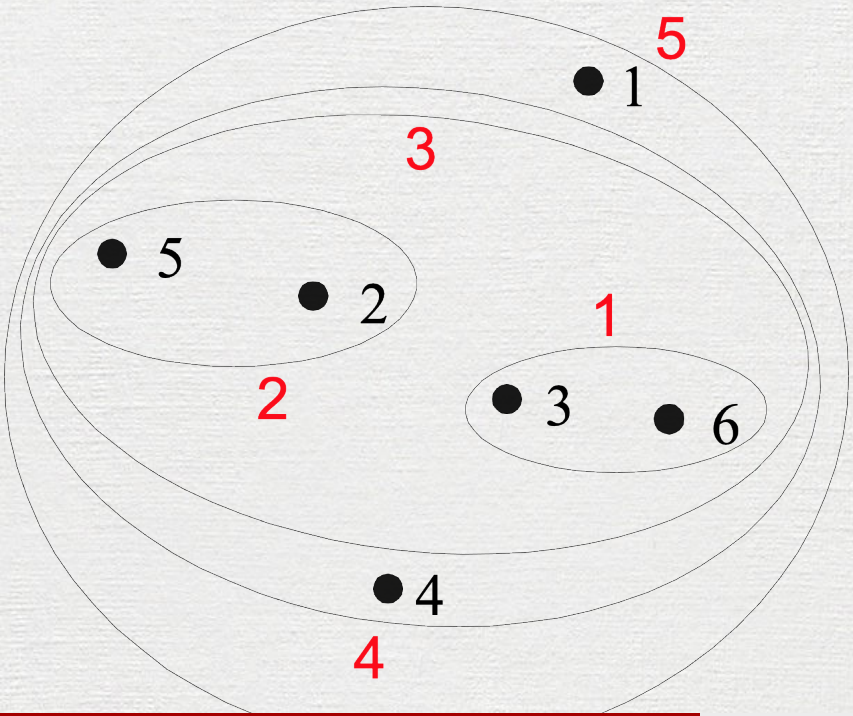


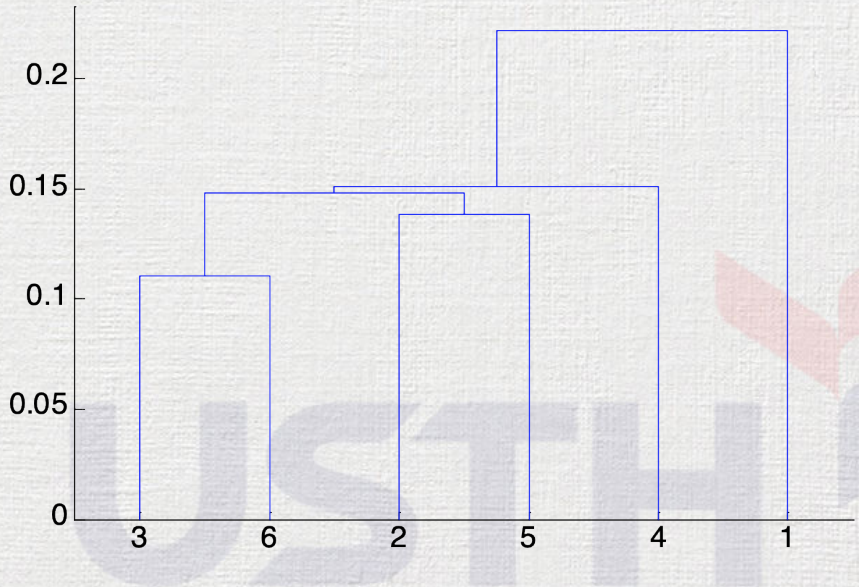
Centroid distance

Agglomerative Hierarchical Clustering

- Min

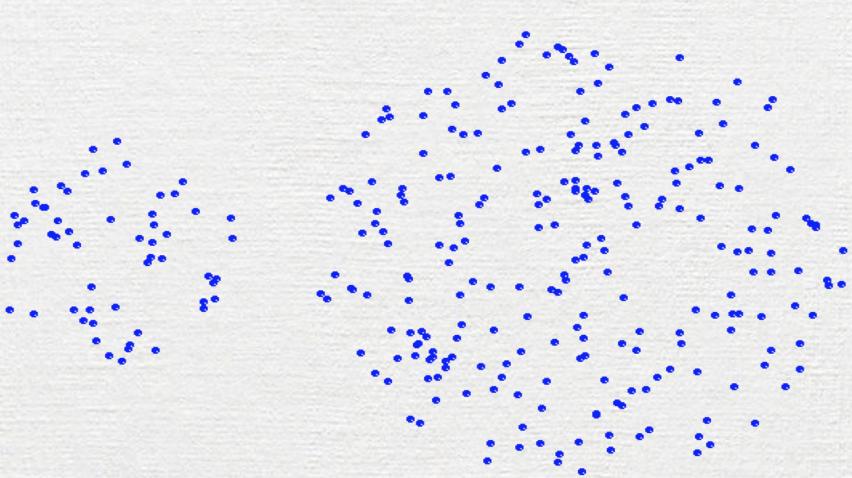
	1	2	3	4	5	6
1	0	.24	.22	.37	.34	.23
2	.24	0	.15	.20	.14	.25
3	.22	.15	0	.15	.28	.11
4	.37	.20	.15	0	.29	.22
5	.34	.14	.28	.29	0	.39
6	.23	.25	.11	.22	.39	0



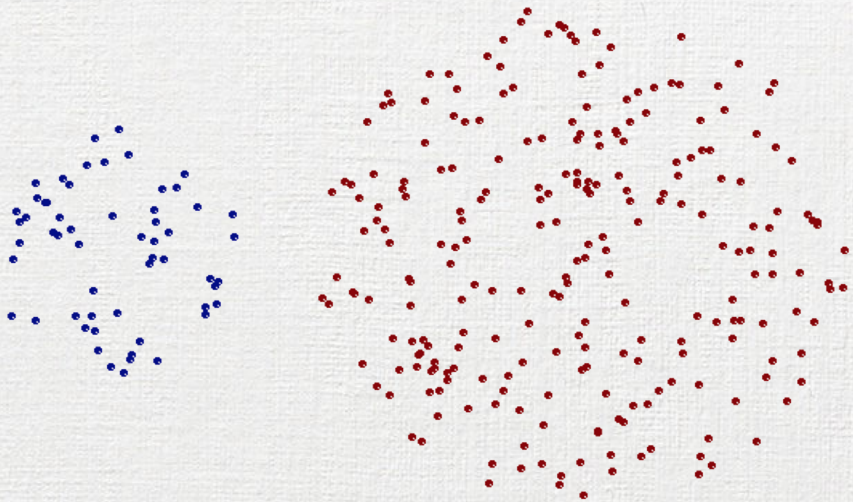


Agglomerative Hierarchical Clustering

- Min: can handle non-elliptical shapes...



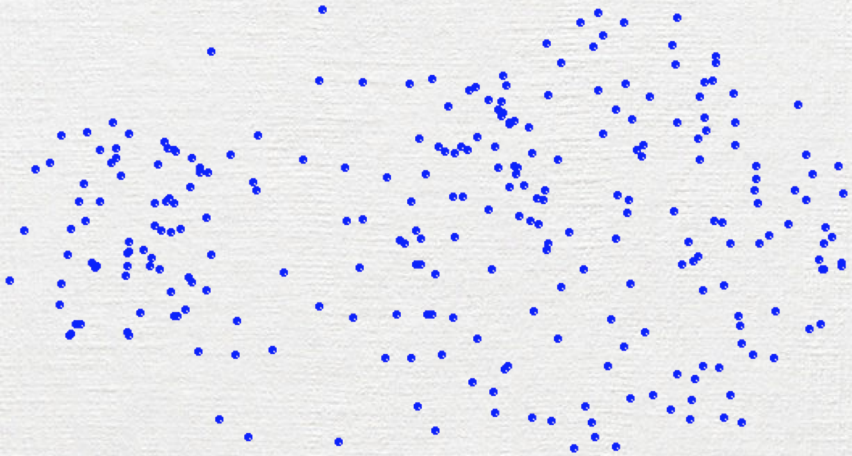
Input



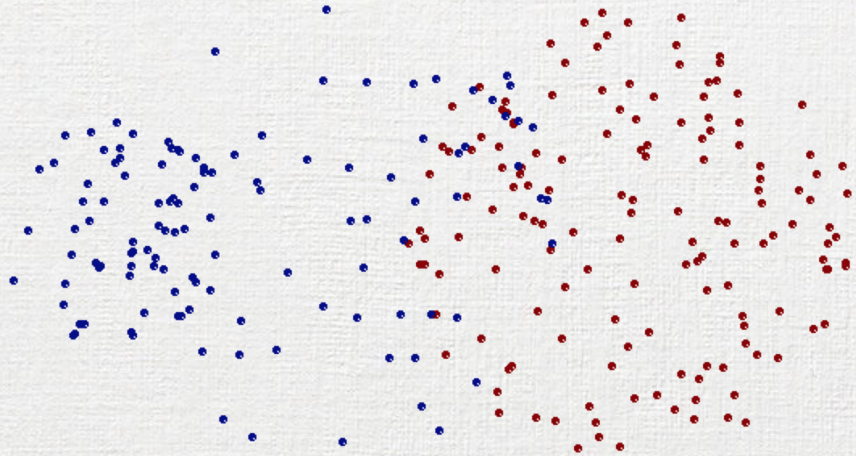
2 clusters

Agglomerative Hierarchical Clustering

- Min: . . . but prone to noise and outliers



Input

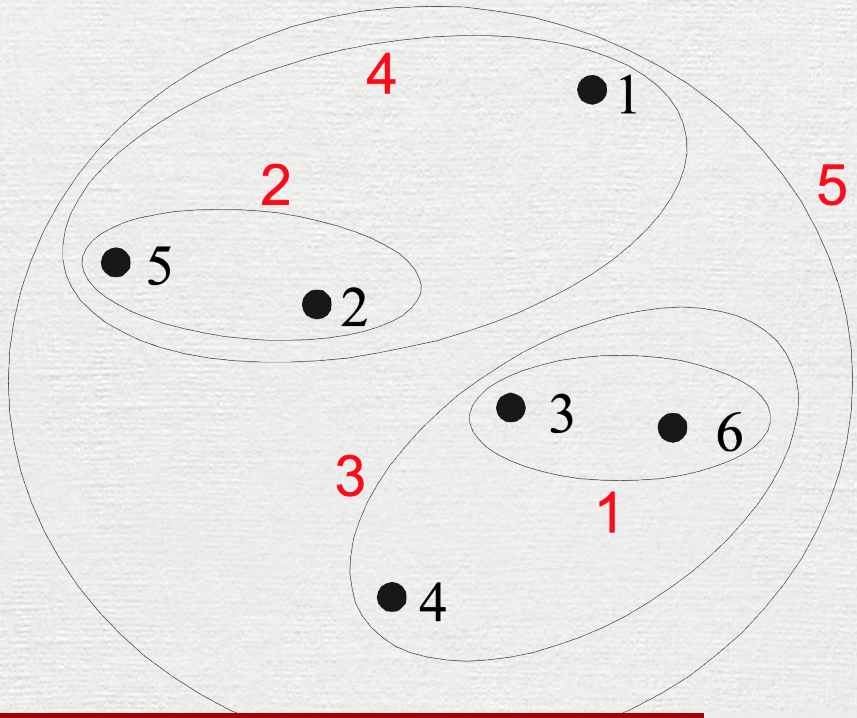


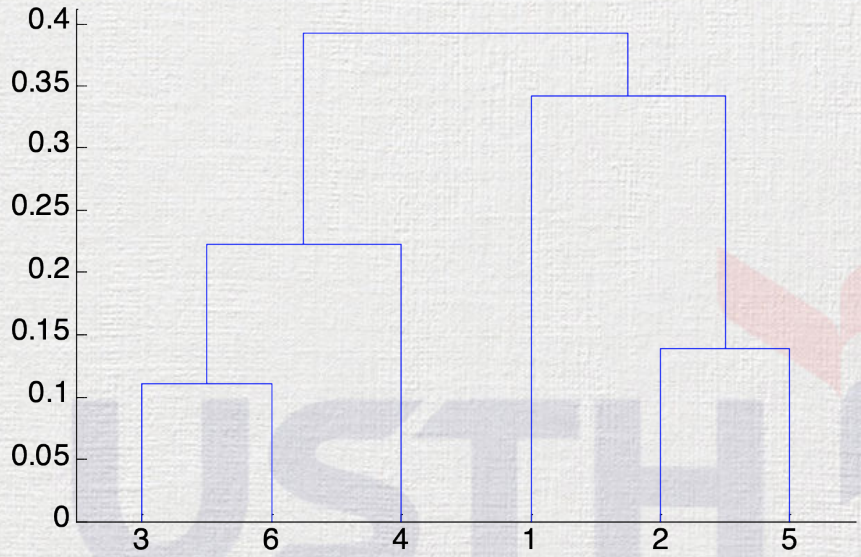
2 clusters

Agglomerative Hierarchical Clustering

- Max

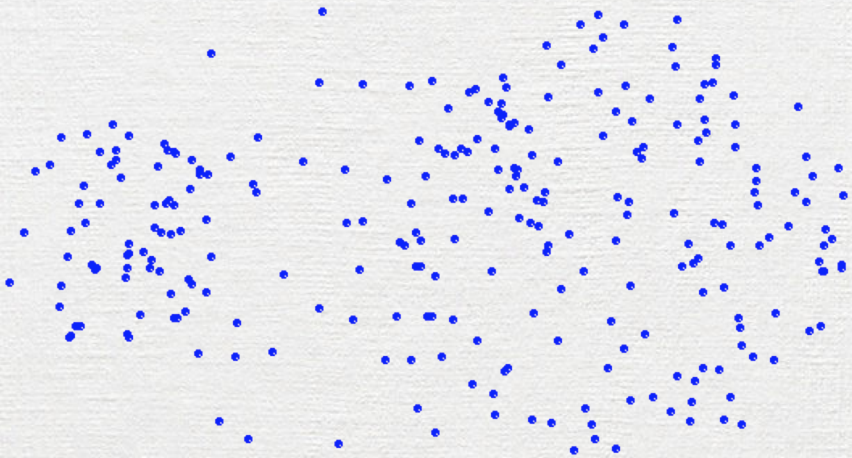
	1	2	3	4	5	6
1	0	.24	.22	.37	.34	.23
2	.24	0	.15	.20	.14	.25
3	.22	.15	0	.15	.28	.11
4	.37	.20	.15	0	.29	.22
5	.34	.14	.28	.29	0	.39
6	.23	.25	.11	.22	.39	0



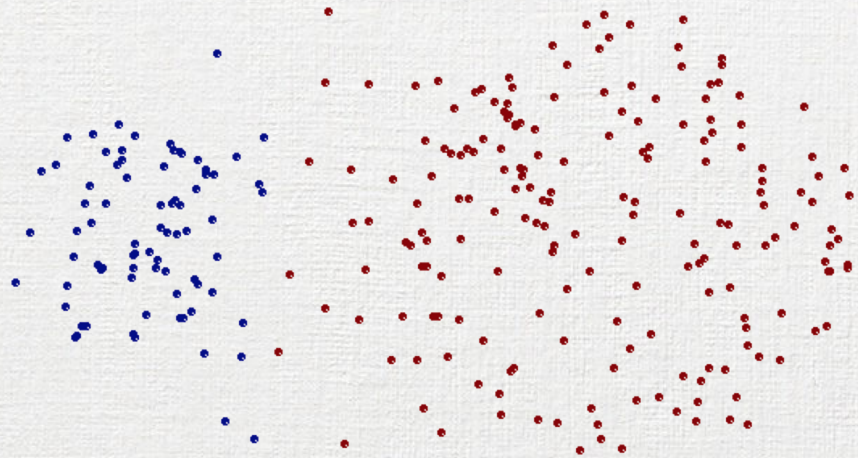


Agglomerative Hierarchical Clustering

- Min: less sensitive to noise...



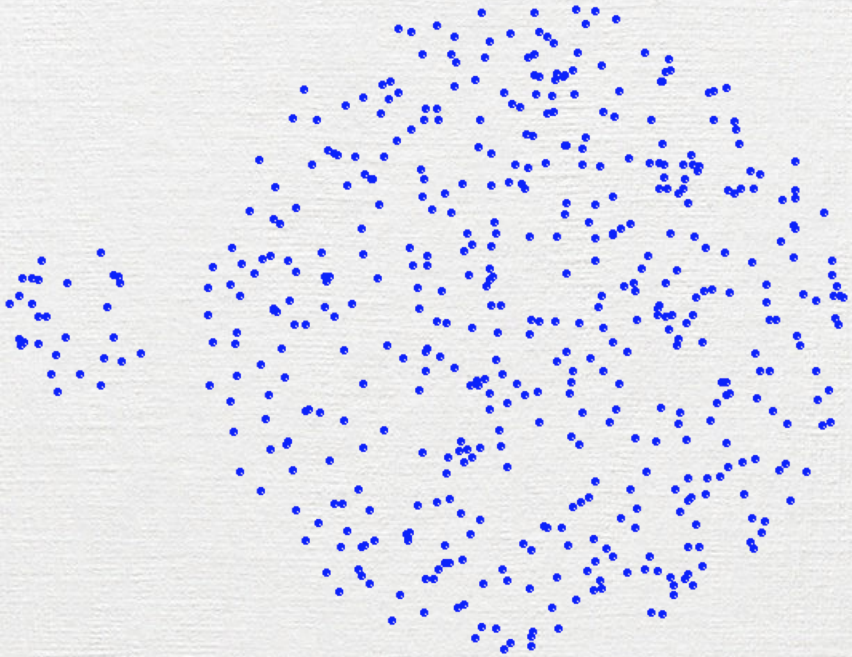
Input



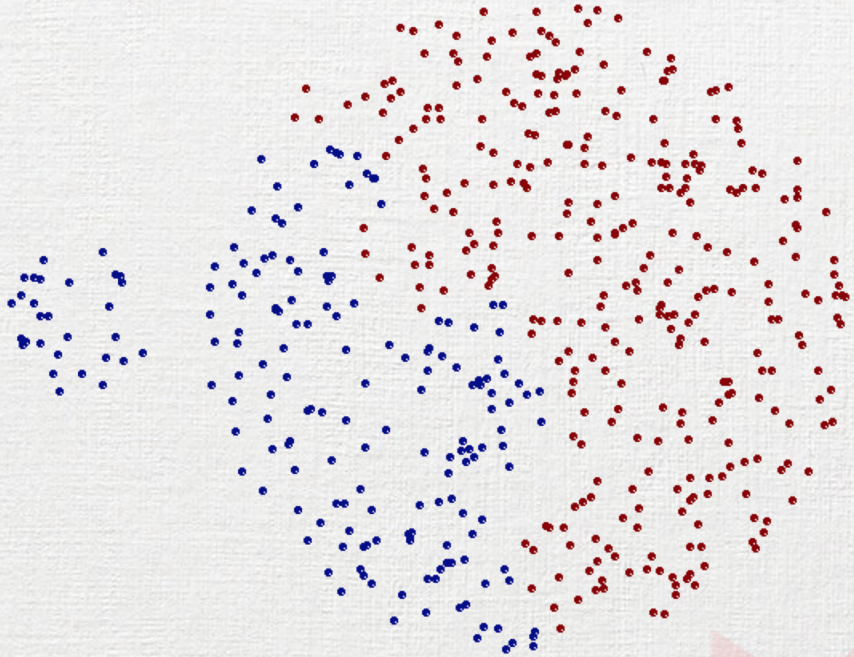
2 clusters

Agglomerative Hierarchical Clustering

- Min: . . . but can break large clusters



Input



2 clusters

Agglomerative Hierarchical Clustering

- Average: average of pairwise distance between points in two clusters

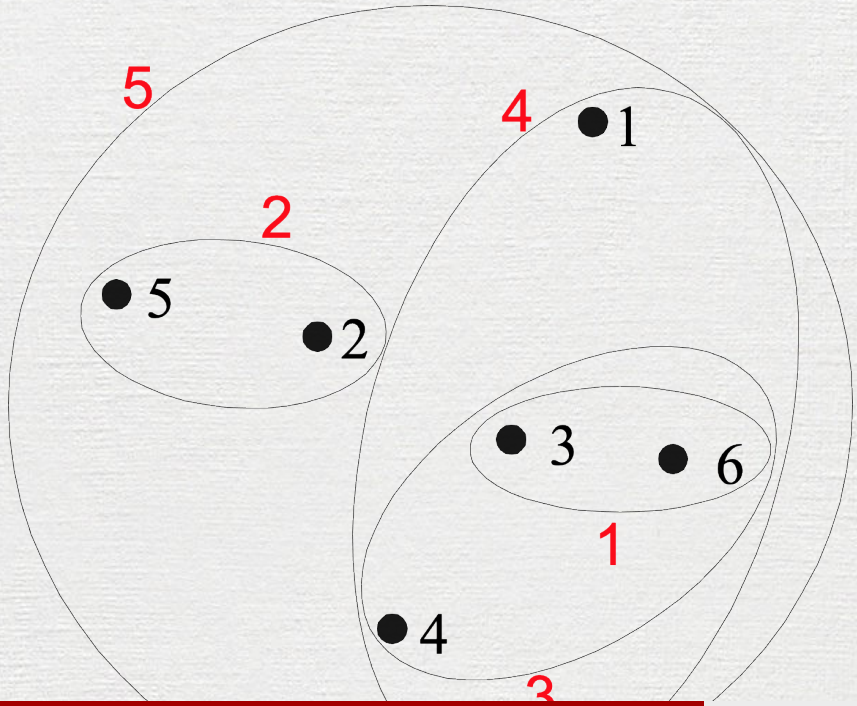
$$d(c_i, c_j) = \frac{\sum_{p_i \in c_i, p_j \in c_j} d(p_i, p_j)}{\mid c_i \mid * \mid c_j \mid} \quad (1)$$

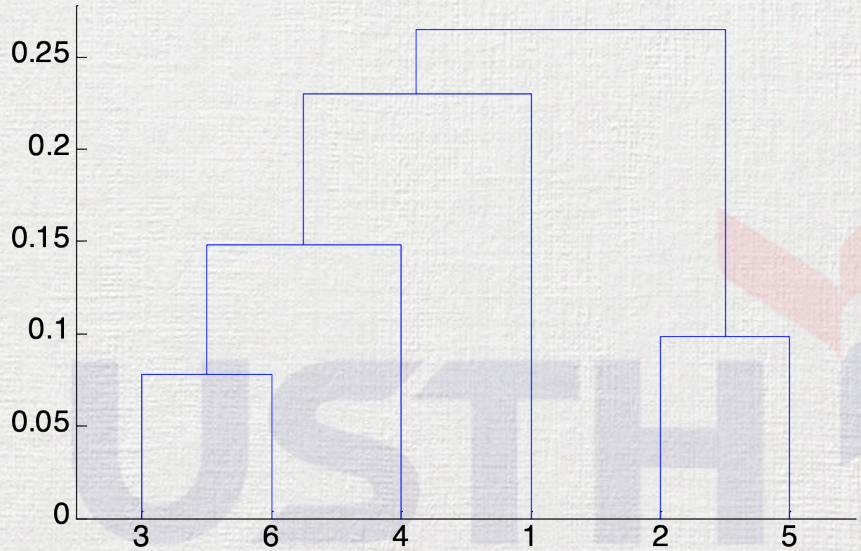
- More complex in time

Agglomerative Hierarchical Clustering

- Average

	1	2	3	4	5	6
1	0	.24	.22	.37	.34	.23
2	.24	0	.15	.20	.14	.25
3	.22	.15	0	.15	.28	.11
4	.37	.20	.15	0	.29	.22
5	.34	.14	.28	.29	0	.39
6	.23	.25	.11	.22	.39	0

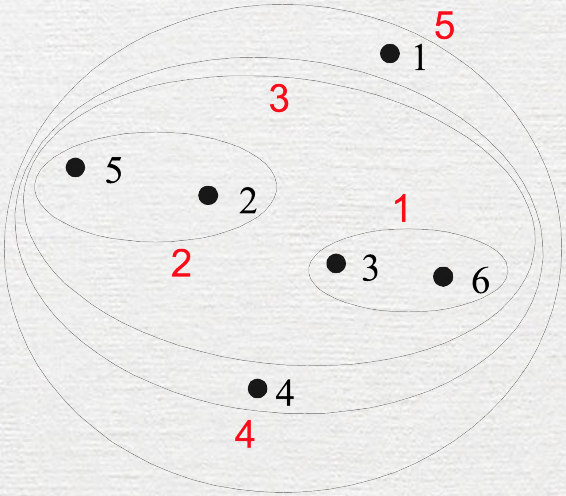




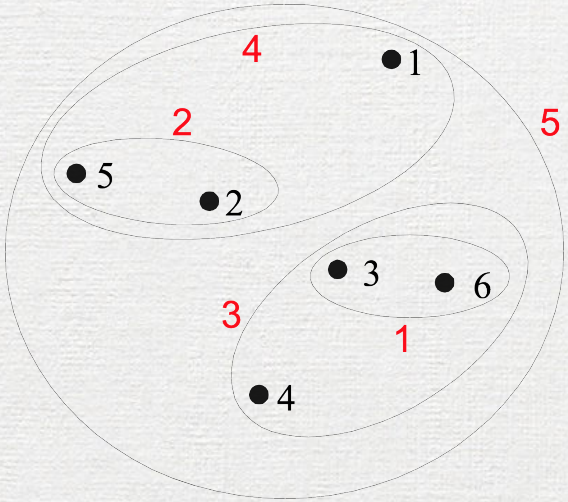
Agglomerative Hierarchical Clustering

- Average
  - Less sensitive to noise and outliers
  - Biased toward globular clusters

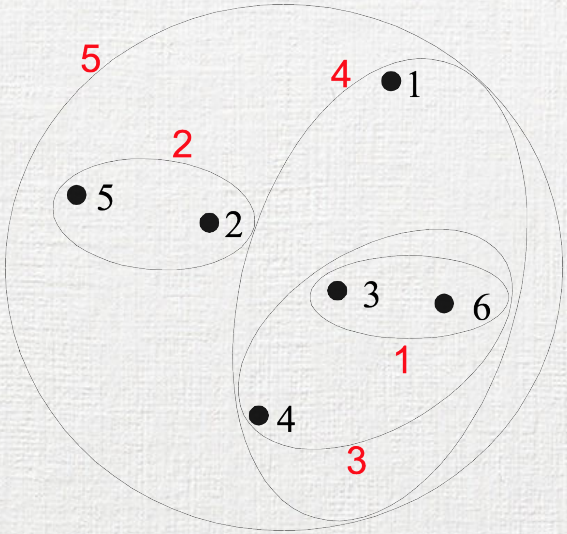
Agglomerative Hierarchical Clustering: Summary



Min



Max



Average

Agglomerative Hierarchical Clustering: Complexity

- Space: $O(N^2)$
  - N: number of points
- Time: $O(N^3)$
  - N steps
  - N^2 updating proximity matrix

Practical work 3: Hierarchical Clustering

- Implement hierarchical clustering
  - Min, max
  - Cluster the reviews into 3 clusters using its length
    - Expected: short, medium, long reviews
  - Name your source code «03.review.length.clustering.py»
- Push your code to corresponding forked Github repository

K-means

What

- Partitional clustering method
  - Assigns items to each cluster
  - Find mutually exclusive cluster of spherical shape based on distance
    - One data point belongs to only one cluster
- K: number of clusters, predefined parameter
- Centroid: center point of a cluster

Why

- Simple!
- Less computationally intensive
- Can be suitable for large dataset
- Guaranteed convergence, quite fast

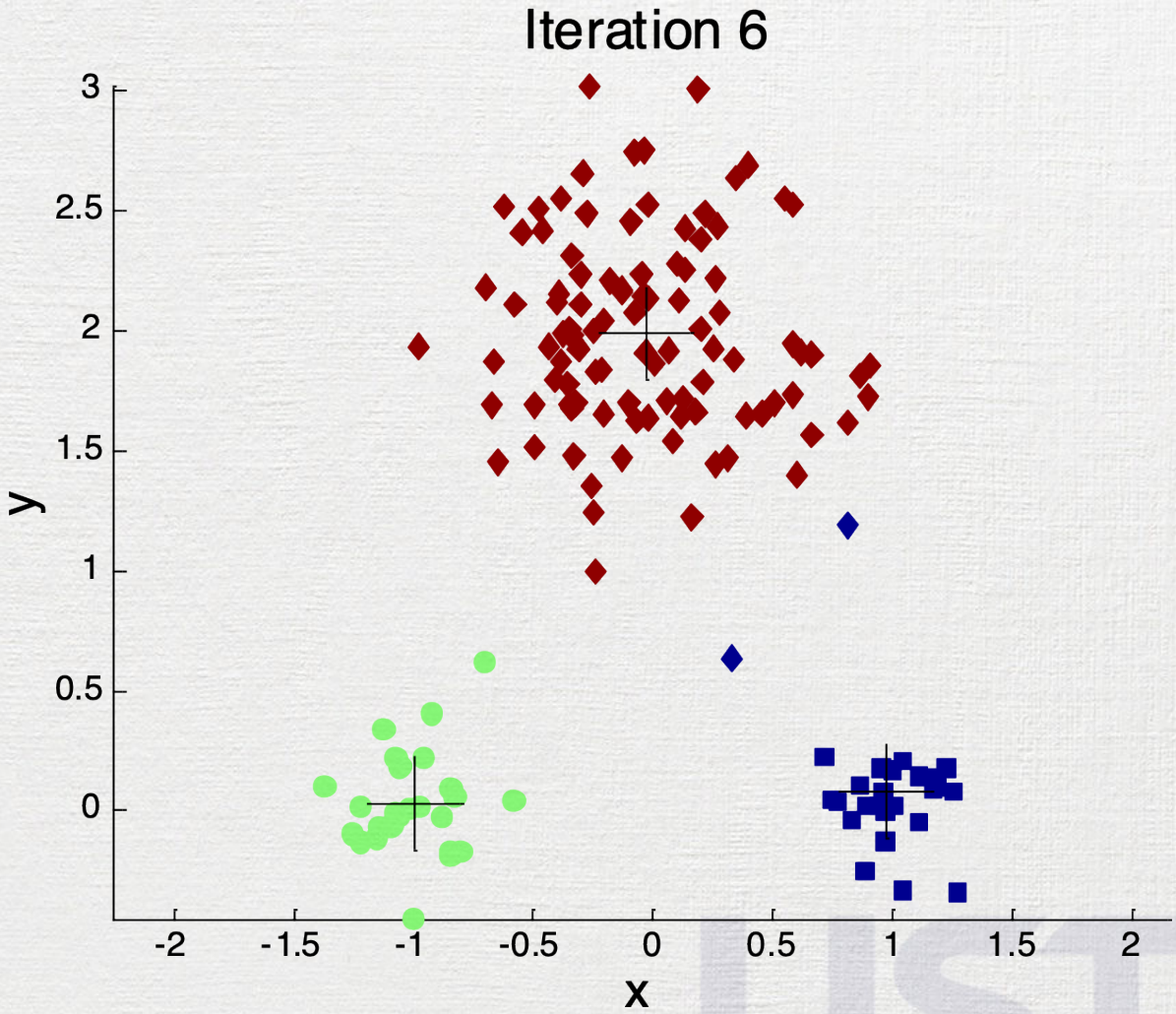
How

Algorithm 1: K-means clustering method

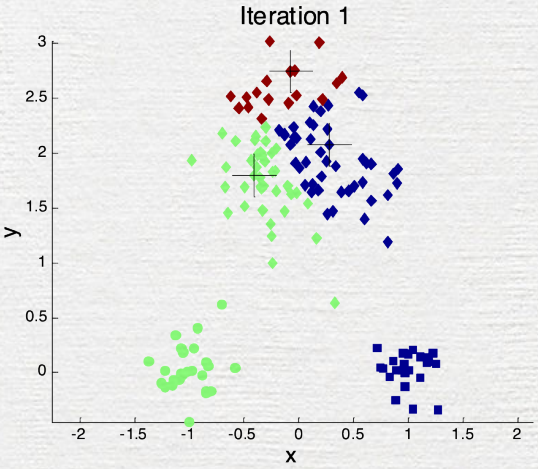
Input : Set of points P_i
Output: Set of clusters $\hat{C}$

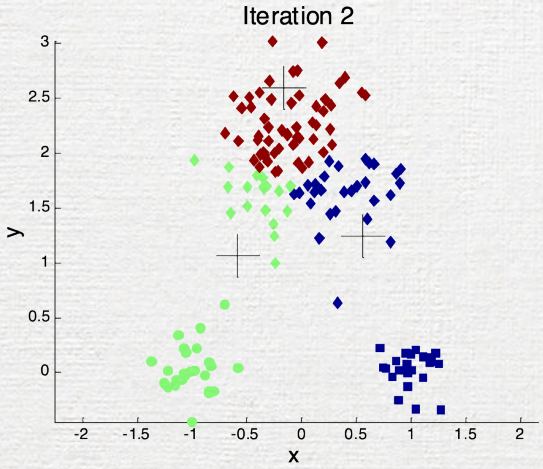
1 Select K initial centroids;
2 **repeat**
3 | Assign all points to the closest centroid;
4 | Recompute centroids;
5 **until** *Centroids don't change*;

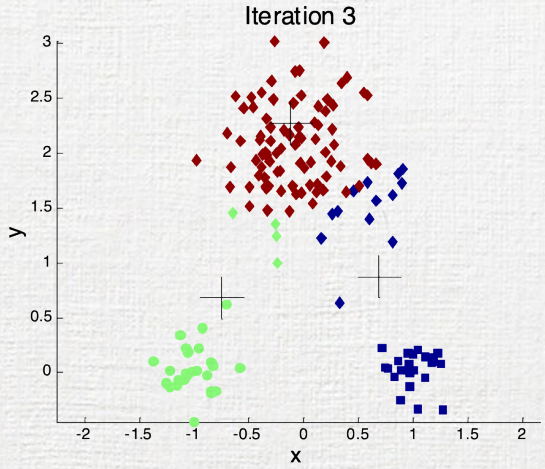
Example

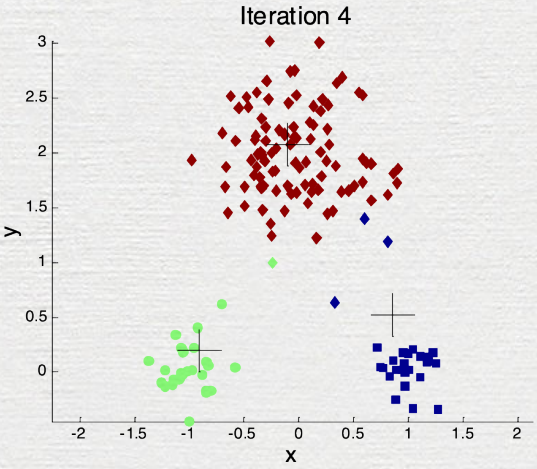


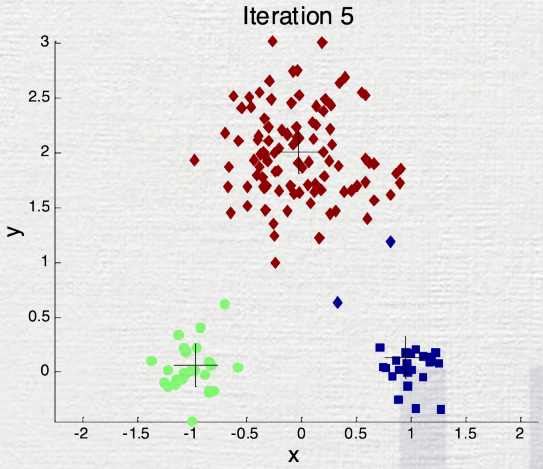
Example

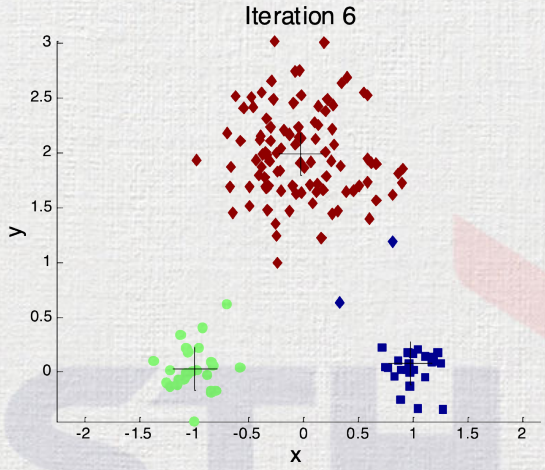

Iteration 1
x
y


Iteration 2
x
y


Iteration 3
x
y


Iteration 4
x
y


Iteration 5
x
y


Iteration 6
x
y

How

- Time complexity: $O(n * K * I * d)$
  - n: number of points
  - K: number of clusters
  - I: number of iterations
  - d: number of attributes

Formulation

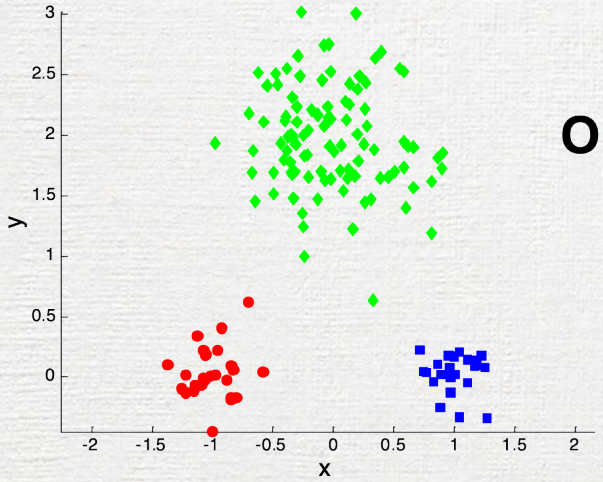
- K-means as an optimization problem
  - Find centroids m_i of cluster C_i
  - Subject to minimizing Sum of Squared Errors (SSE)

$$\text{Minimize } SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(x, m_i) \tag{2}$$

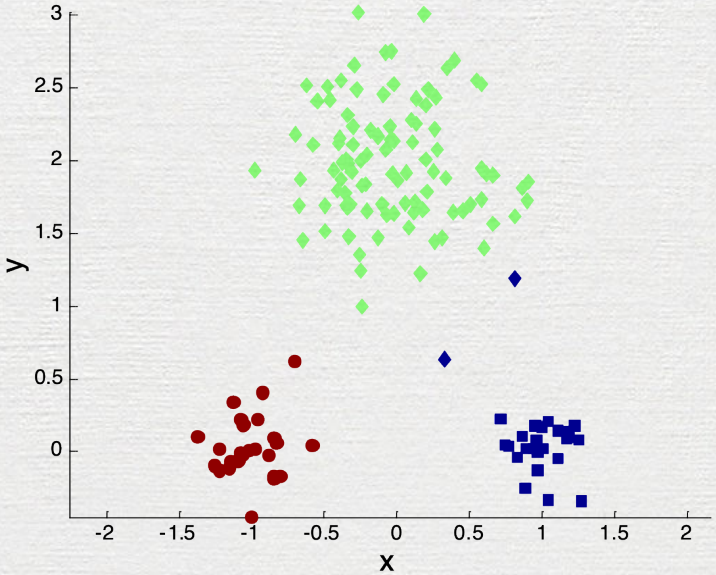
How

- Selecting initial centroids
  - Hardcoded (!)
  - Randomly
  - Farthest
- Centroid: center of a cluster
  - Mean
  - Median
- Distance between points
  - Manhattan (L1)
  - Euclidean (L2)

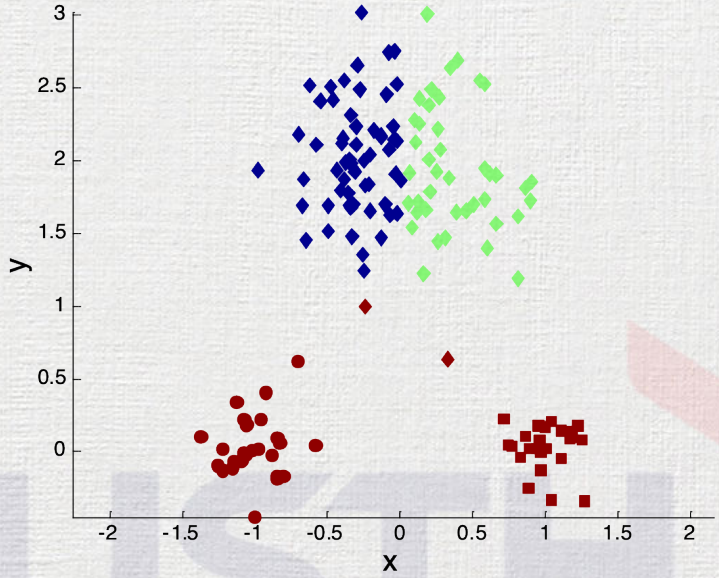
Importance of initial centroids



Original Points

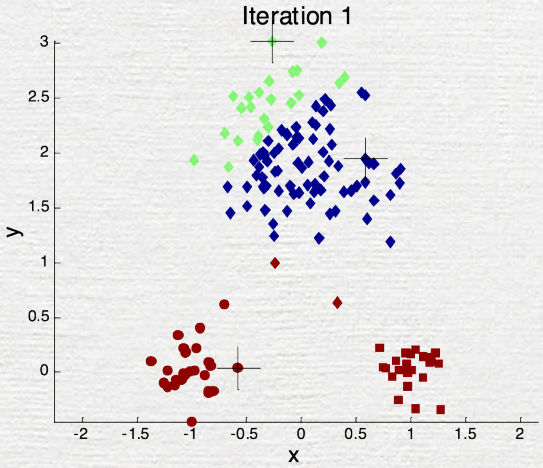


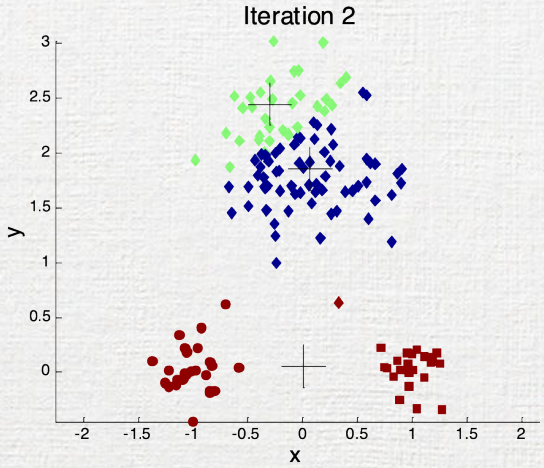
Optimal Clustering

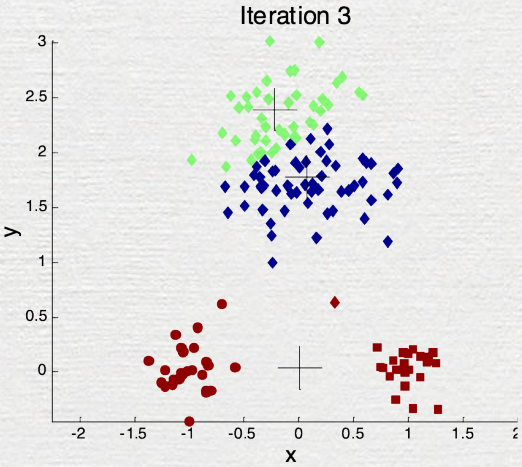


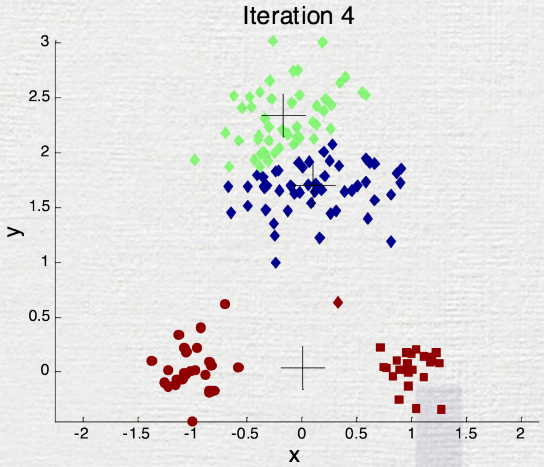
Sub-optimal Clustering

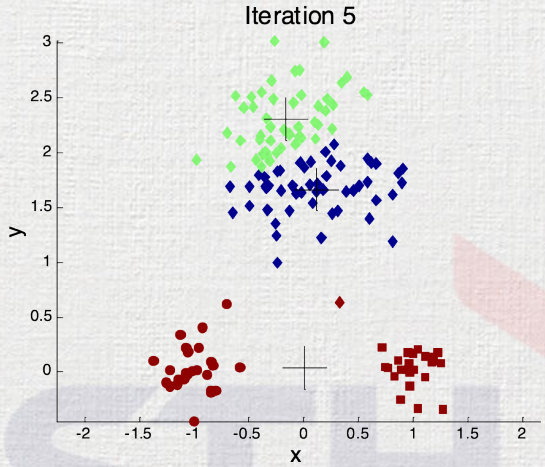
Importance of initial centroids











Practical work 4: K-means Clustering

- Implement K-means clustering
  - Random initial centroids
  - Mean centroids
  - Euclidean distance
  - Cluster the reviews into 3 clusters using its length
    - Expected: short, medium, long reviews
  - Compare the results with practical work 3
  - Name your source code «04.review.length.kmeans.py»
- Push your code to corresponding forked Github repository