INT104 ARTIFICIAL INTELLIGENCE

L9- Unsupervised Learning I
Clustering

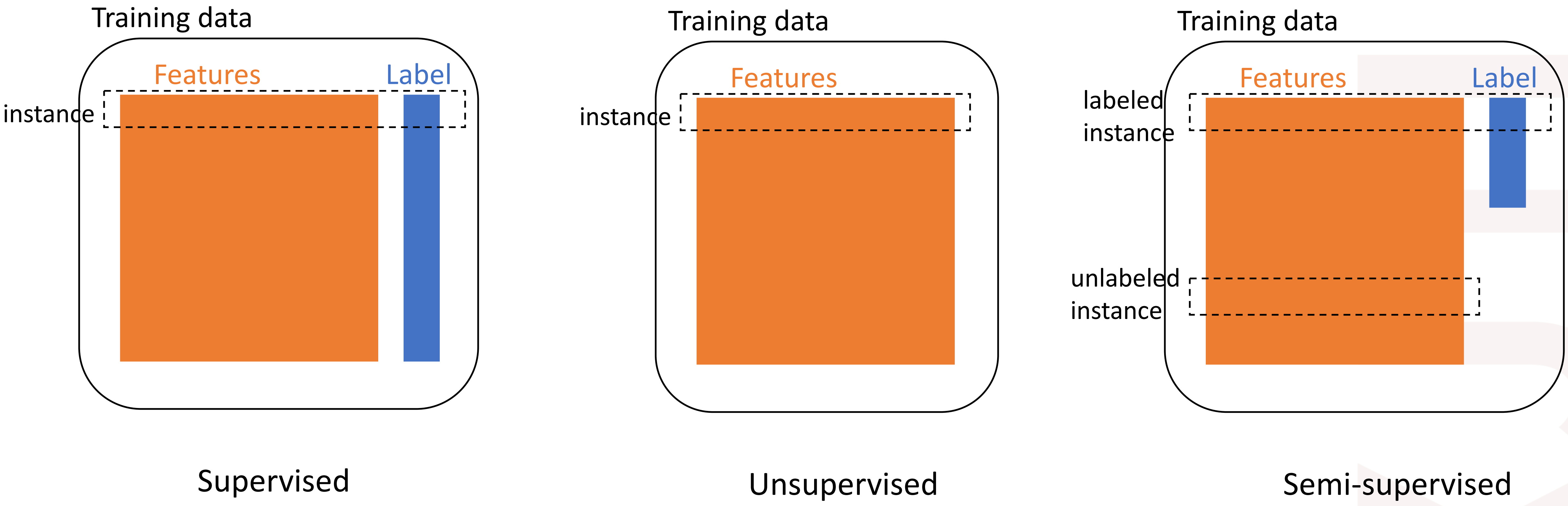
Fang Kang
Fang.kang@xjtlu.edu.cn

Xi'an Jiaotong-Liverpool University
西交利物浦大學

CONTENT

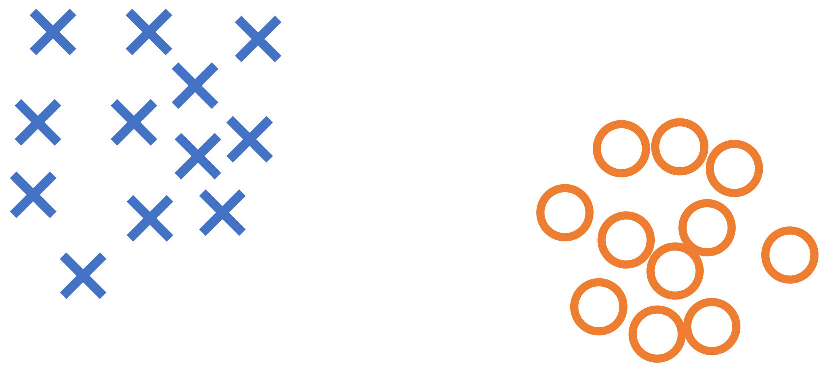
- K-Means
  - K-means clustering
  - Centroid initialization methods
  - Parameters and Evaluation
- Hierarchical Clustering
- DBSCAN

Supervised vs. unsupervised



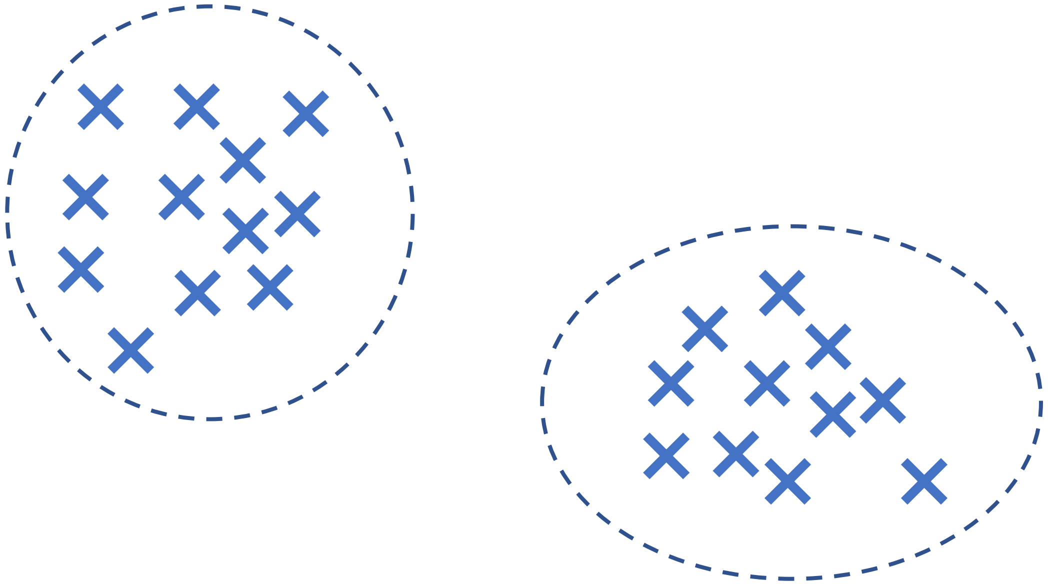
Supervised learning

- The correct labels for each training example are known



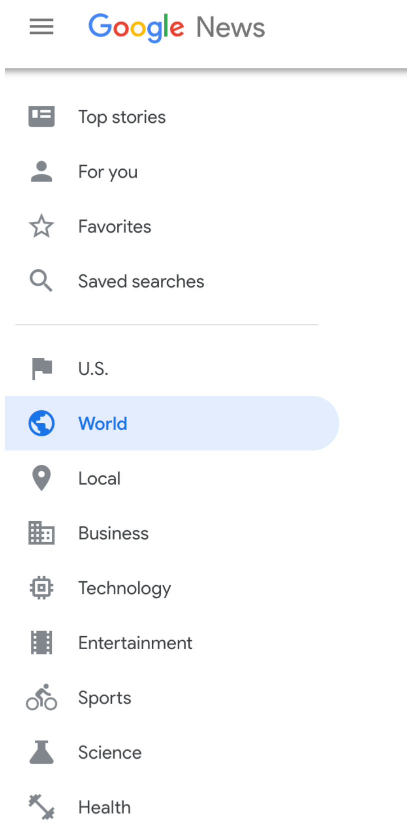
Unsupervised learning

Labels are unknown → Need to automatically discover the ***clustering*** pattern and structure in data



Real world clustering example

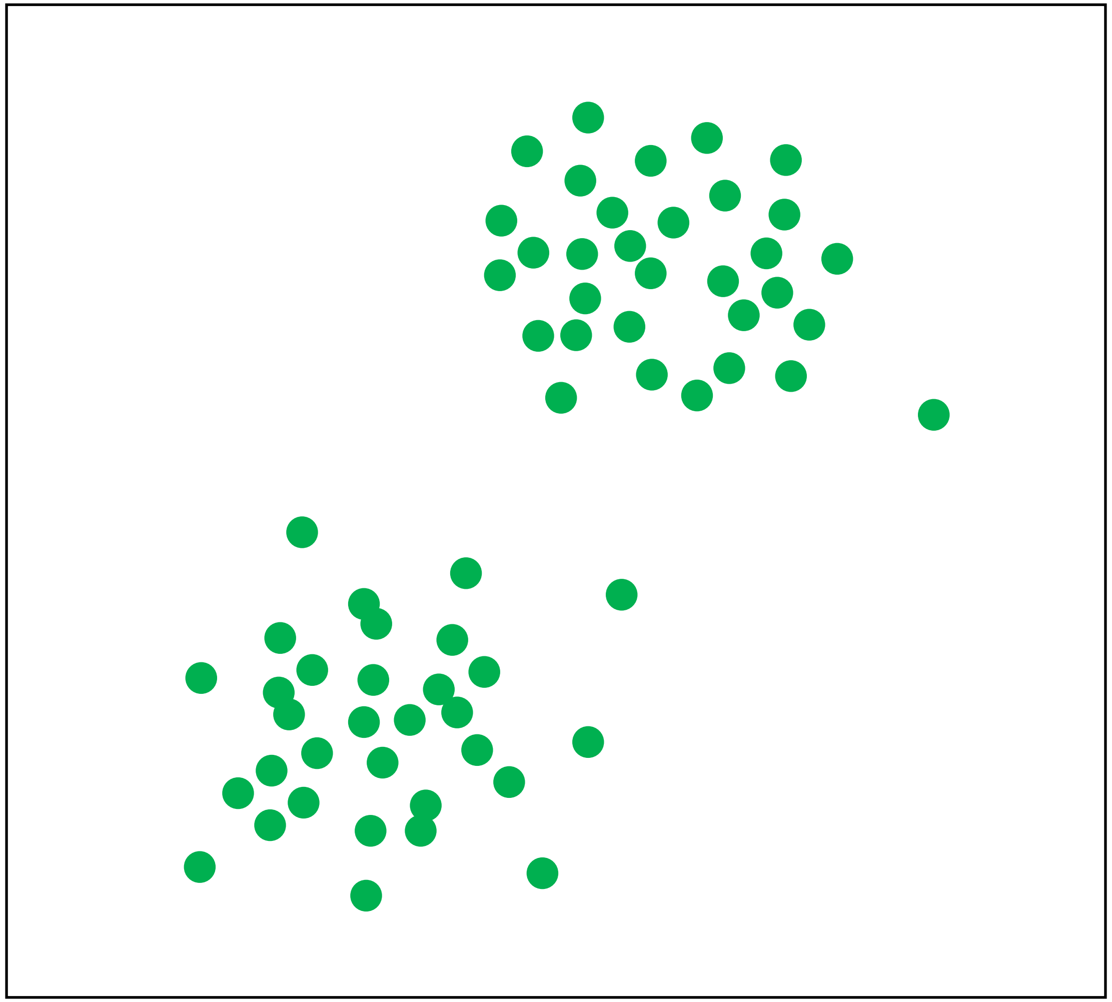
- Clustering on text documents



- Image segmentation

- Streaming services often use clustering analysis to identify viewers who have similar behavior.

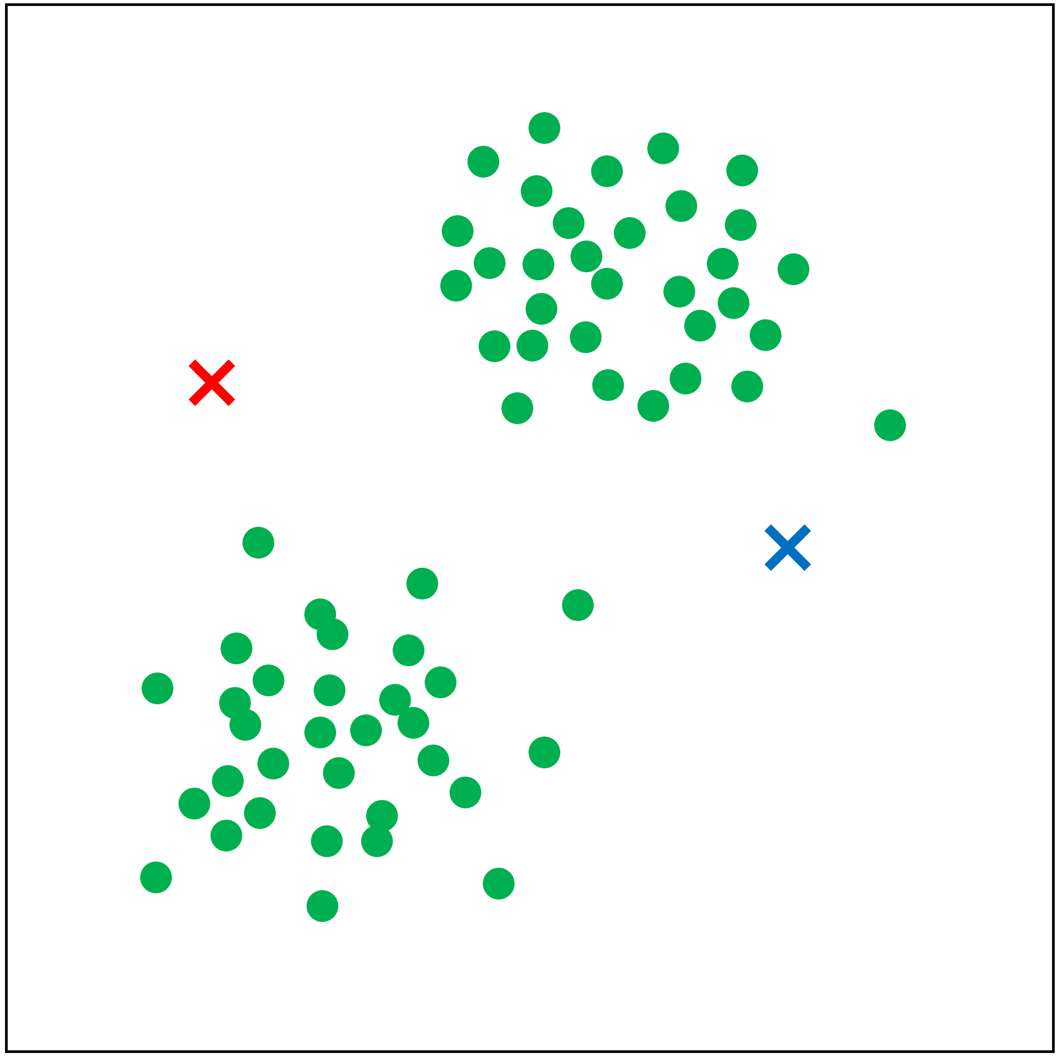
K-means clustering algorithm



Term first used by James MacQueen, 1967

Algorithm proposed by Stuart Lloyd, 1957, published in 1982.

K-means clustering algorithm

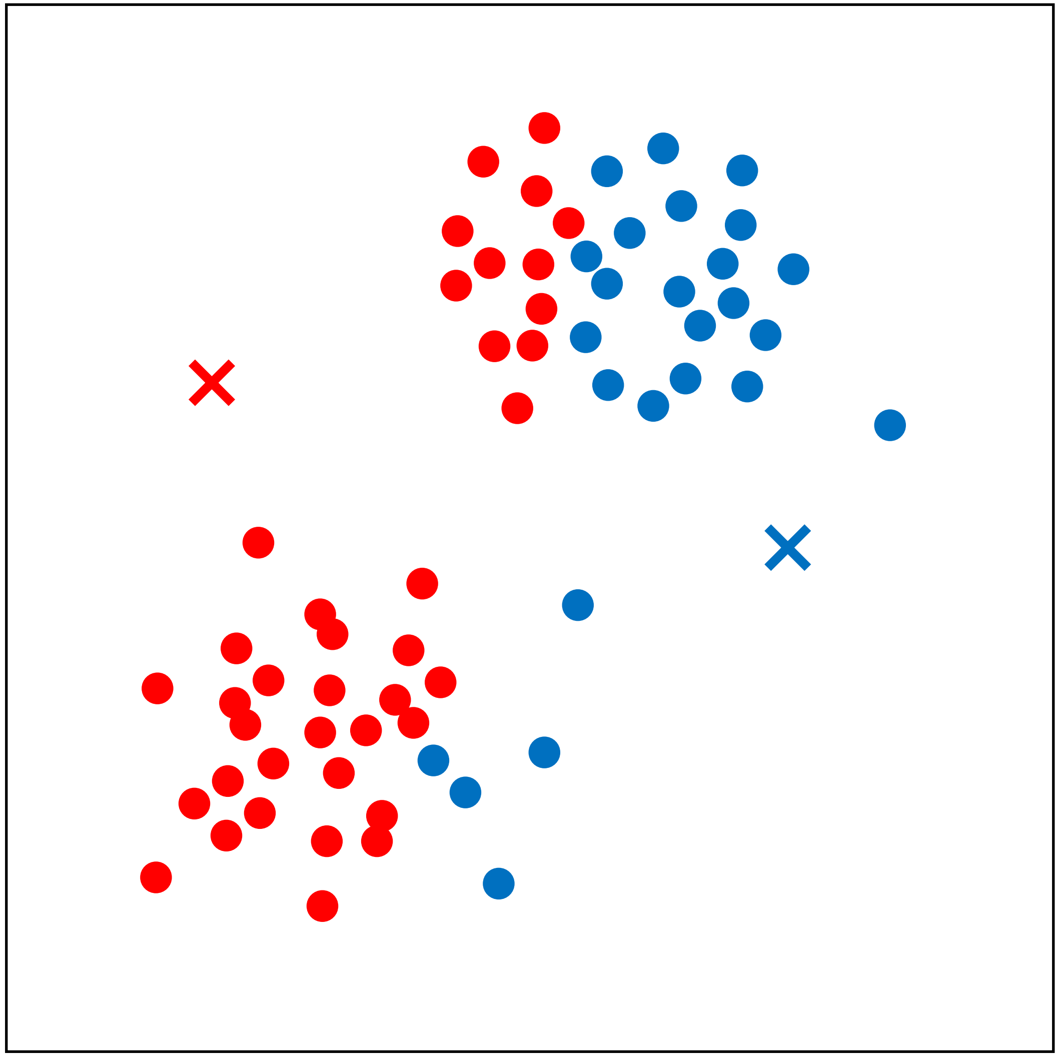


Goal: Assign all data points to 2 clusters

Step 1: Pick 2 *random* initial cluster centroids

Step 2: Paint the data points that are closer to red centroid red, and those closer to blue centroid blue

K-means clustering algorithm



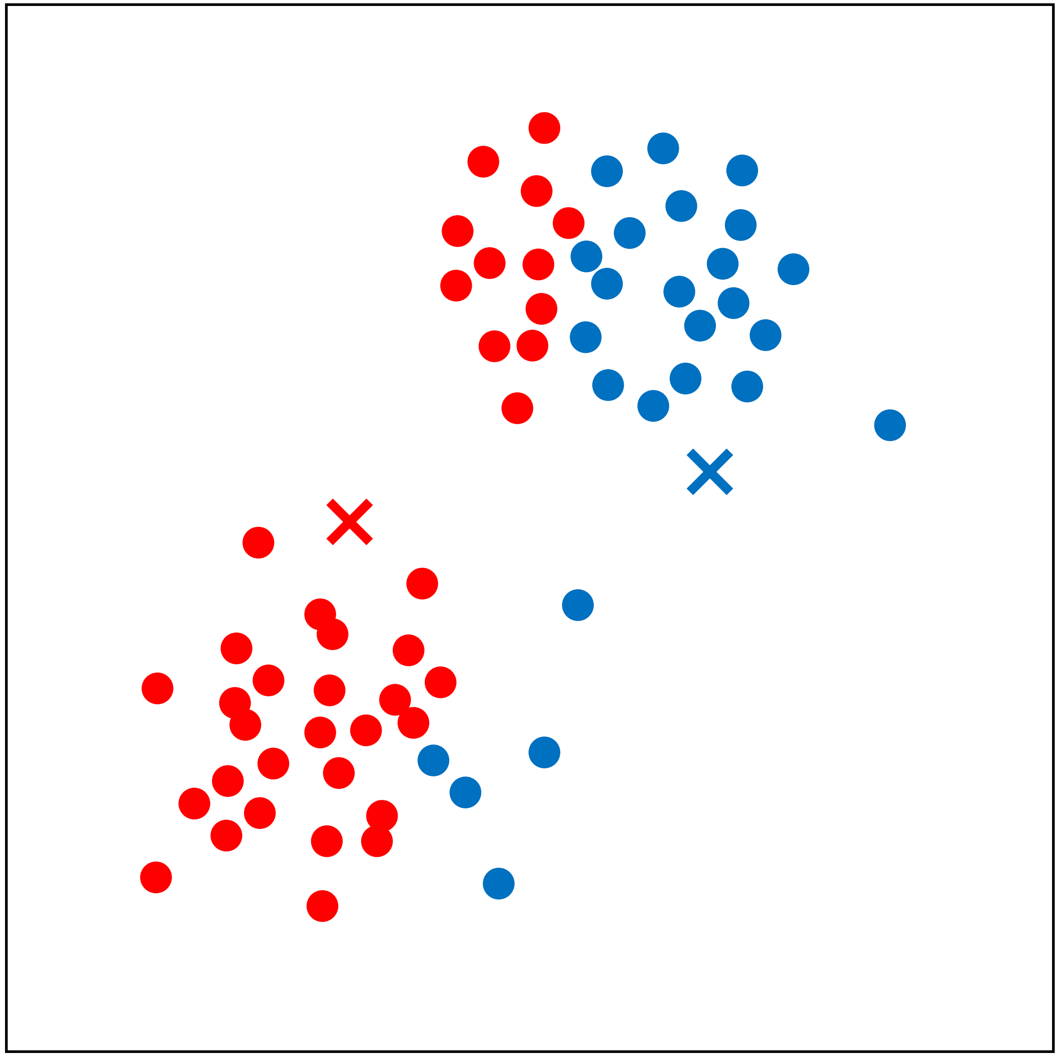
Goal: Assign all data points to 2 clusters

Step 1: Pick 2 *random* initial cluster centroids

Step 2: Paint the data points that are closer to red centroid red, and those closer to blue centroid blue

Step 3: Update the positions of centroids
Red centroid := average of current red points
Blue centroid := average of current blue points

K-means clustering algorithm



Goal: Assign all data points to 2 clusters

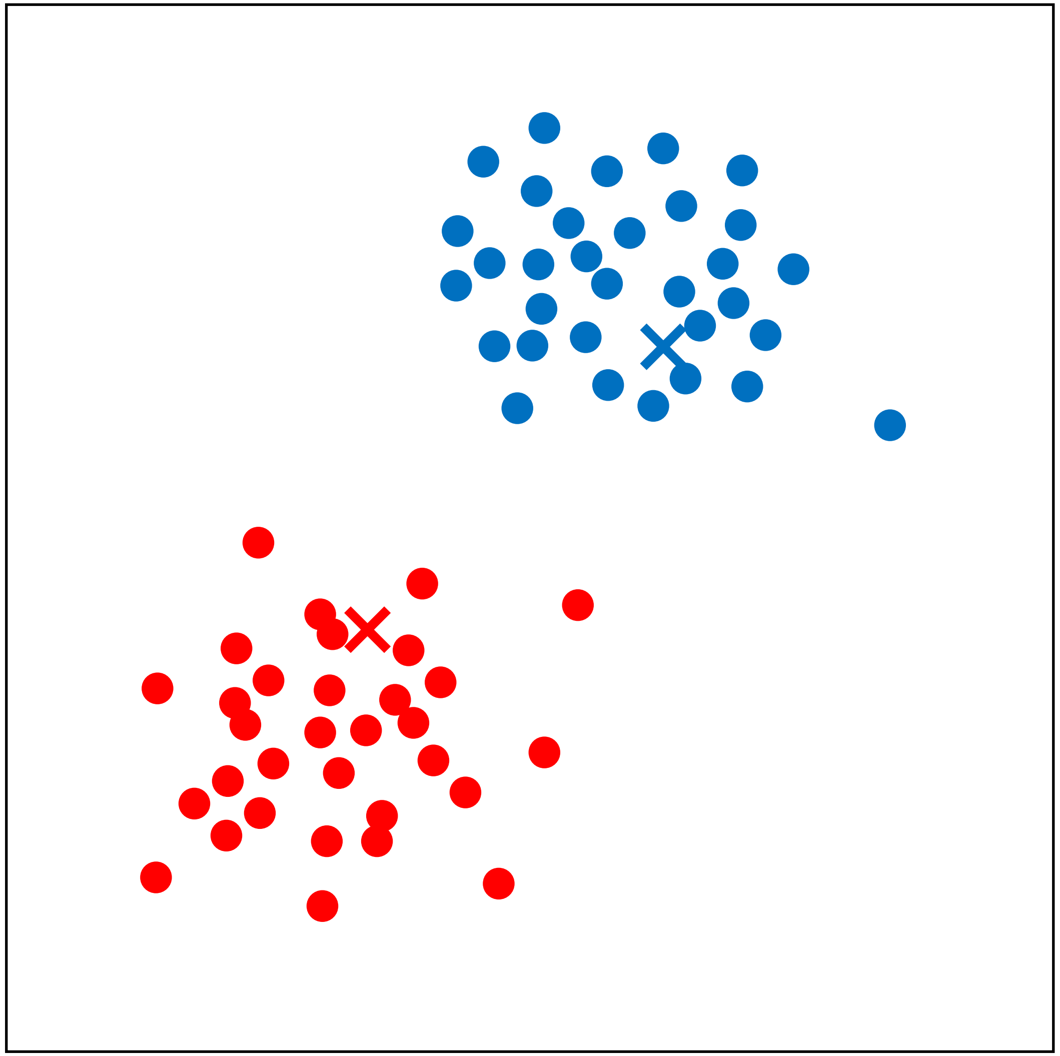
Step 1: Pick 2 *random* initial cluster centroids

Step 2: Paint the data points that are closer to red centroid red, and those closer to blue centroid blue

Step 3: Update the positions of centroids
Red centroid := average of current red points
Blue centroid := average of current blue points

Repeat

K-means clustering algorithm



Goal: Assign all data points to 2 clusters

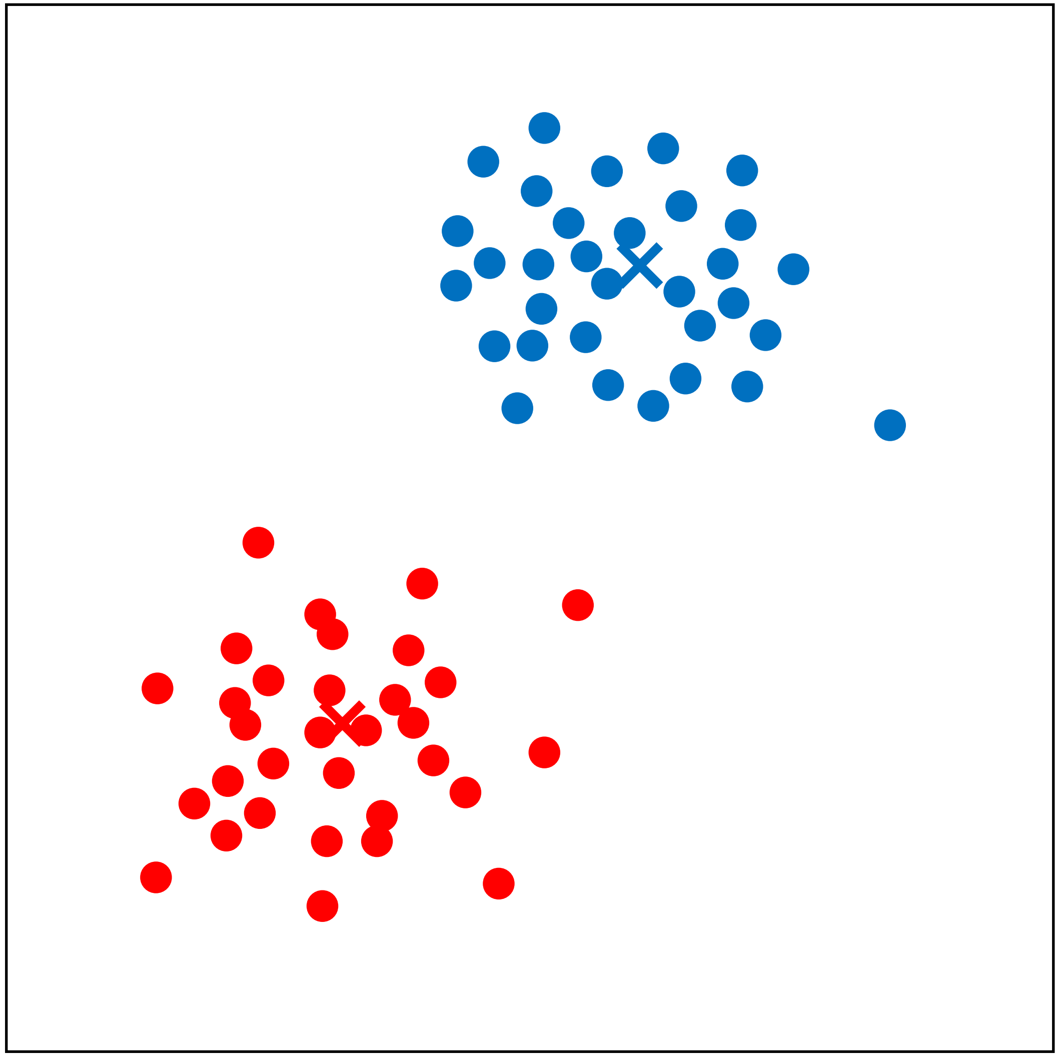
Step 1: Pick 2 *random* initial cluster centroids

Step 2: Paint the data points that are closer to red centroid red, and those closer to blue centroid blue

Step 3: Update the positions of centroids
Red centroid := average of current red points
Blue centroid := average of current blue points

Repeat

K-means clustering algorithm



Goal: Assign all data points to 2 clusters

Step 1: Pick 2 *random* initial cluster centroids

Step 2: Paint the data points that are closer to red centroid red, and those closer to blue centroid blue

Step 3: Update the positions of centroids

Red centroid := average of current red points

Blue centroid := average of current blue points

Repeat

Until no more pointes need to be repainted, i.e., the centroids no longer change

Clustering is done

K-means formal definition

Given a dataset $\{x^{(1)}, \cdots, x^{(m)}\}$, $x^{(i)} \in \mathbb{R}^n$, and want to group the data into k clusters

1. Initialize k **cluster centroids** $\mu_1, \mu_2, \cdots, \mu_k \in \mathbb{R}^n$ randomly

2. Repeat until convergence: {

For $i = 1, \cdots, m$:

$$c^{(i)} := \arg\min_j \left\| x^{(i)} - \mu_j \right\|^2 \longrightarrow$$

Usually, Euclidean distance is the best option

Assign $x^{(i)}$ to the closest cluster j

For $j = 1, \cdots, k$:

$$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)}=j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)}=j\}} \longrightarrow$$

Update centroid μ_j with mean of all within-cluster data points

}

The City block distance between two points, a and b, with k dimensions is calculated as:

$$\sum_{j=1}^{k} |a_j - b_j|$$

From a machine learning perspective, K-means minimize the cost function:

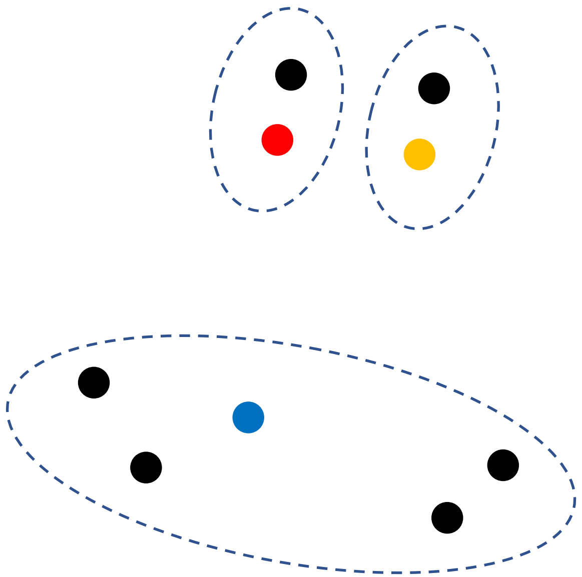
$$J(c, \mu) = \sum_{i=1}^{m} \left\| x^{(i)} - \mu_{c^{(i)}} \right\|^2$$

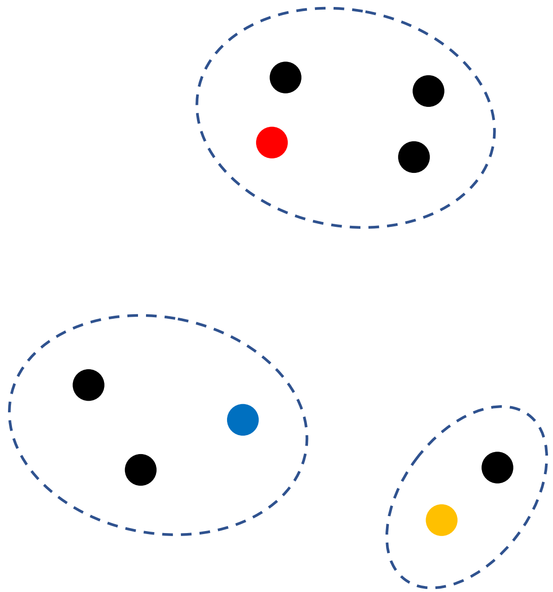
guaranteed to converge

How to initialize centroids?

Randomly pick k data points as the initial centroids

Sometimes it leads to different clustering results





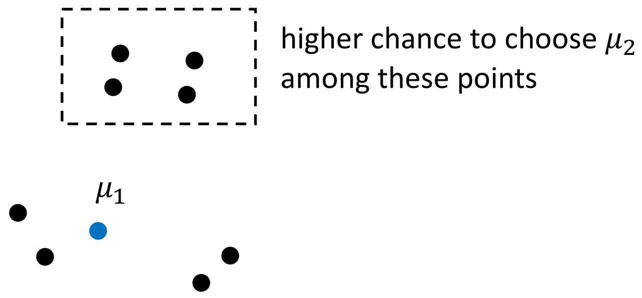
Solutions:

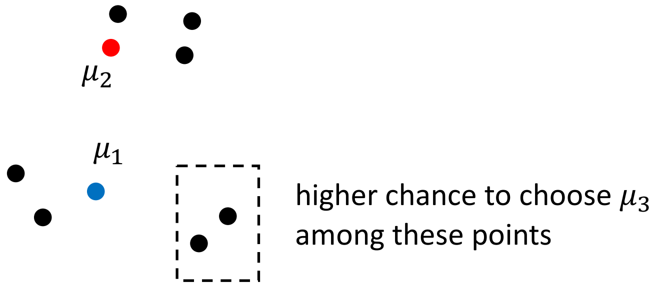
- Run multiple times with different initializations and evaluate
- K-means++ (Arthur & Vassilvitskii, 2007)

Better initialization with *K*-means++

Arthur & Vassilvitskii, 2007

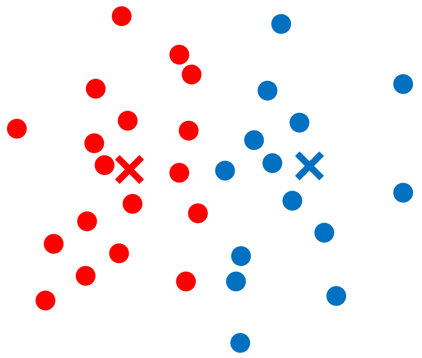
1. Choose one centroid uniformly at random from data points.
2. For each $x^{(i)}$, compute $D(x^{(i)})$, the distance between $x^{(i)}$ and the nearest centroid that has already been chosen.
3. Choose one new data point at random as a new centroid, where the probability of choosing point $x^{(i)}$ is ***proportional*** to $D(x^{(i)})$.
4. Repeat until k centroids have been chosen.



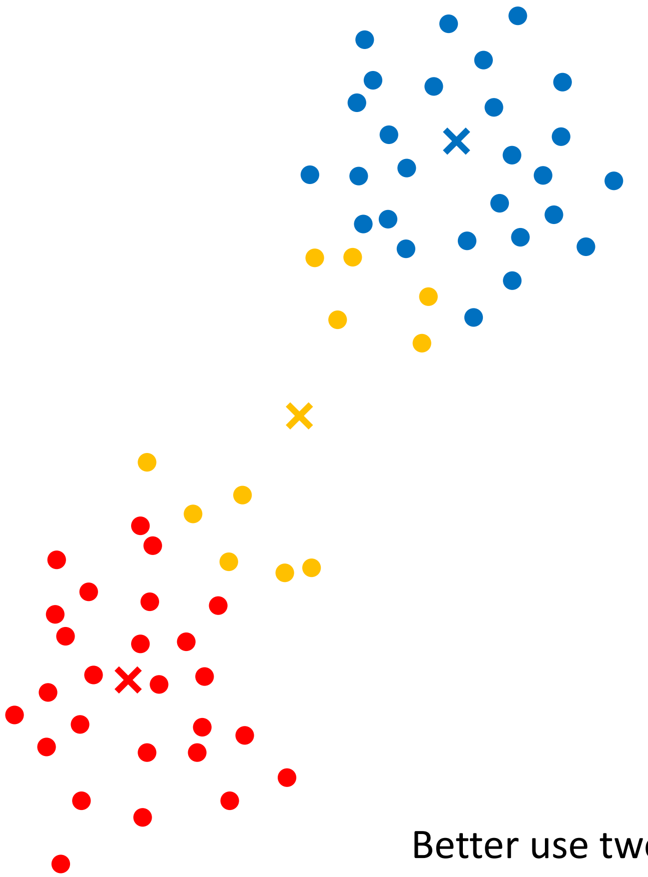


How to choose *K*?

By intuitive observation



Better use one cluster



Better use two clusters

Is there a systematic evaluation?

Parameters and Evaluation

- The number of clusters k is a hyperparameter. How do we find a good k?

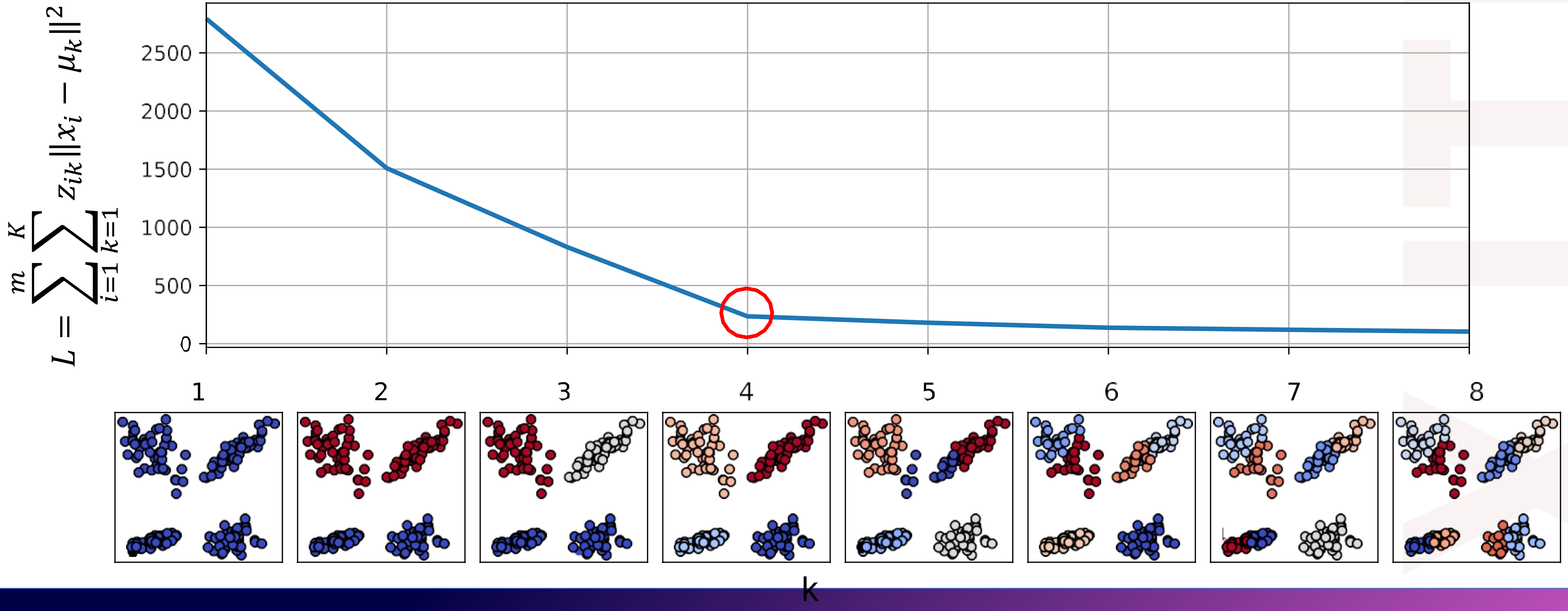
1. Elbow method:
   - ➢ Start with a small k value and increase it until adding another cluster does not result in a much lower distortion value
   - ➢ In other words, the new cluster does not explain so much the variance in data

2. Silhouette Coefficient:
   - ➢ A measure of how tight each cluster is and how far apart clusters are from each other
   - ➢ Choose a value of k that results in clustering with a large silhouette coefficient

Silhouette /ˌsilo͞oˈet/: contour, outline

The Elbow Method

- Choose k such that adding another cluster will not explain the variance in data by much (i.e. does not give a much lower distortion value)



The Silhouette Coefficient

A good clustering algorithm: $\begin{cases} \underline{\textit{high}} \text{ similarity } \textbf{\textit{within}} \text{ cluster} \\ \underline{\textit{low}} \text{ similarity } \textbf{\textit{between}} \text{ cluster} \end{cases}$

- **silhouette coefficient** Measures the tightness of clusters and separation between clusters:

$$S = \frac{1}{m}\sum_{i=1}^{m} s(x_i)$$

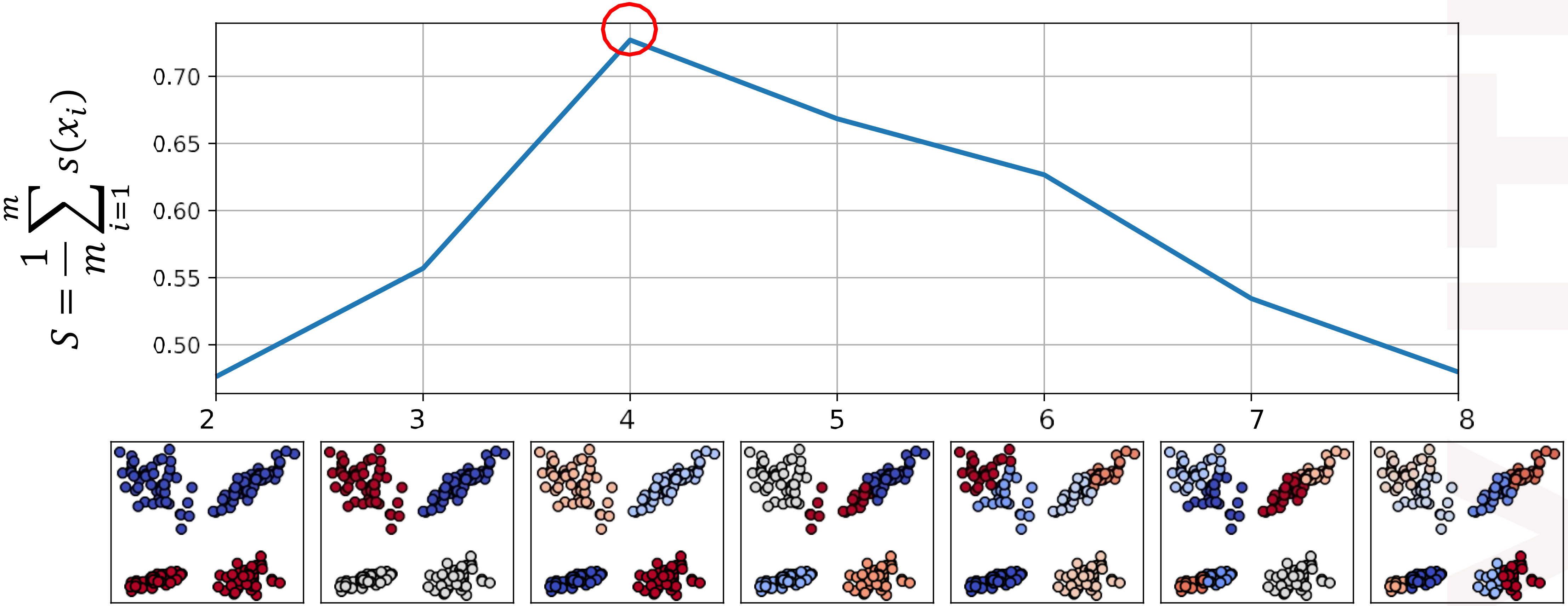
$$s(x_i) = \frac{b(x_i) - a(x_i)}{\max\{a(x_i), b(x_i)\}}$$

where:

- $a(x_i)$ is the average distance between x_i and all other points in the same cluster
- $\mathrm{b}(x_i)$ is the average distance between x_i and all other points in the next neighbor cluster (i.e., the average distance to the nearest neighboring cluster)

The Silhouette Coefficient

- Choose k that gives the highest mean silhouette

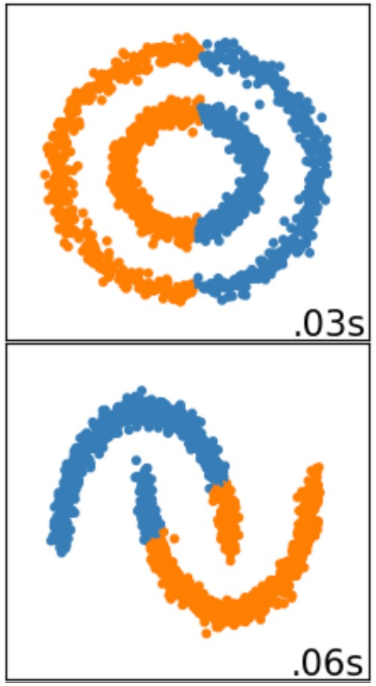


K-means: pros and cons

- Pros:
  - Easy to implement
  - Scales to very large datasets

- Cons:
  - Difficult to choose *K*.
  - Only works on spherical, convex clusters.

Where K-means does not work well



Hierarchical clustering

- Hierarchical Clustering is a set of clustering methods that aim at building a hierarchy of clusters
  - A cluster is composed of smaller clusters

- There are two strategies for building the hierarchy of clusters:
  - Agglomerative (bottom-up): we start with each point in its own cluster and we merge pairs of clusters until only one cluster is formed.
  - Divisive (top-down): we start with a single cluster containing the entire set of points and we recursively split until each point is in its own cluster.

- The most popular strategy in practical use is bottom-up (agglomerative)!

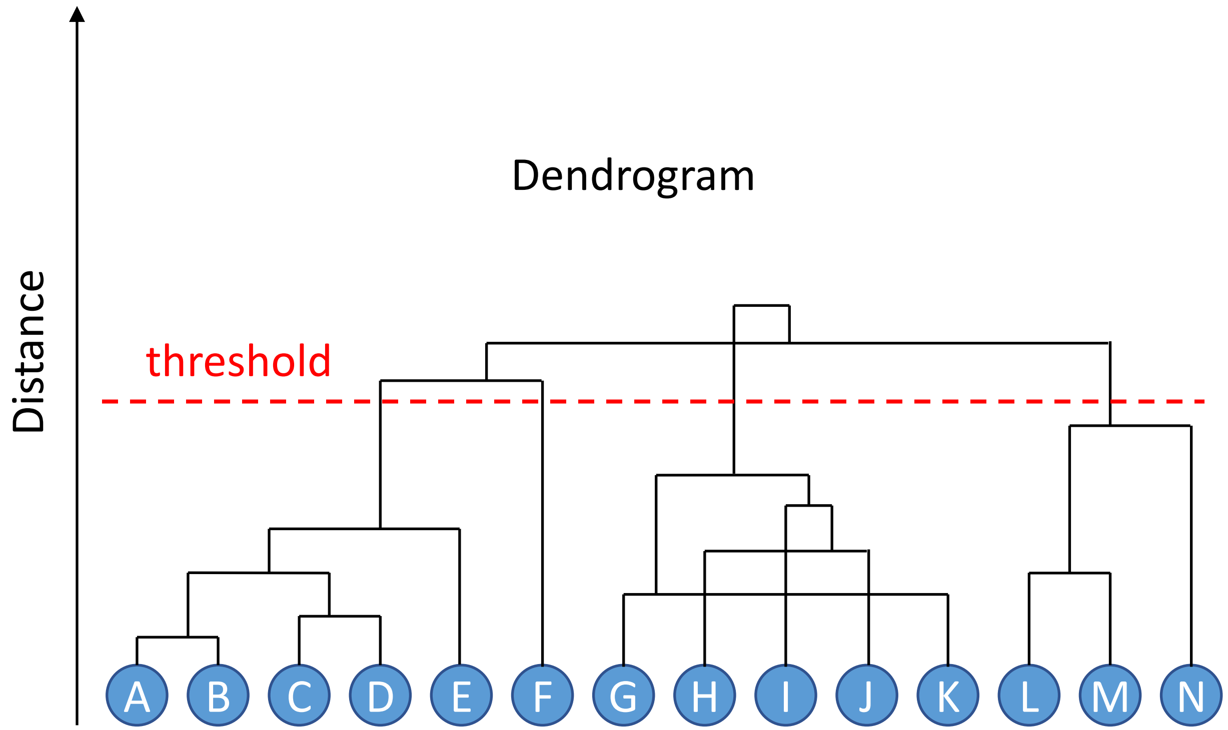
Hierarchical clustering- Agglomerative

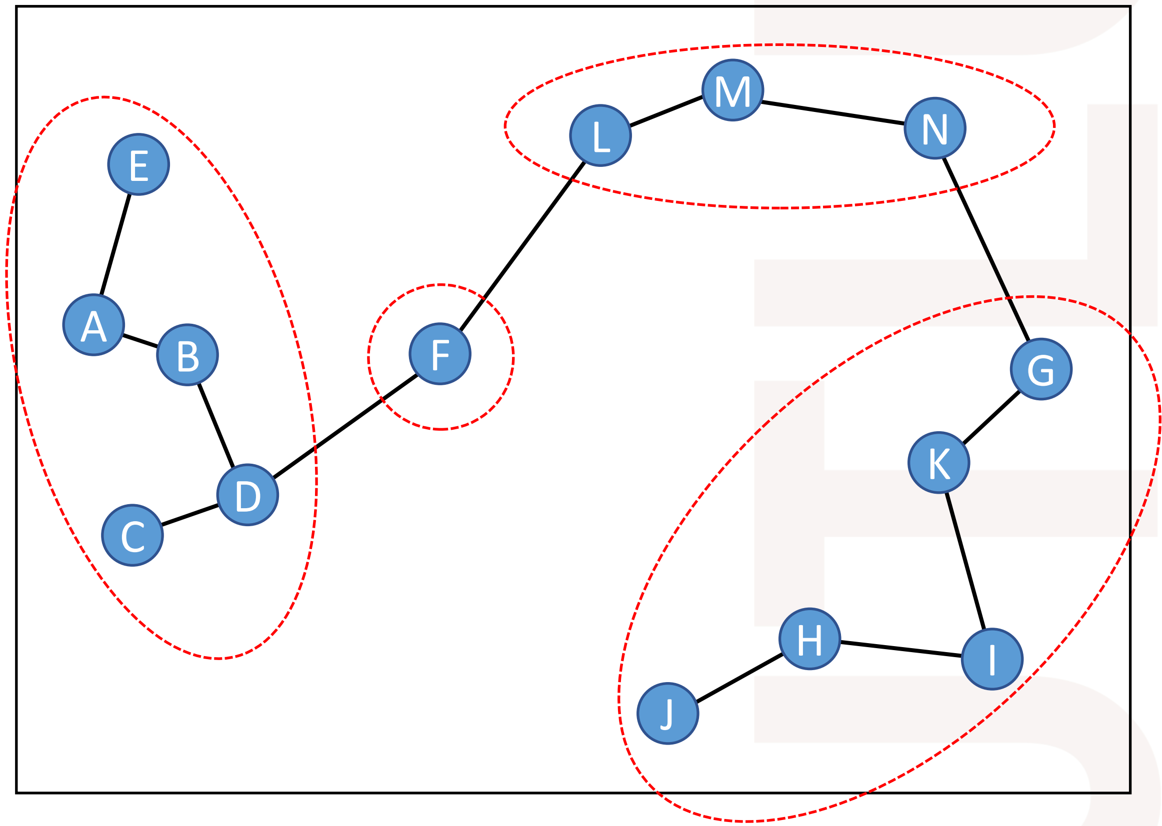
Idea: make sure *nearby* data points end up in the same cluster

- Initialize a collection C of m singleton clusters, i.e., $c^{(i)} = \{x^{(i)}\}$
- Repeat until only one cluster is left:
    - Find a pair of clusters that is closest: $\arg\min_{i,j} D(c^{(i)}, c^{(j)})$
    - Merge the two clusters $c^{(i)}, c^{(j)}$ into a new cluster $c^{(i\&j)}$
    - Remove $c^{(i)}, c^{(j)}$ from the collection C, and add $c^{(i\&j)}$
- Produce a ***dendrogram***: a hierarchical ***tree*** of clusters

Need to define ***distance***

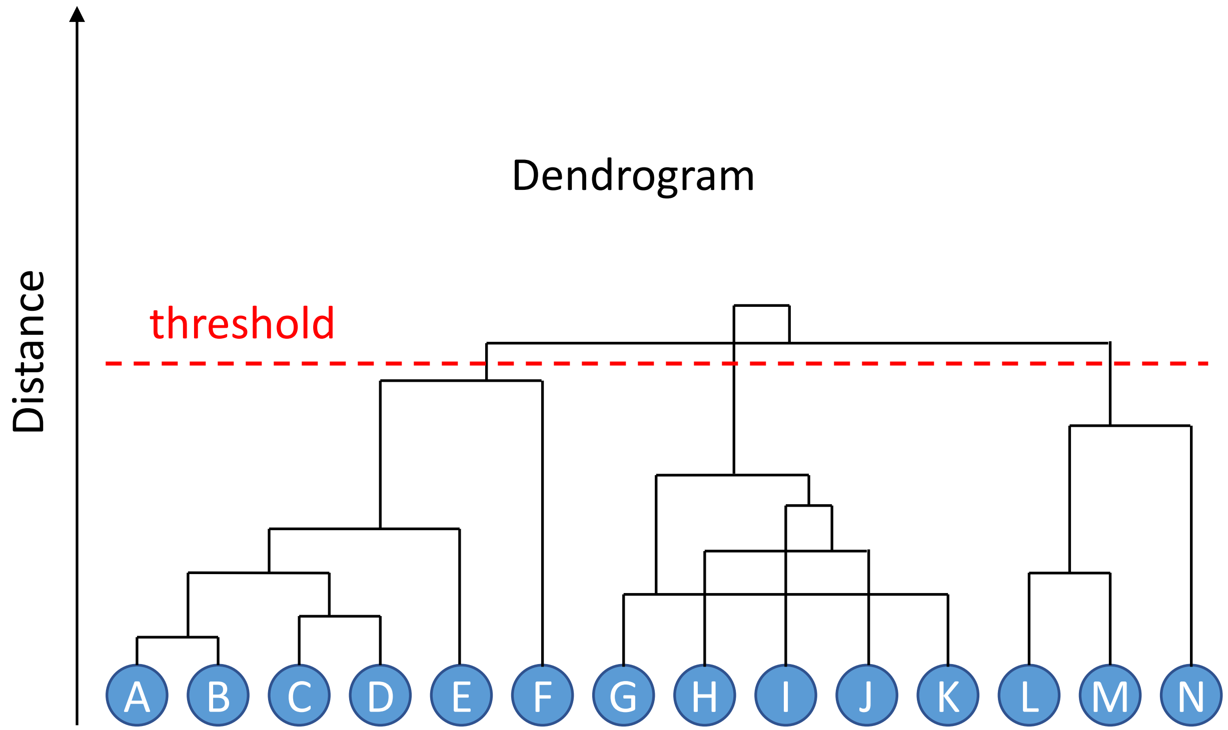
Hierarchical clustering example

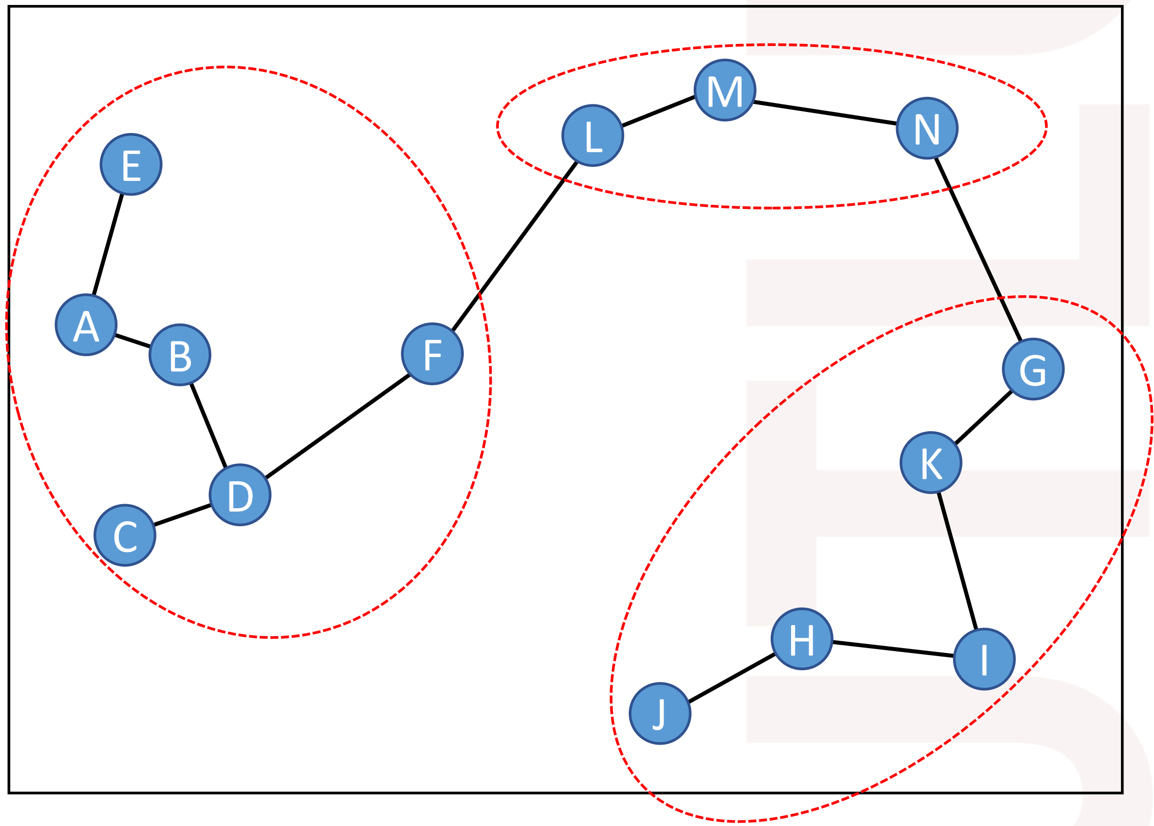




Hierarchical clustering example

In this case, distance between clusters is defined by the **closest** pair

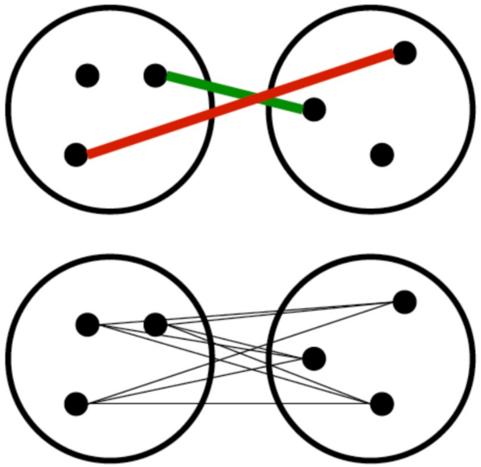




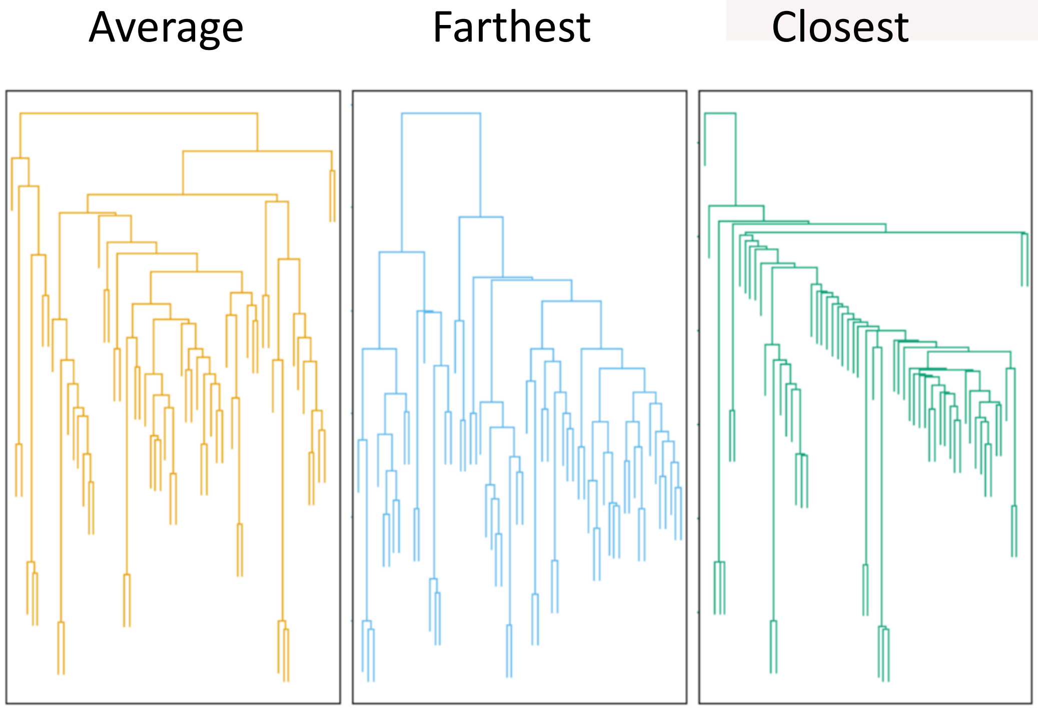
Hierarchical clustering

Distance options:

- ***single linkage (closest pair):***
  the minimum distance between samples in sub-clusters
- ***complete linkage (farthest pair) :***
  maximum distance between samples in sub-clusters
- ***average linkage (average of all pairs) :***
  average distance between each pair of samples in sub-clusters
- There are also other grouping strategies (such as centroid linkage)



Distance option influences the clustering result



Hierarchical clustering

Exercise

Given the following table that shows the distance between samples ("city block distance"), using agglomerative clustering method with ***single linkage***, draw the final dendrogram obtained.

city block distance :

A (3,5) B(2,7) -> D(A,B) = |3-2|+|5-7|=3

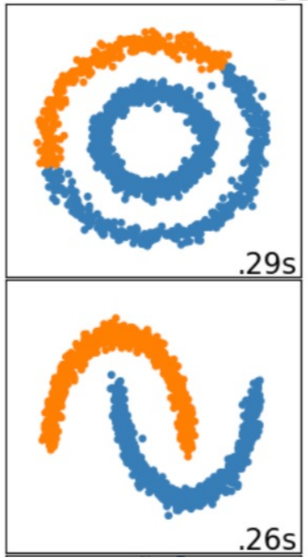
	A	B	C	D	E
A	0				
B	8	0			
C	3	6	0		
D	5	5	8	0	
E	13	10	2	7	0

Hierarchical clustering: pros and cons

- Pros
  - Hierarchical structure is more informative than flat clusters (*K*-means)
  - Easier to decide the number of clusters

- Cons:
  - Slow to compute. Time complexity $O(n^3)$.
  - Sensitive to outliers, because it tries to connect all data points.

Hierarchical clustering with Ward's method



Density-based clustering

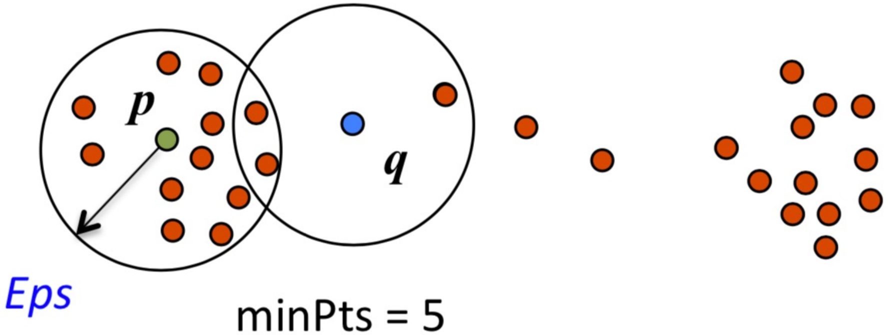
Idea: Clustering based on density (local clustering criterion), e.g., number of *densely* connected points.

Most well-known algorithm: DBSCAN (Density-based spatial clustering of applications with noise)

DBSCAN Ester, et al (1996)

- DBSCAN classifies all points as ***core*** *points*, (*density-*)***reachable*** *points* and ***outliers*** (or ***noise*** points):

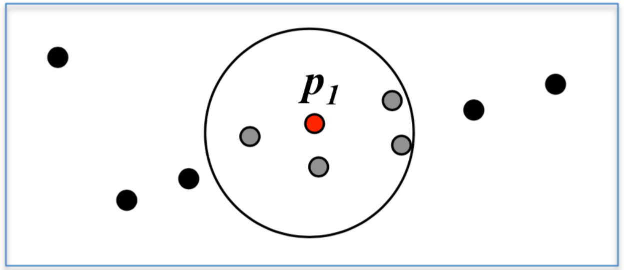
- A point ***p*** is a core point if at least `minPts` points are within distance ***ε*** (***ε*** is the maximum radius)
- A point ***q*** is directly reachable from ***p*** if point ***q*** is within distance ***ε*** from point ***p*** and ***p*** must be a core point.
- A point ***q*** is reachable from ***p*** if there is a path $\boldsymbol{p_1, \ldots, p_n}$ with $p_1 = p$ and $p_n = q$, where each $\boldsymbol{p_{i+1}}$ is directly reachable from $\boldsymbol{p_i}$.
- All points not reachable from any other point are outliers.



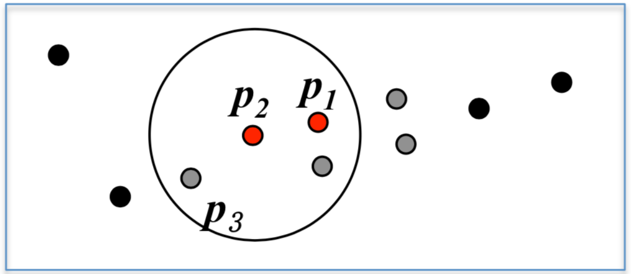
minPts = 5
P is a core point, q is not a core point

DBSCAN breakdown

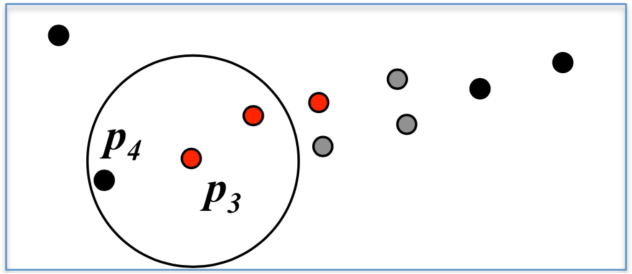
minPts = 4



Start from p_1
p_1 is a *core* point. Create a new **cluster C1**
There are 4 neighbor points and they all become candidates to expand

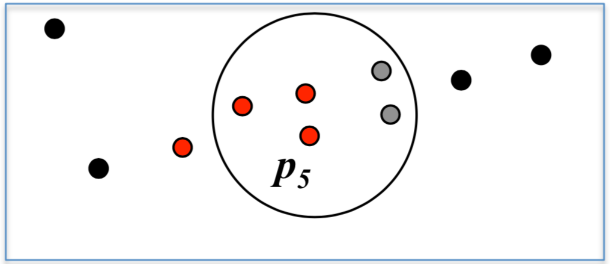


Add p_2 to C1
Found a new candidate p_3

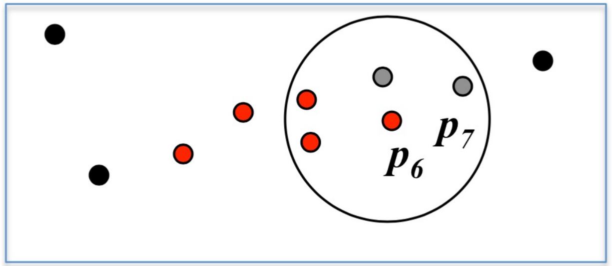


Add p_3 to C1
Found a new neighbor p_4, but it cannot be a candidate
Because p_3 is not a core point.

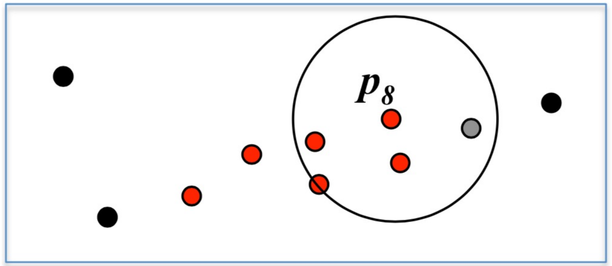
DBSCAN breakdown (cont.)



Add p_5 to C1
No new candidate is found

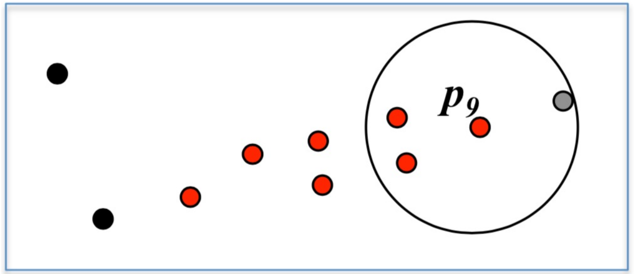


Add p_6 to C1
A new candidate p_7 is found

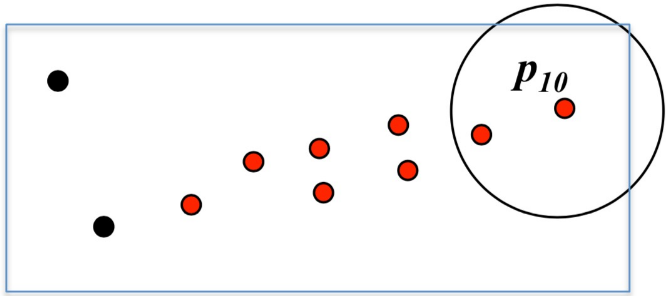


Add p_8 to C1
No new candidate is found

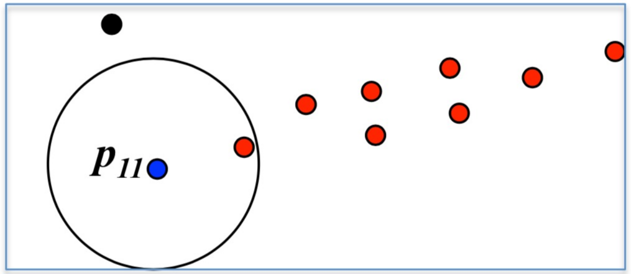
DBSCAN breakdown (cont.)



Add p_9 to C1
A new candidate p_{10} is found



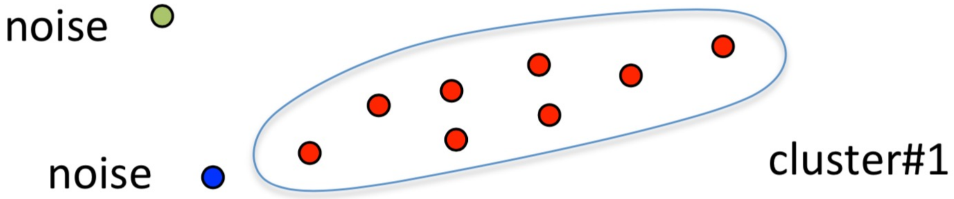
Add p_{10} to C1
p_{10} is not core point, stop expanding



Mark noise points

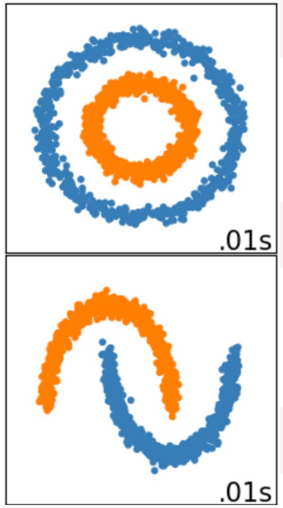
DBSCAN breakdown (cont.)

Final output



DBSCAN pros and cons

- Pros:
  - Pretty fast. Time complexity is $O(n \log n)$ when optimized.
  - Can find arbitrarily shaped clusters
  - Robust to outliers (recognized as noise points)



- Cons:
  - Cannot work well if density varies in different regions of data
  - Choosing a proper distance threshold ε can be difficult