

# Machine Learning (CE 40717)

Fall 2024

Ali Sharifi-Zarchi

CE Department  
Sharif University of Technology

November 16, 2025



## 1 Clustering Overview

## 2 Partitional Clustering

## 3 Challenges in K-means

## 4 Hierarchical Clustering

## 5 Other Clustering Algorithms

## 1 Clustering Overview

## 2 Partitional Clustering

### 3 Challenges in K-means

## 4 Hierarchical Clustering

## 5 Other Clustering Algorithms

# Supervised vs Unsupervised Learning

- **Supervised learning:** Given  $(\mathbf{x}_i, y_i)$ ,  $i = 1, \dots, n$ , learn a function

$$f:X \rightarrow Y.$$

- Categorical  $Y$ : classification
  - Continuous  $Y$ : regression
  - **Unsupervised learning:** Given only  $(\mathbf{x}_i), i = 1, \dots, n$ , can we infer the underlying structure of  $X$ ?
    - Involves analyzing unlabeled data to uncover hidden patterns or structures within the data.

# Supervised vs Unsupervised Learning

- **Supervised Learning**
    - *Classification*: partition examples into groups according to pre-defined categories.
    - *Regression*: assign values to feature vectors.
    - Requires labeled data for training.
  - **Unsupervised Learning**
    - *Clustering*: partition examples into groups when no pre-defined categories/classes are available.
    - *Novelty detection*: find changes in data.
    - *Outlier detection*: find unusual events (e.g., hackers).
    - Requires only instances, with no labels.

## Some Common Tasks

- **Clustering:** grouping data points into clusters based on similarity.
  - **Dimensionality Reduction:** reducing the number of features under consideration and keeping (perhaps approximately) the most informative ones.
  - **Anomaly Detection:** identifying data points that deviate significantly from the norm (e.g., fraud detection).
  - **Generative Modeling:** learning the distribution of data to generate new, similar instances.

## Clustering

- Clustering organizes data points into groups of similar objects.
  - Data points within a cluster are more similar to each other than to those in other clusters.
  - The notion of similarity depends on the task at hand (e.g., purchase behavior in market segmentation).

# Clustering

- **Goal:** automatically segment data into groups of similar points.
- **Question:** when and why would we want to do this?
- **Useful for:**
  - Automatically organizing data.
  - Understanding hidden structure in data.
  - Representing high-dimensional data in a low-dimensional space.
- **Some Applications of Clustering:**
  - Customer segmentation (marketing).
  - Image segmentation and object detection (computer vision).
  - Anomaly detection (cybersecurity, finance).
  - Genomics and bioinformatics.
  - Social network analysis and community detection.

# Clustering Set-up

- Our data are

$$D = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}.$$

- Each data point is  $p$ -dimensional, i.e.,

$$\mathbf{x}_n = \langle x_{n,1}, \dots, x_{n,p} \rangle.$$

- Define a *distance function* between data,

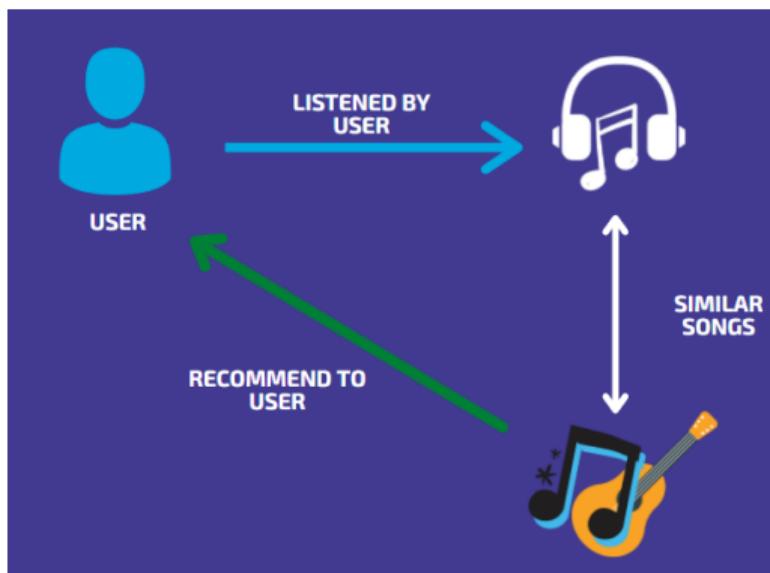
$$d(\mathbf{x}_n, \mathbf{x}_m).$$

- Goal: segment the data into  $K$  groups:

$$\{z_1, \dots, z_N\} \quad \text{where } z_i \in \{1, \dots, K\}.$$

# Clustering in Action: Music Recommendation Systems

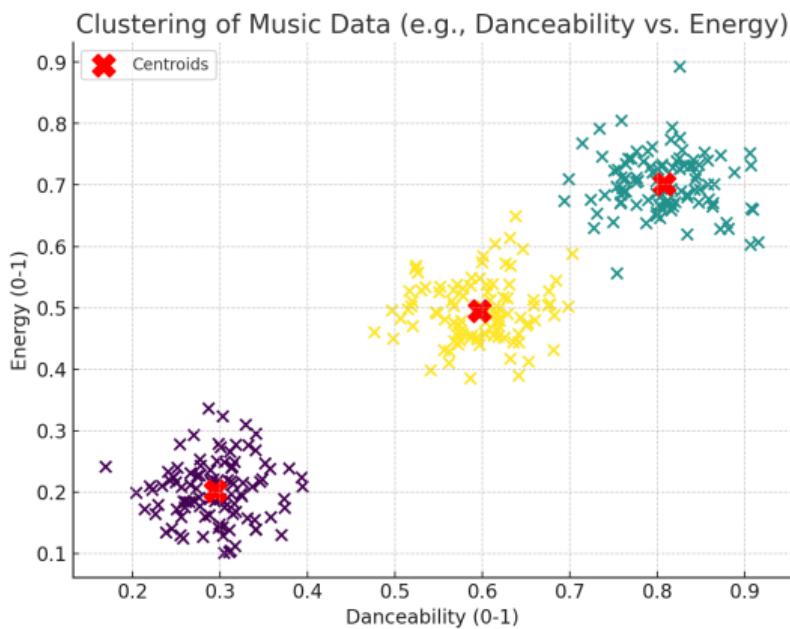
- Music recommendation systems cluster songs based on similarity.



Adopted from machinelearninggeek.com

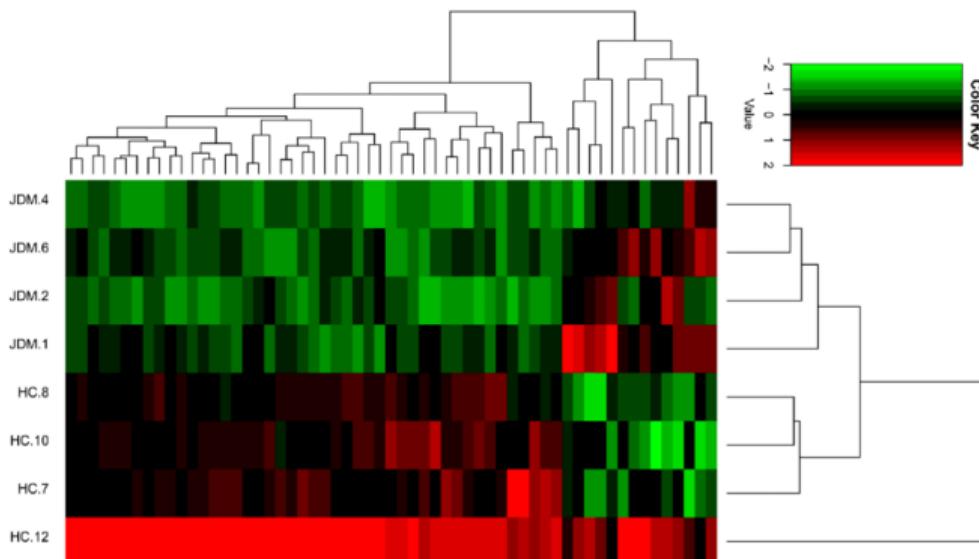
# Clustering in Action: Music Recommendation Systems

- When you like a song, the system suggests others from the same cluster.



# Clustering in Action: Gene Expression Clustering

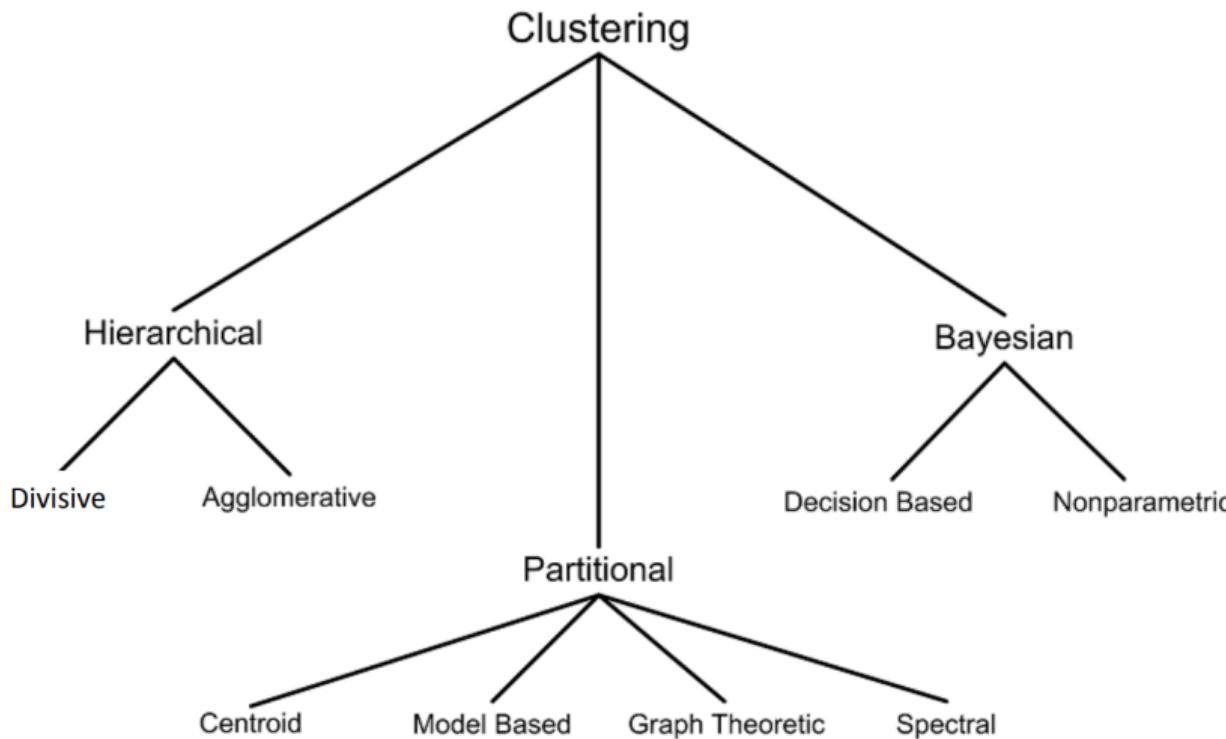
- Clustering can decipher hidden patterns in gene expression data, which can help in understanding disease mechanisms or genetic variations.



# Two Beginning Questions

- How do we create good clusters?
- How many clusters do we need?

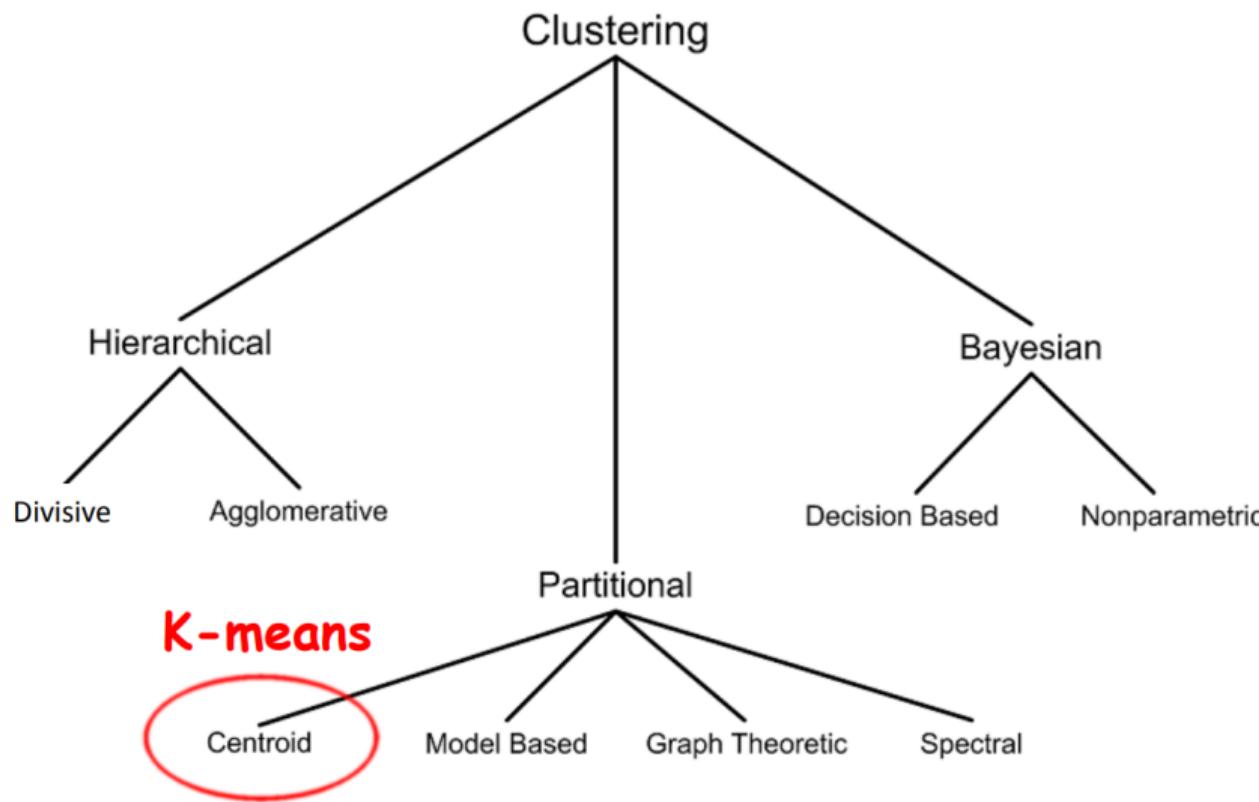
# Clustering Techniques



# Clustering Techniques

- **Partitional** algorithms typically determine all clusters at once, but can also be used as divisive algorithms in hierarchical clustering.
- **Hierarchical** algorithms find successive clusters using previously established clusters. These algorithms can be either **agglomerative** (bottomup) or **divisive** (topdown):
  - **Agglomerative algorithms** begin with each element as a separate cluster and merge them into successively larger clusters.
  - **Divisive algorithms** begin with the whole set and proceed to divide it into successively smaller clusters.
- **Bayesian** algorithms try to generate an *a posteriori distribution* over the collection of all partitions of the data.

# Clustering Techniques



## 1 Clustering Overview

## 2 Partitional Clustering

## 3 Challenges in K-means

## 4 Hierarchical Clustering

## 5 Other Clustering Algorithms

# Partitioning Algorithms: Basic Concept

- Construct a partition of a set of  $N$  objects into a set of  $K$  clusters:
  - The number of clusters  $K$  is given in advance.
  - Each object belongs to **exactly one** cluster in hard clustering methods.
- K-means is the most popular partitioning algorithm.

# Objective Based Clustering

- **Input:** a set of  $N$  points, and a distance/dissimilarity measure.
- **Output:** a partition of the data.
- **K-median:** find center points  $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_K$  to minimize

$$\sum_{i=1}^N \min_{j \in \{1, \dots, K\}} d(\mathbf{x}^{(i)}, \mathbf{c}_j)$$

- **K-means:** find center points  $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_K$  to minimize

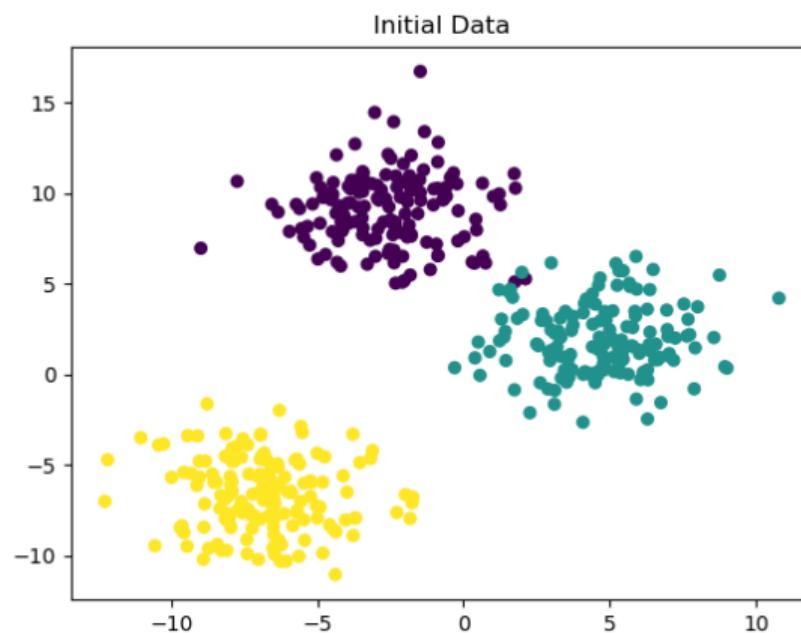
$$\sum_{i=1}^N \min_{j \in \{1, \dots, K\}} d^2(\mathbf{x}^{(i)}, \mathbf{c}_j)$$

- **K-center:** find a partition that minimizes the maximum radius.

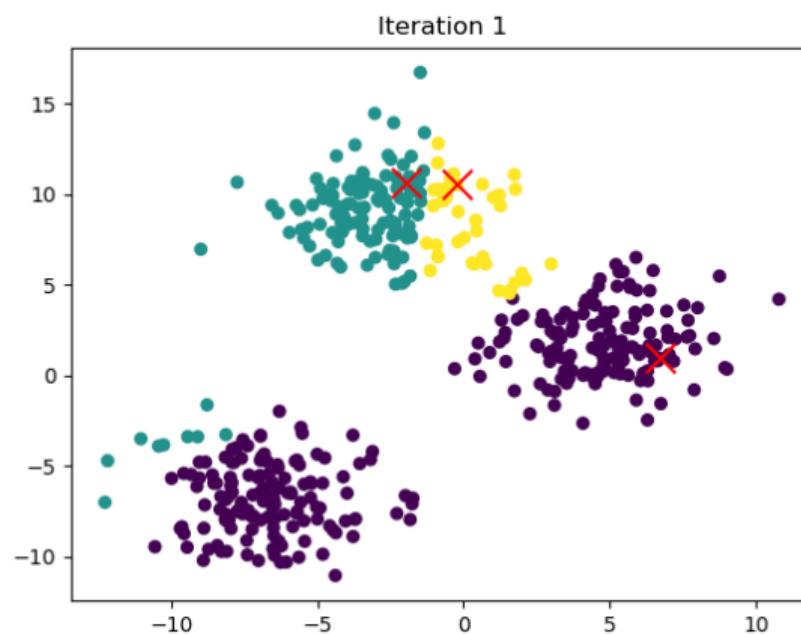
## K-Means Overview

- The most widely used clustering algorithm.
- Partitions data into  $K$  distinct groups based on feature similarity.
- It works by **iteratively** assigning data points to the nearest centroid (mean of the group) and then recalculating the centroids based on the new group memberships.
- The K-means algorithm is a **heuristic**.
- It requires initial centroids, and the choice is important.
- The process repeats until the assignments no longer change.

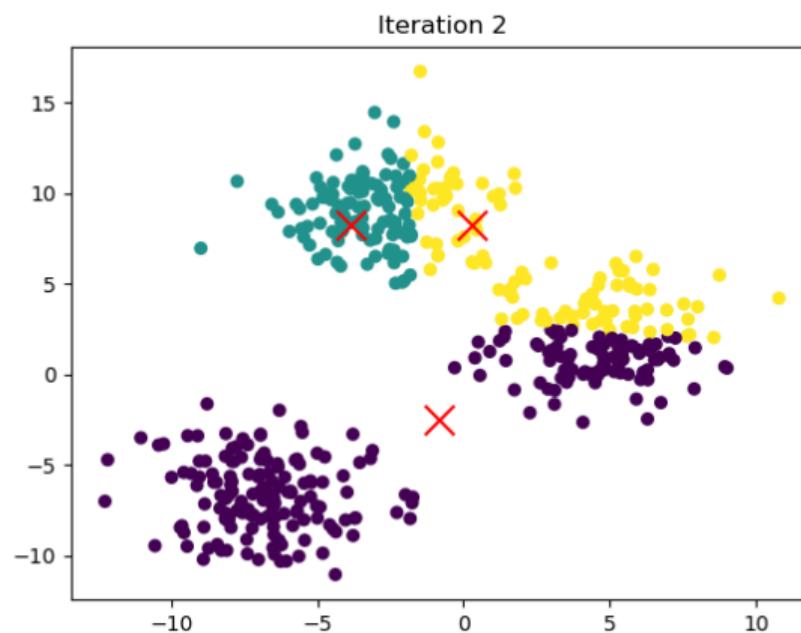
# K-Means in Action



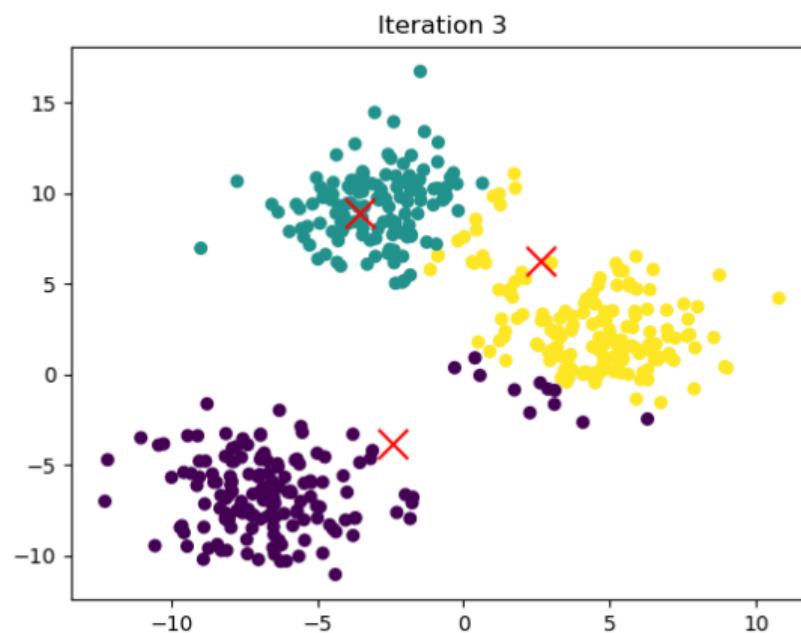
# K-Means in Action



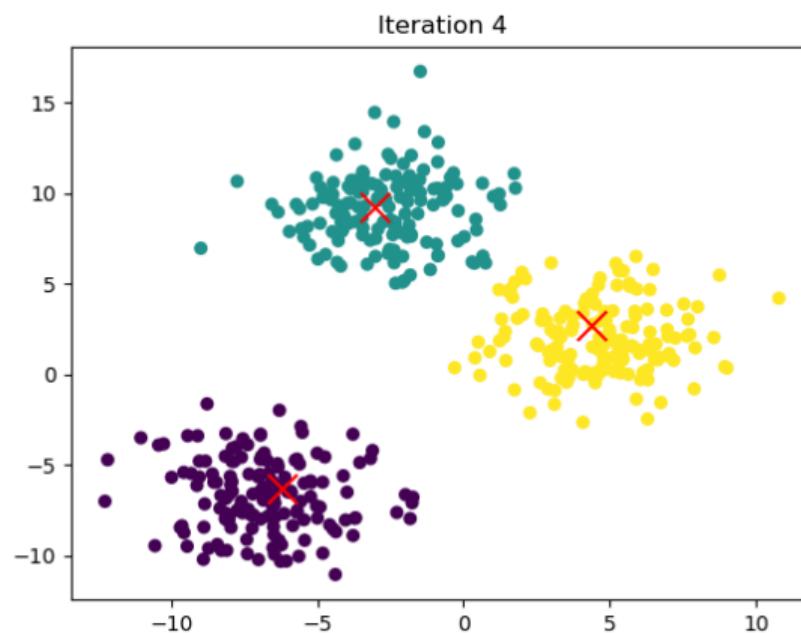
# K-Means in Action



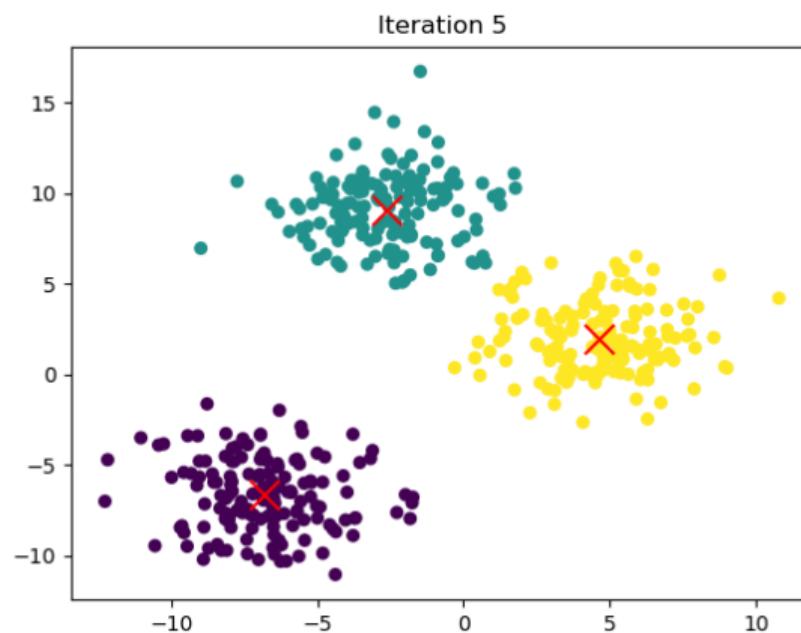
## K-Means in Action



## K-Means in Action



# K-Means in Action



# Problem Definition

- Formally: we have

$$X_{\text{train}} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}\} \subseteq \mathbb{R}^d.$$

- K is the number of clusters.
- We are learning:

- A function or mapping

$$f: \mathbb{R}^d \rightarrow \{1, 2, \dots, K\}$$

that assigns a cluster to each data point.

- A set of K prototypes

$$\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K\} \subseteq \mathbb{R}^d$$

as the cluster representatives, called **centroids**.

# Algorithm

---

## Algorithm 1 K-means Clustering

---

- 1: **Input:**  $K$  (number of clusters),  $D = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$  (data points)
  - 2: **Initialize:** Select  $K$  random points as centroids  $\{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K\}$
  - 3: **repeat**
  - 4:     Assign each point  $\mathbf{x}^{(i)}$  to nearest centroid  $f(\mathbf{x}^{(i)}) = \arg \min_j \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_j\|$
  - 5:     For each  $1 \leq j \leq K$  set  $C_j = \{x^{(i)} | f(x^{(i)}) = j\}$
  - 6:     Update centroids  $\boldsymbol{\mu}_j = \frac{1}{|C_j|} \sum_{x^{(i)} \in C_j} \mathbf{x}^{(i)}$
  - 7: **until** Centroids do not change
  - 8: **Output:** Final clusters  $\{C_1, C_2, \dots, C_K\}$
-

# Objective Function

- We want samples in the same cluster to be similar.
- In K-means, this is expressed as:

$$J = \sum_{j=1}^K \sum_{\mathbf{x}^{(i)} \in C_j} \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_j\|^2$$

- Choose  $f$  and

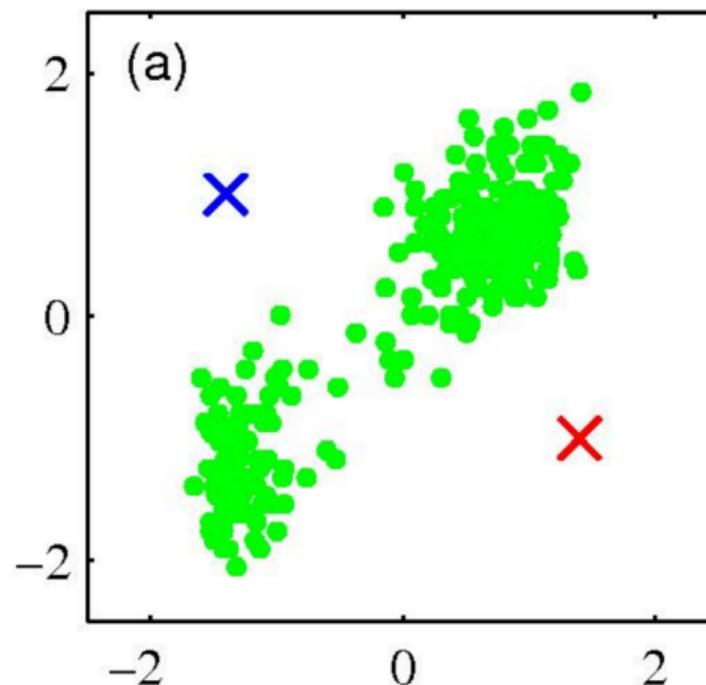
$$\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K\}$$

to minimize this.

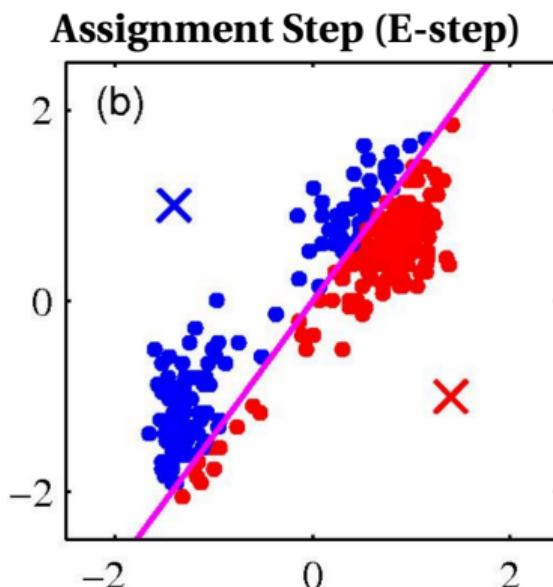
- This problem is NP-hard. K-means is a heuristic solution, which is **not** guaranteed to find an optimal solution.

## K-Means Process Example

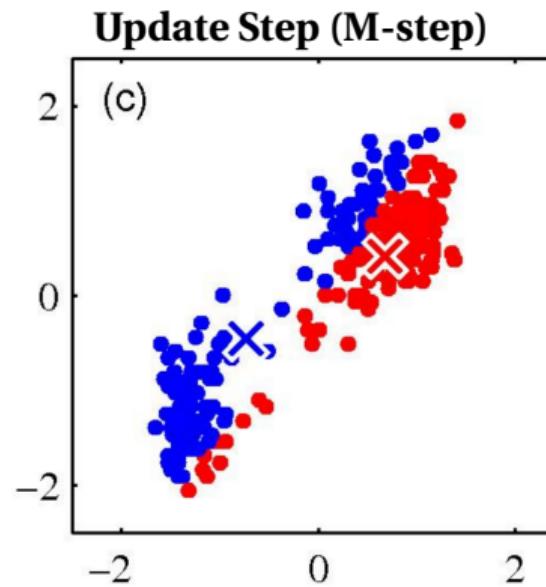
- Pick K random points as cluster centers (means).
- Shown here for  $K = 2$ .



## K-Means Process Example



**Iterative Step 1:** assign data points to the closest cluster center.

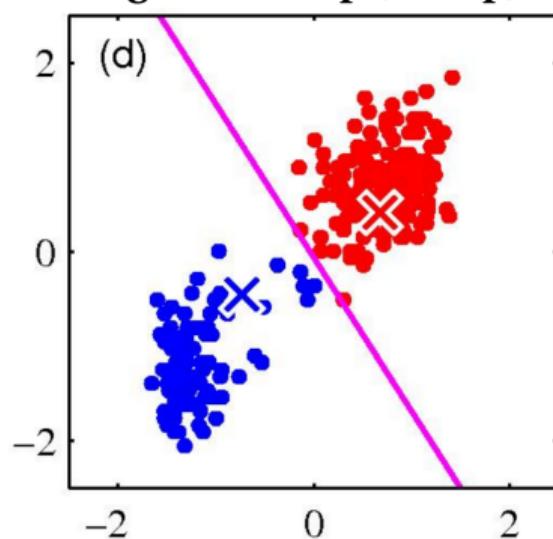


**Iterative Step 2:** update the cluster center to the average of the assigned points.

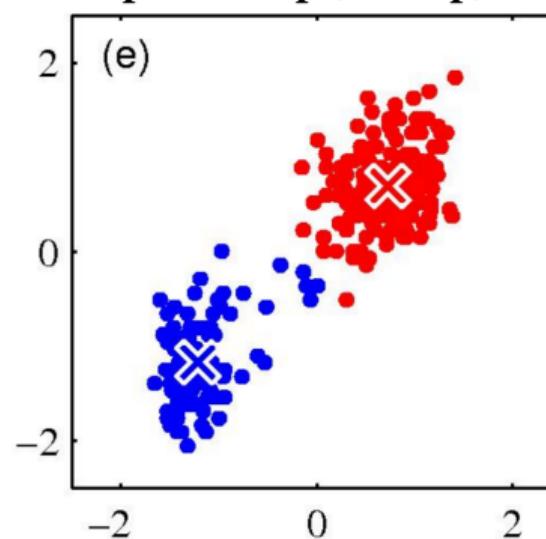
## K-Means Process Example

- Repeat until convergence.

**Assignment Step (E-step)**



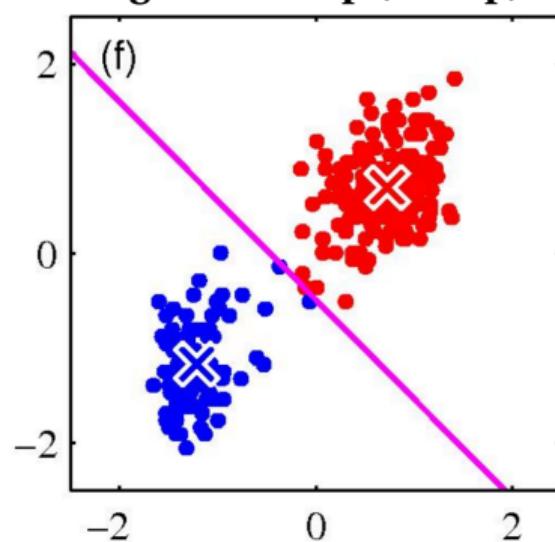
**Update Step (M-step)**



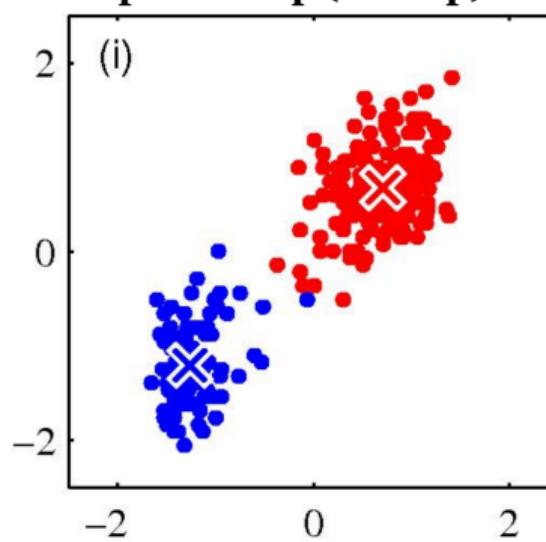
# K-Means Process Example

- Repeat until convergence.

**Assignment Step (E-step)**



**Update Step (M-step)**



# Convergence

- How do we know K-means will converge in a finite number of steps?
- First, we show that in each step  $J$  will decrease as long as we have not converged.

# Convergence

- We initially assign each sample to the nearest centroid:

$$f(\mathbf{x}) := \arg \min_j \|\mathbf{x} - \boldsymbol{\mu}_j\|^2.$$

- Keep each sample's assignment fixed until a closer centroid is found.
- Each time a sample is reassigned, the total distance between samples and their centroids decreases.
- The number of possible sample-to-centroid assignments is finite.
- The algorithm terminates when no sample changes its assigned centroid.

# Convergence

- In the updating step, with  $f(\mathbf{x})$  fixed,  $J$  is a quadratic function of  $\boldsymbol{\mu}_j$  (like SSE), and by taking the derivative we can minimize it as:

$$\frac{\partial J}{\partial \boldsymbol{\mu}_j} = 0 \implies \sum_{\mathbf{x}^{(i)} \in C_j} 2(\mathbf{x}^{(i)} - \boldsymbol{\mu}_j) = 0.$$

- This means we should **update** each  $\boldsymbol{\mu}_j$  as the mean of cluster  $C_j$ :

$$\boldsymbol{\mu}_j = \frac{\sum_{\mathbf{x}^{(i)} \in C_j} \mathbf{x}^{(i)}}{|C_j|}.$$

# Convergence

- For each cluster, the mean of its samples minimizes squared distances.
- For  $C_j$ , if  $\mu'_j$  was the old centroid, we have:

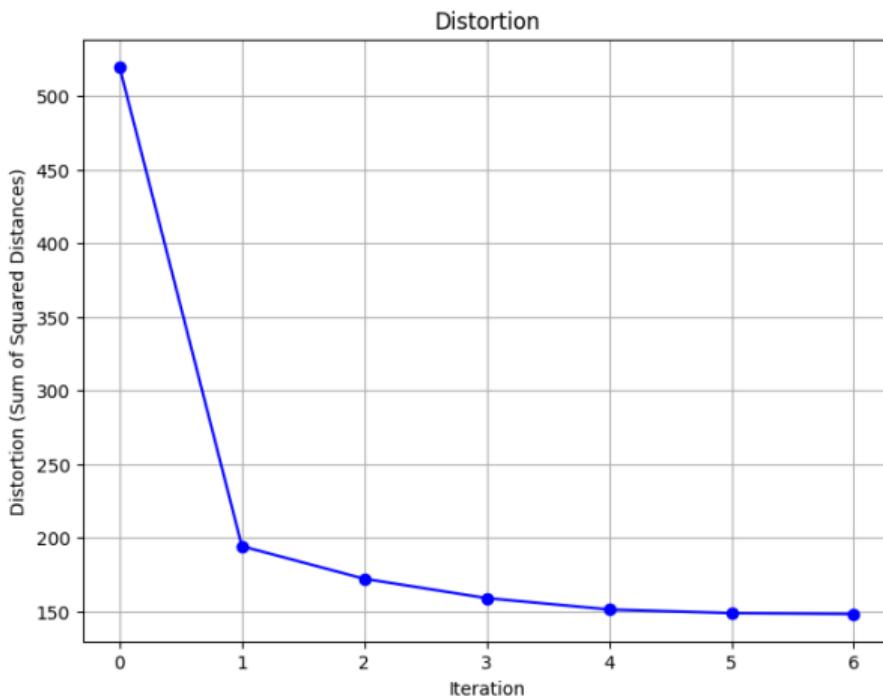
$$\sum_{\mathbf{x}^{(i)} \in C_j} \|\mathbf{x}^{(i)} - \mu'_j\|^2 \geq \sum_{\mathbf{x}^{(i)} \in C_j} \|\mathbf{x}^{(i)} - \mu_j\|^2.$$

So  $J_{\text{new}} \leq J_{\text{old}}$ .

# Convergence

- $J$  is non-negative, and there are a finite number of partitions, so there is a minimum for  $J$  and we cannot decrease  $J$  forever.
- Therefore, we must converge at some point.
- The convergence properties of the K-means algorithm were studied by MacQueen (1967).

## Convergence



# K-means Application: Compressing Images



- Each pixel is associated with a red, green, and blue value.
- A  $1024 \times 1024$  image is a collection of 1048576 values  $\langle x_1, x_2, x_3 \rangle$ , represented as a vector  $\mathbf{x}$ , which requires 3M of storage.
- How can we use K-means to compress this image?

# Vector Quantization



- Replace each pixel  $\mathbf{x}_n$  with its assignment  $\mathbf{m}_{z_n}$  (paint by numbers).
- The K means are called the *codebook*.
- With  $K = 100$ , we need 7 bits per pixel plus  $100 \times 3$  bytes  $\approx 897K$ .

# Vector Quantization



2 means



4 means



8 means



16 means



32 means



64 means

## 1 Clustering Overview

## 2 Partitional Clustering

## 3 Challenges in K-means

## 4 Hierarchical Clustering

## 5 Other Clustering Algorithms

# Strengths

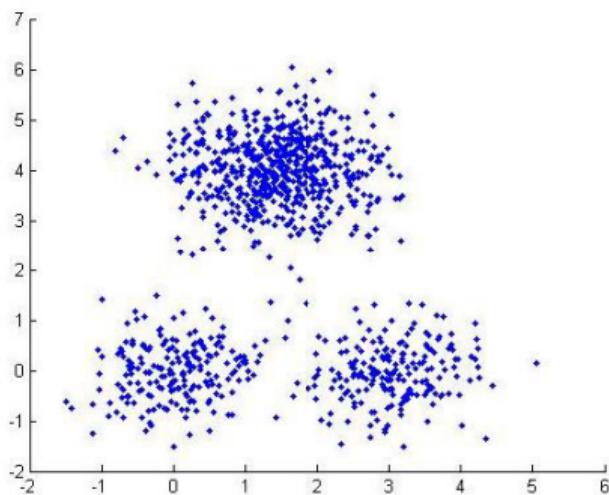
- **Simple:** easy to understand and implement.
- **Efficient:** time complexity  $O(tKn)$ , where
  - $n$  is the number of data points,
  - $K$  is the number of clusters, and
  - $t$  is the number of iterations (it requires initial centroids, and the choice is important as it could affect  $t$  in  $O(tkn)$ ).
- K-means is the most popular clustering algorithm.
- Note that it terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity.

# Local Optimum

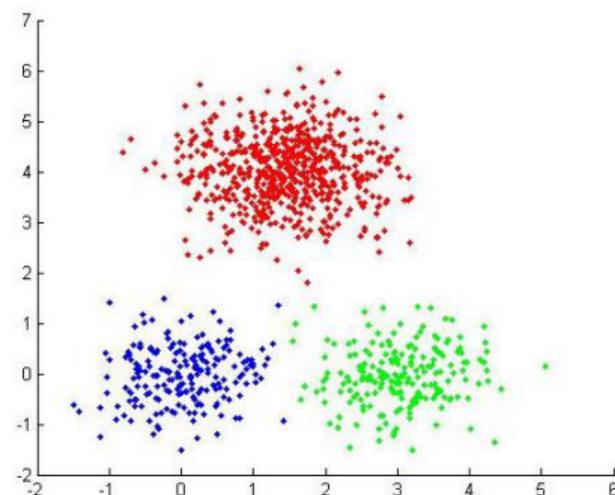
- The algorithm finds a local minimum, but there is no guarantee of finding the global minimum.
- Its result is highly affected by the initialization.
- Some suggestions are:
  - Multiple runs with random initial centroids, then select the best result.
  - Initialization heuristics (K-means++, Furthest Traversal).
  - Initializing with the suggested results of another method.



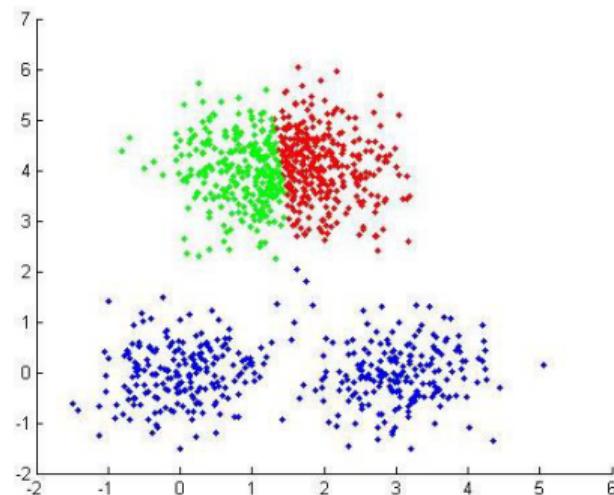
# Local Optimum



# Local Optimum



Optimal clustering



Possible clustering

## Weaknesses of K-means

- The algorithm is only applicable if the **mean** is defined.
  - For categorical data, K-modes can be used — the centroid is represented by the most frequent values.
- The user needs to specify K.
- The algorithm is sensitive to **outliers**.
  - Outliers are data points that are very far away from other data points.
  - Outliers could be errors in data recording or some special data points with very different values.

## Definition of Mean

- We assume  $\mathbf{x}^{(i)} \in \mathbb{R}^d$ , which is not always the case. K-means requires a space where the sample **mean** is defined.
  - Categorical data.
  - A suggested solution: K-modes — the centroid is the most frequent category (the mode) in each cluster.
  - The closest centroid is found using the Hamming distance.

# How many clusters?



How many clusters?



Six Clusters



Two Clusters



Four Clusters

Adopted from slides of Dr. Soleymani, Modern Information Retrieval Course, Sharif University of technology.

# How Many Clusters?

- The number of clusters  $K$  is given in advance in the K-means algorithm.
  - However, finding the right number of clusters is itself part of the problem.
- There is a trade-off between having better focus within each cluster and having too many clusters.
- Hold-out validation / cross-validation on an auxiliary task (e.g., a supervised learning task).
- Optimization problem: penalize having many clusters.
  - Some criteria can be used to automatically estimate  $K$ .
  - Penalize the number of bits needed to describe the extra parameter.

$$J'(C) = J(C) + |C| \times \log N$$

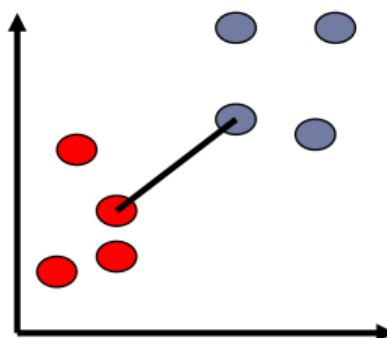
- First, we need to know how we can evaluate a clustering.

# Clustering Evaluation

- Evaluating clusters involves two key aspects:
  - Intra-cluster cohesion (compactness)**: how similar the data points are within a cluster.
    - Often measured by the within-cluster sum of squares (WCSS):
$$WCSS = \sum_{i=1}^K \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$
  - Inter-cluster separation (isolation)**: how different the data points are between clusters.

# Inter-cluster Separation

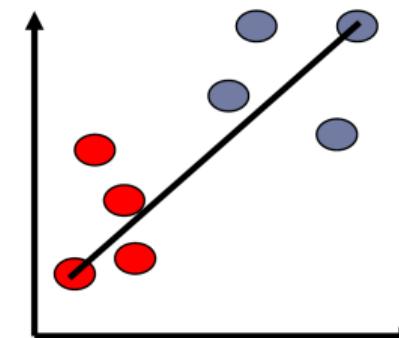
## Single-link (Minimum Distance)



Measures the minimum distance between any two points from different clusters.

$$d_{\text{single}}(C_i, C_j) = \min_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y})$$

## Complete-link (Maximum Distance)

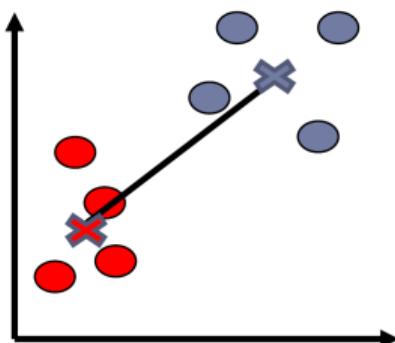


Measures the maximum distance between any two points from different clusters.

$$d_{\text{complete}}(C_i, C_j) = \max_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y})$$

# Inter-cluster Separation

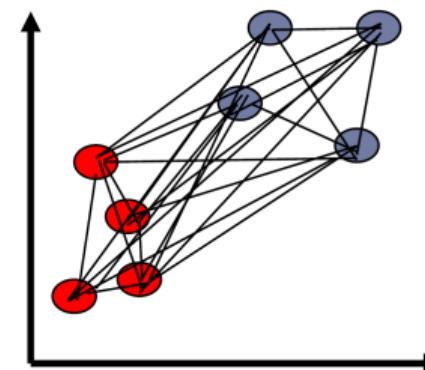
## Centroid (Wards Method)



Measures the distance between the centroids of two clusters.

$$d_{\text{centroid}}(C_i, C_j) = d(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)$$

## Average-link

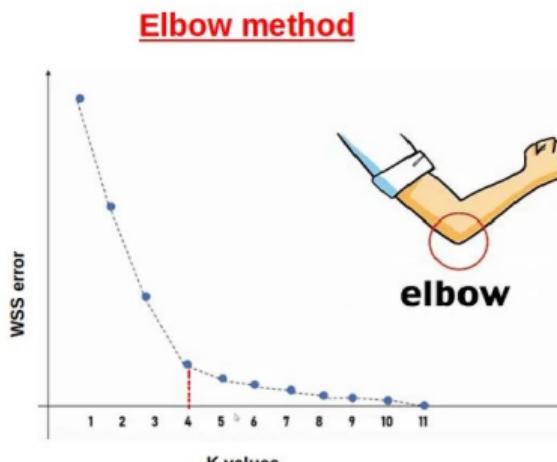


Measures the average distance between all pairs of points from different clusters.

$$d_{\text{average}}(C_i, C_j) = \frac{1}{|C_i| \cdot |C_j|} \sum_{\mathbf{x} \in C_i} \sum_{\mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y})$$

## Elbow Method for Optimal K

- Finds the optimal number of clusters K by minimizing the within-cluster sum of squares (WCSS).
- Elbow point:
  - Plot WCSS versus K.
  - The point where the rate of decrease sharply slows down (resembling an elbow) is considered the optimal K.



## Silhouette Method for Cluster Evaluation

- Silhouette score for a single point  $i$ :

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

- where:
  - $a(i)$  is the average distance between  $i$  and all other points in the same cluster.
  - $b(i)$  is the average distance between  $i$  and points in the nearest neighboring cluster.
- Interpretation:
  - $S(i) \in [-1, 1]$
  - $S(i) \approx 1$  : well-clustered.
  - $S(i) \approx 0$  : on or near the decision boundary between clusters.
  - $S(i) \approx -1$  : misclustered.

# How Many Clusters?

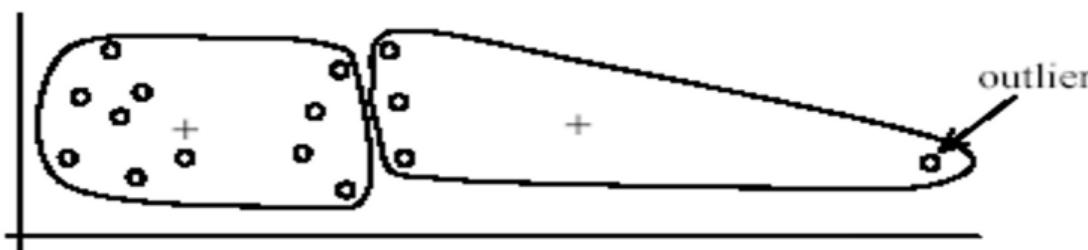
- There is a trade-off between having better focus within each cluster or having too many clusters.
- Don't want one-element clusters.
- **Optimization problem:** penalize having too many clusters

$$K^* = \arg \min_k (J(k) + \lambda k)$$

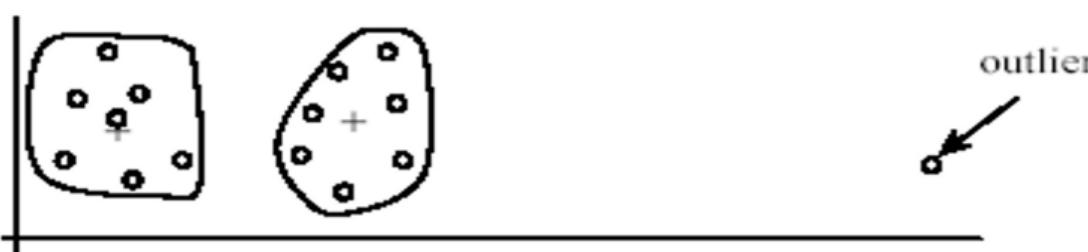
# Outliers

- The algorithm is sensitive to outliers.
- Outliers are data points that are very far away from other data points.
- Outliers could be errors in data recording or unique data points with significantly different values.

## Weaknesses of K-means: Problems with Outliers



Undesirable clusters

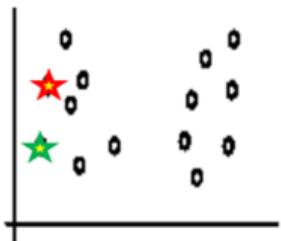


Ideal clusters

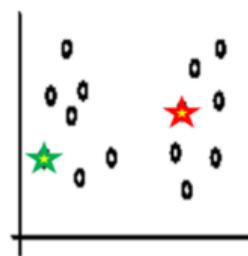
# Dealing with Outliers

- Remove data points that are much farther away from the centroids than other data points.
  - To be safe, we may want to monitor these possible outliers over a few iterations and then decide whether to remove them.
- Perform random sampling: by choosing a small subset of the data points, the chance of selecting an outlier is much smaller.
  - Assign the rest of the data points to the clusters by distance or similarity comparison, or by classification.

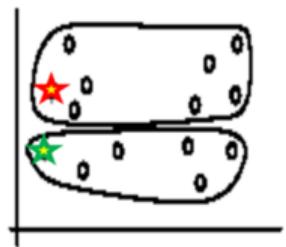
## Sensitivity to Initial Seeds



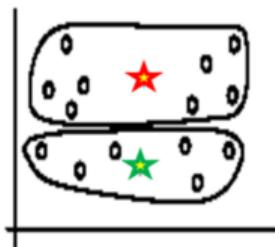
Random selection of seeds (centroids)



Random selection of seeds (centroids)



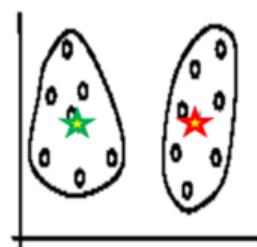
Iteration 1



Iteration 2



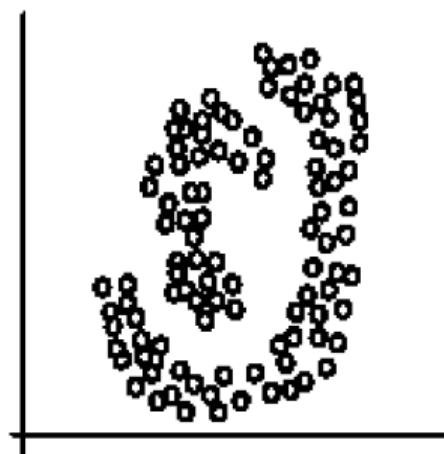
Iteration 1



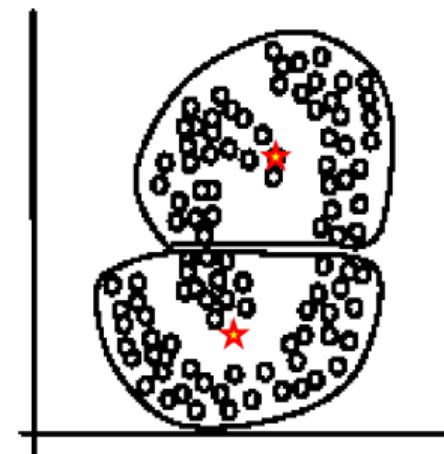
Iteration 2

# Special Data Structures

- The K-means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).



Two natural clusters



K-means clusters

## Data Distribution

- There is a problem with how K-means defines clusters.
- K-means assumes clusters are spherical and separated with equal variance, which limits its effectiveness on non-spherical or complex-shaped clusters.

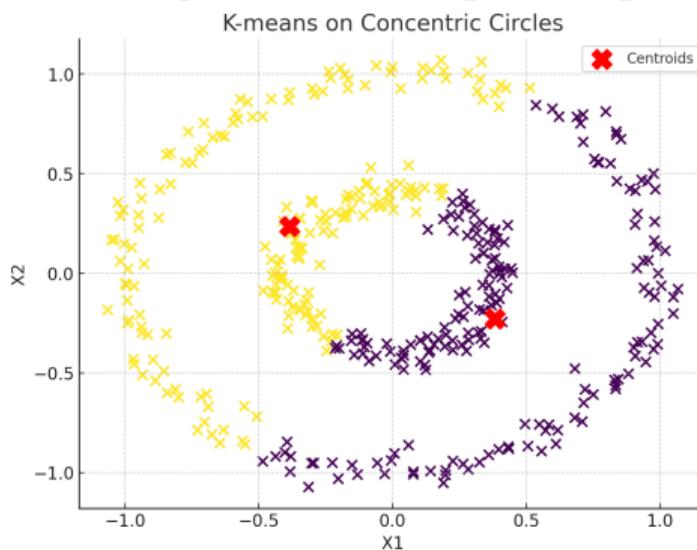
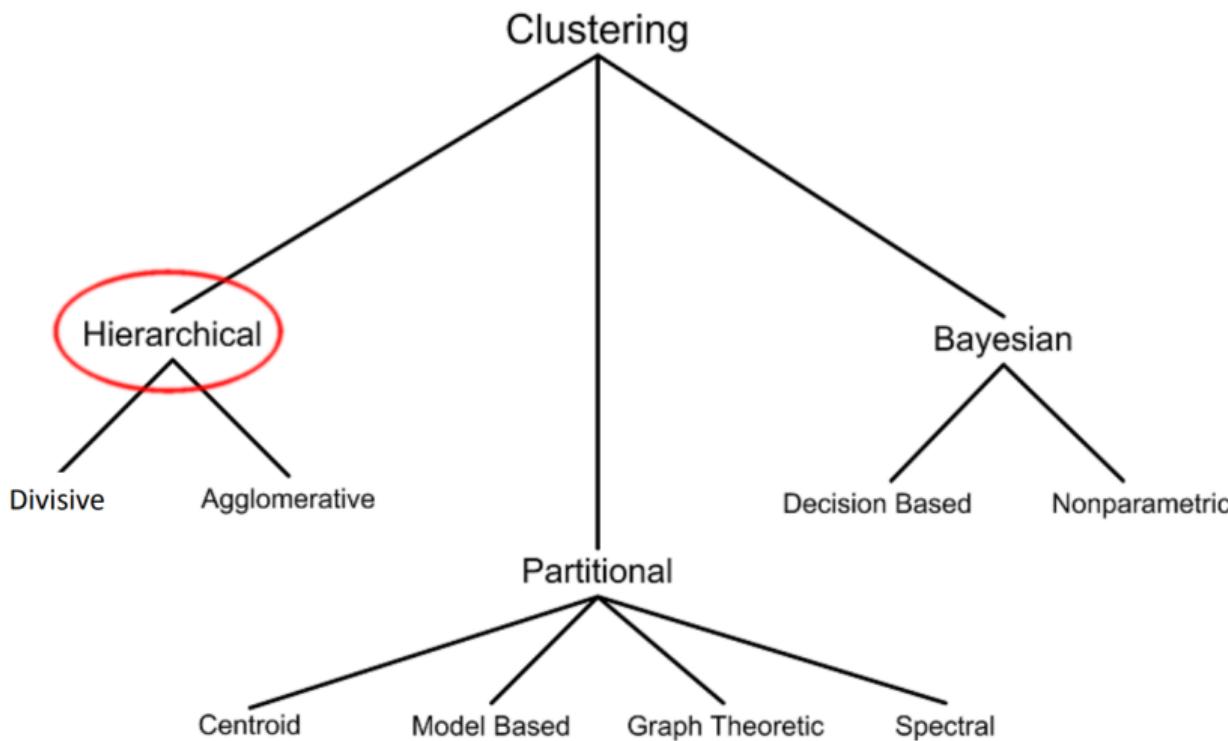


Figure 1: Example when K-means won't work

# Hierarchical Algorithm



## 1 Clustering Overview

## 2 Partitional Clustering

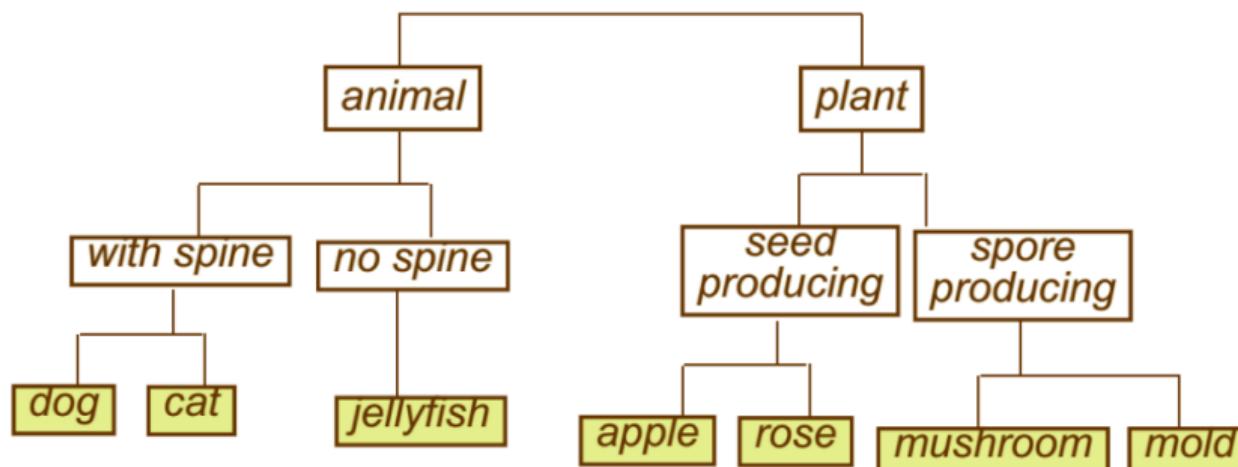
## 3 Challenges in K-means

## 4 Hierarchical Clustering

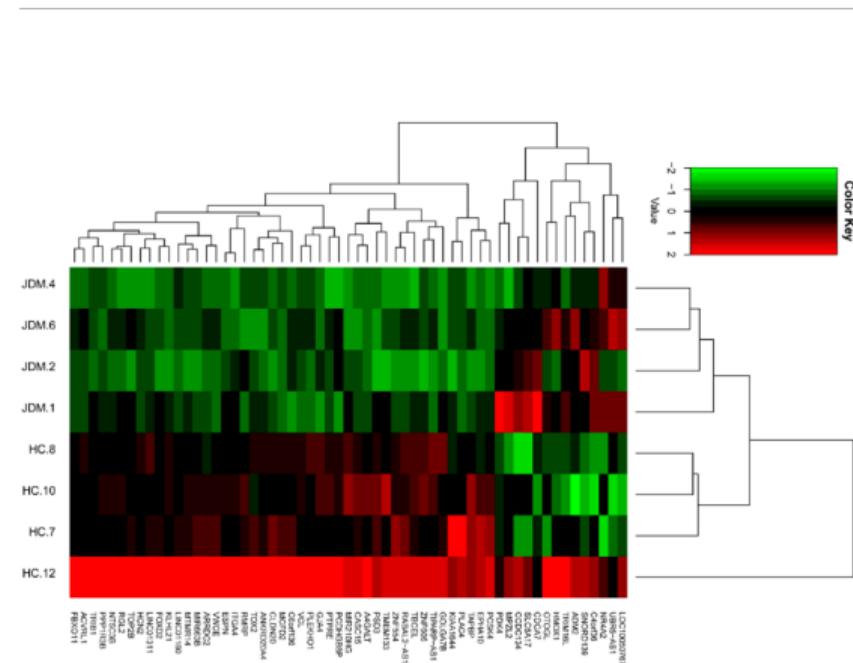
## 5 Other Clustering Algorithms

# Hierarchical Clustering

- **Hierarchical** algorithms find successive clusters using previously established clusters. Two types:
  - **Agglomerative (bottom-up)**: start with individual points and merge clusters.
  - **Divisive (top-down)**: start with all points and split clusters.
- **Result:** a hierarchy of clusters represented by a dendrogram.



## Dendrogram



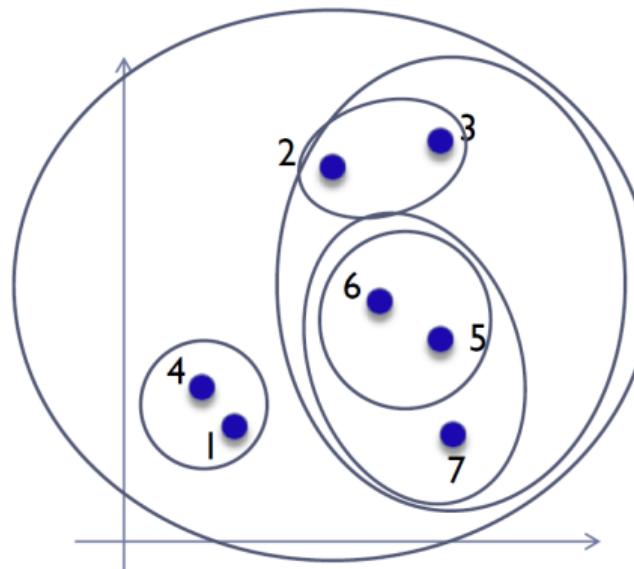
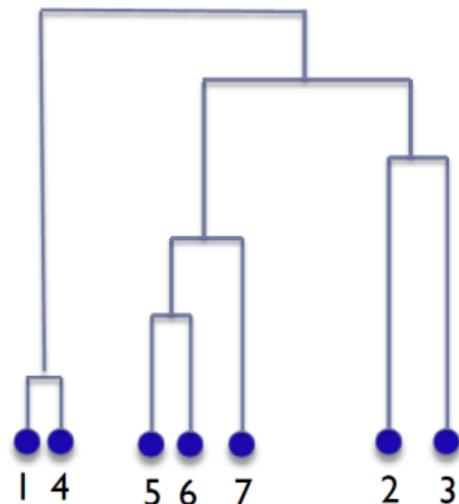
# Types of Hierarchical Clustering

- **Agglomerative (bottom-up) clustering:** It builds the dendrogram (tree) from the bottom level.
  - Merges the most similar (or nearest) pair of clusters.
  - Stops when all data points are merged into a single cluster (i.e., the root cluster).
- **Divisive (top-down) clustering:** It starts with all data points in one cluster, the root.
  - Splits the root into a set of child clusters; each child cluster is recursively divided further.
  - Stops when only singleton clusters of individual data points remain (i.e., each cluster has only a single point).

# Hierarchical Agglomerative Clustering (HAC)

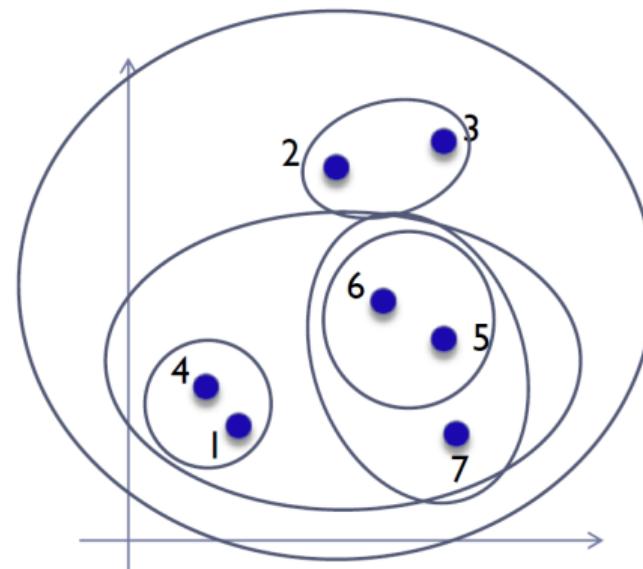
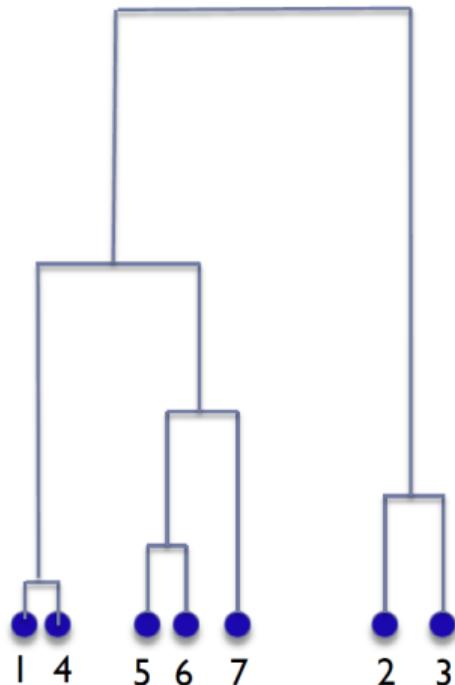
- Start with each point as its own cluster.
- Merge the closest clusters.
- Repeat until one cluster remains or the desired number is reached.
- The closest clusters can be determined using inter-cluster separation measures.
- Basic algorithm:
  - Start with all instances in their own cluster.
  - Until there is only one cluster:
    - Among the current clusters, determine the two clusters,  $c_i$  and  $c_j$ , that are closest.
    - Replace  $c_i$  and  $c_j$  with a single cluster  $c_i \cup c_j$ .

# Single-Link



Keep the maximum bridge length as small as possible.

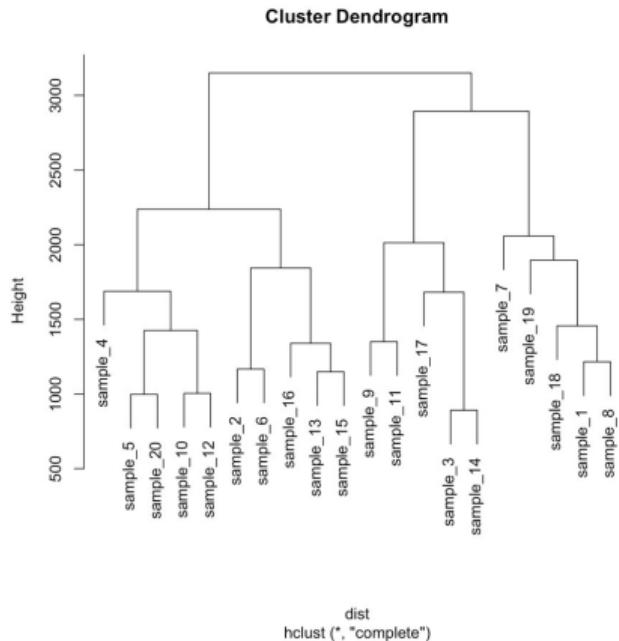
# Complete Link



Keep the maximum diameter as small as possible.

# Dendrogram and Cutting

- A dendrogram shows the hierarchy of merges.
- Cut the dendrogram at a desired level to form clusters.



# Hierarchical Algorithms

- Advantages:
  - No need to specify the number of clusters.
  - Produces a dendrogram for visualization.
  - Works with arbitrary-shaped clusters.
- Disadvantages:
  - High computational cost.
  - Sensitive to noise and outliers.
  - Greedy: cannot undo merges.

## 1 Clustering Overview

## 2 Partitional Clustering

## 3 Challenges in K-means

## 4 Hierarchical Clustering

## 5 Other Clustering Algorithms

# Hard vs Soft Clustering

- **Hard Clustering (Partitional):** Each data point belongs to exactly one cluster
  - More common and easier to use.
- **Soft Clustering (Bayesian):**

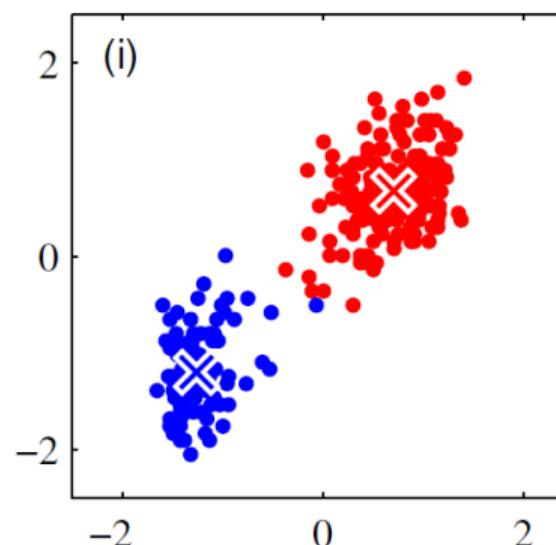


Figure adapted from Machine Learning and  
Pattern Recognition, Bishop

# Hard vs Soft Clustering

- **Hard Clustering (Partitional)**
- **Soft Clustering (Bayesian):** Each sample is assigned to different clusters with probabilities, rather than {0, 1}.
  - A data point belongs to each cluster with a probability.

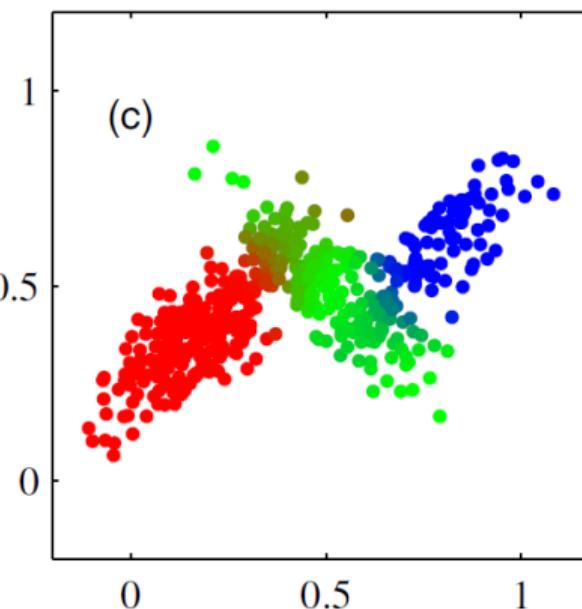


Figure adapted from Machine Learning and  
Pattern Recognition, Bishop

# DBSCAN

## DBSCAN (Density-Based Spatial Clustering of Applications with Noise):

- Groups points in high-density regions.
- Labels points in low-density regions as noise.
- Does not require specifying the number of clusters  $K$ .

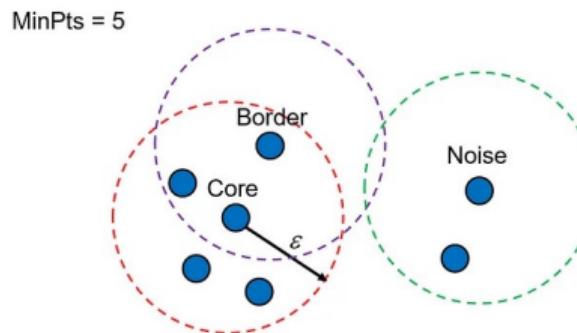
### Parameters:

- $\epsilon$  (epsilon): Maximum distance for neighbors.
- minPts: Minimum number of points to form a dense region.

# Core Concepts in DBSCAN

**DBSCAN defines three types of points:**

- **Core Point:** A point with at least  $\text{minPts}$  neighbors within distance  $\epsilon$ .
- **Border Point:** A point within  $\epsilon$  of a core point but with fewer than  $\text{minPts}$  neighbors.
- **Noise:** Points that are neither core points nor border points.



Adopted from [ai.plainenglish.io](http://ai.plainenglish.io)

# Core Concepts in DBSCAN

## Definitions:

- A point  $x_i$  is a core point if:

$$|\{x_j : d(x_i, x_j) \leq \epsilon\}| \geq \text{minPts}$$

- A point is a border point if it is within distance  $\epsilon$  of a core point but is not itself a core point.

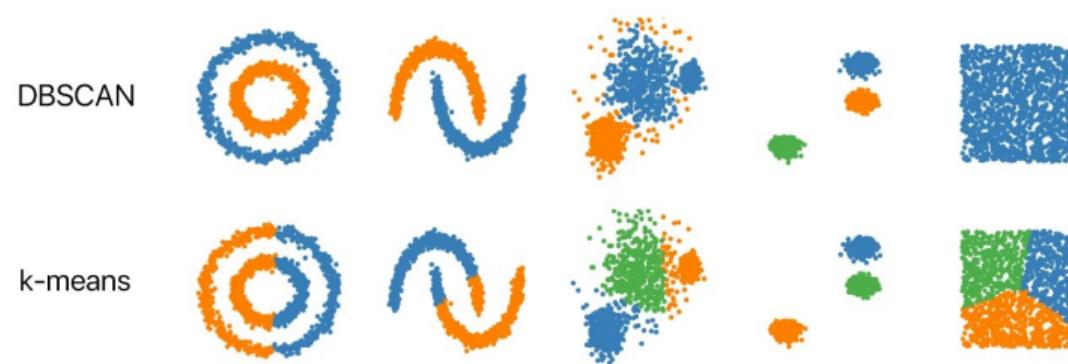
# DBSCAN Algorithm Steps

## Algorithm Steps:

- For each unvisited point  $x_i$ :
  - Mark  $x_i$  as visited.
  - Find all points within distance  $\epsilon$  (neighborhood).
- If  $x_i$  is a core point:
  - Create a new cluster and expand it by recursively adding all reachable core and border points.
- If  $x_i$  is not a core point:
  - Label it as noise if it does not belong to any cluster.

## Advantages of DBSCAN

- Can find clusters of arbitrary shape (non-spherical).
- Does not require specifying the number of clusters  $K$  in advance.
- Robust to noise and outliers.
- Works well with large datasets.



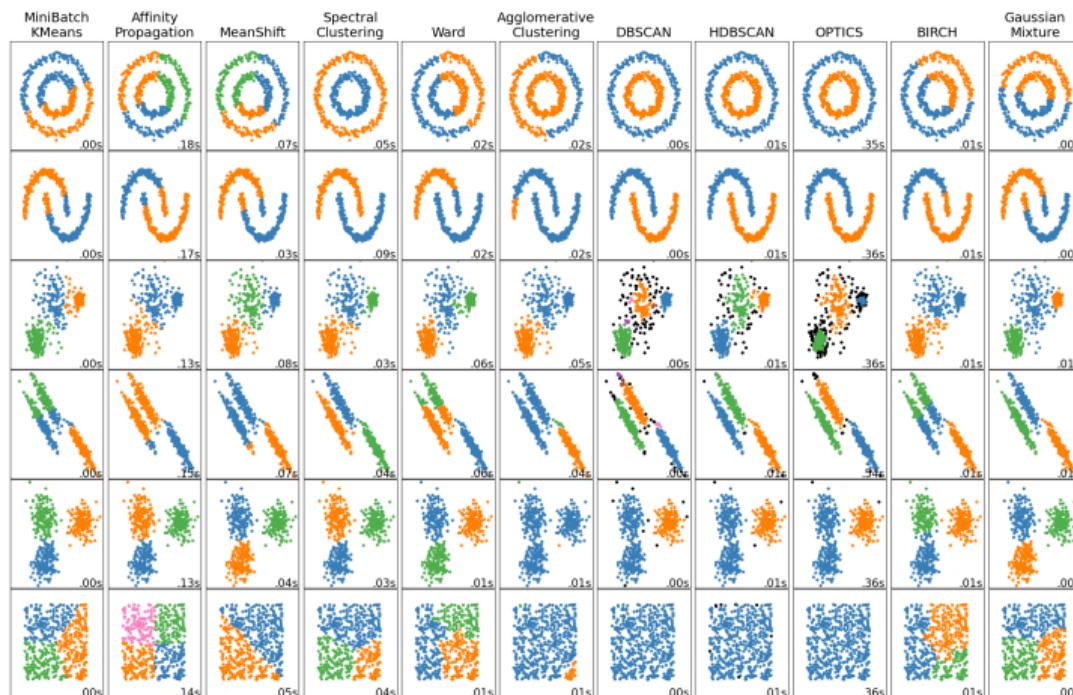
Adopted from [mrinalyadav7.medium.com](https://mrinalyadav7.medium.com)

## Limitations of DBSCAN

- DBSCAN struggles with datasets of varying densities.
- Sensitive to the selection of parameters  $\epsilon$  and `minPts`.
- Does not perform well with high-dimensional data.

# Clustering Algorithms

- Each algorithm is suited for different kinds of patterns and information in data.



# Contributions

- **This slide has been prepared thanks to:**
  - Hooman Zolfaghari
  - Mahdi Aghaei

- [1] M. Soleymani Baghshah, “Machine learning.” Lecture slides.
- [2] B. Póczos, “Advanced introduction to machine learning.” Lecture slides.  
CMU-10715.
- [3] M. Gormley, “Introduction to machine learning.” Lecture slides.  
10-701.
- [4] M. Gormley, “Introduction to machine learning.” Lecture slides.  
10-301/10-601.
- [5] F. Seyyedsalehi, “Machine learning and theory of machine learning.” Lecture slides.  
CE-477/CS-828.
- [6] G. Strang, “Linear algebra and its applications,” 2000.