Machine Learning

Clustering

Cigdem Beyan

A. Y. 2024/2025

Supervised vs. Unsupervised Learning

- Up to now we considered supervised learning scenarios, where we are given
  - samples $x_1, \ldots, x_n$
  - class labels for all samples $w_1, \ldots, w_n$ (or $c_1, \ldots, c_n$)
- In the next few lectures, we consider unsupervised learning and particularly clustering, where we have only
  - samples $x_1, \ldots, x_n$
  - But not class labels

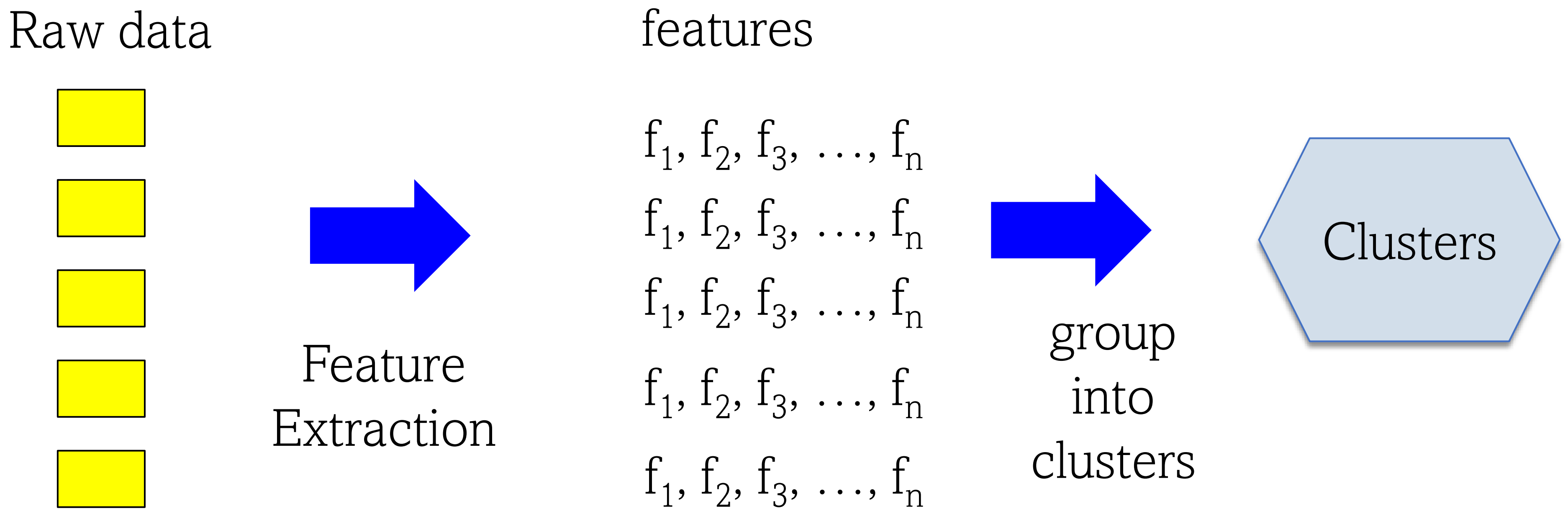
Unsupervised Learning

- **Parametric approach**
  - Assume parametric distribution of data
  - Estimate parameters of this distribution
  - Much “harder” than supervised parametric approaches

- **Non-Parametric Approach**
  - Group the data into clusters, each cluster (hopefully) involved something about categories (classes) present in the data

Clustering

Clustering is an area of unsupervised learning in which we find "clusters" or groups of similar points within the data without having a label.


Raw data
Feature Extraction
features
$f_1, f_2, f_3, \ldots, f_n$
$f_1, f_2, f_3, \ldots, f_n$
$f_1, f_2, f_3, \ldots, f_n$
$f_1, f_2, f_3, \ldots, f_n$
$f_1, f_2, f_3, \ldots, f_n$
group into clusters
Clusters

Why Clustering?

- **Pattern Recognition:** Clustering helps identify patterns and structures within data. It allows the algorithm to discover inherent groupings or clusters in the data, revealing relationships that might not be apparent through other means.
- In many real-world scenarios, obtaining labeled data can be **time-consuming** and **expensive**. Clustering enables the algorithm to learn and organize information without explicit supervision.
- **Data preprocessing:** Clustering helps in segmenting the data into meaningful groups, making it easier for subsequent tasks such as classification or regression.

What is good Clustering?

- Internal (within the cluster) distances/similarities should be small/big.
- External (intra-cluster) distances/similarities should be big/small.
- Typically, we need either
  - A similarity measure $s(x_i,x_k)$: large if x_i,x_k are similar
  - Dissimilarity (or distance) measure $d(x_i,x_k)$: small if x_i,x_k are similar
- We need a criterion function to evaluate a clustering
- We need an algorithm to compute the clustering
  - For example, by optimizing the criterion function

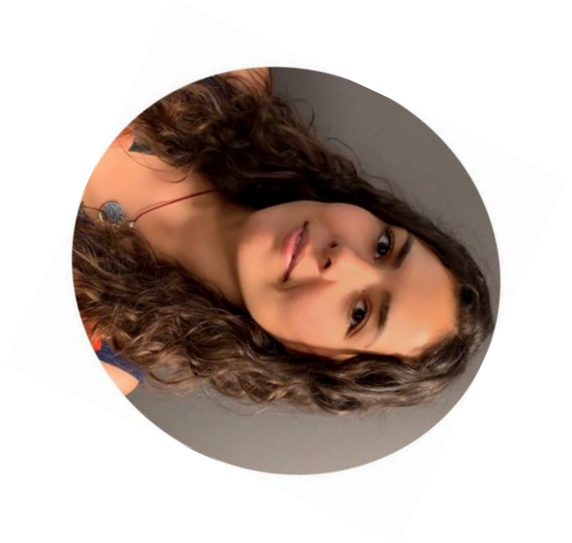
How many clusters?

- Some approaches fix the number of clusters
- Some finds the best clustering according to a criterion function; therefore, the number of clusters may vary.

Proximity Measures (Distance Function)

- Depends on application. For example,
  - Object recognition and face recognition should be invariant to rotation.

Distance =0



  - For character recognition: no invariance to rotation
    - For example, 9 and 6 have large distance

Distance (Dissimilarity) Measures

- **Euclidean distance** (numerical attributes):

$$Dist(x, x') = \sqrt{\sum_d |x_d - x'_d|^2}$$

  - Translation invariant, symmetric, spherical, all dimensions are equally treated
  - Sensitive to large differences in single attributes
- **Manhattan (city block) distance**
  - An approximation to Euclidean distance, cheaper to compute

$$Dist(x, x') = \sum_d |x_d - x'_d|$$

Distance (Dissimilarity) Measures

- **Hamming Distance (categorical attributes):**

$$Dist(x, x') = \sum_{d} \mathbf{1}_{x_d \neq x'_d}$$

  - Counts the number of attributes where $x \neq x'$
- **Cosine Similarity:**
  - The smaller the angle, the larger the similarity
  - Scale invariant measure
  - Very popular, also in deep learning

Distance (Dissimilarity) Measures

- **Cosine Similarity:**

$$\text{Cosine Similarity}(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^T \cdot \mathbf{x}'}{\|\mathbf{x}\|_2 \|\mathbf{x}'\|_2}$$

- $\mathbf{x}^T$ denotes the transpose of vector $\mathbf{x}$.
- $\mathbf{x}^T \cdot \mathbf{x}'$ represents the dot product of the transposed vector $\mathbf{x}$ and vector $\mathbf{x}'$.
- $\|\mathbf{x}\|_2$ and $\|\mathbf{x}'\|_2$ represent the Euclidean norms (vector magnitudes) of vectors $\mathbf{x}$ and $\mathbf{x}'$ respectively.

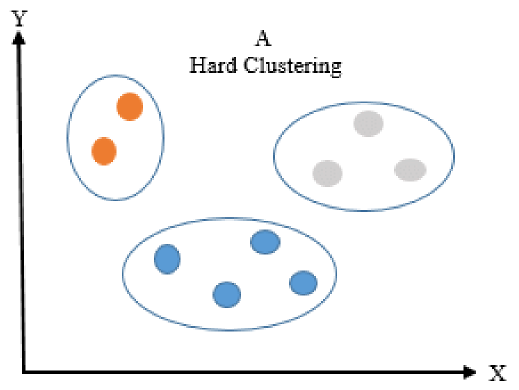
- The *cosine similarity value ranges* from -1 to 1.
- A value of 1 indicates that the vectors are identical, 0 means the vectors are orthogonal (no similarity), and -1 implies complete dissimilarity or opposition.

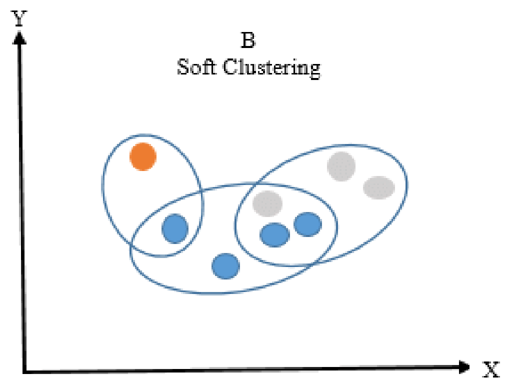
Clustering approaches

- Clustering methods can be divided into different categories as:

1) Hard clustering vs. Soft Clustering:

- **Hard Clustering:** Each data point belongs to exactly one cluster. There is no overlap between clusters.
- **Soft Clustering (Fuzzy Clustering):** Data points can belong to more than one cluster with varying degrees of membership (e.g., based on probability). It allows for cluster overlap, and each point has a degree of association with each cluster.

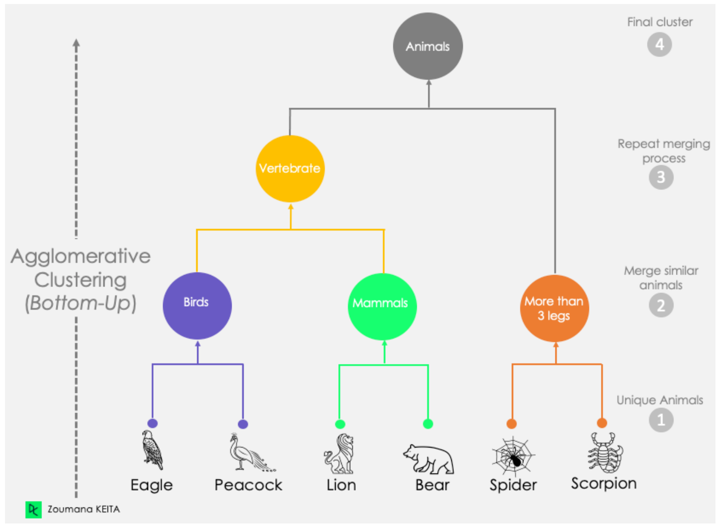




Clustering approaches

2) Partitioning Clustering: The data is divided into non-overlapping subsets or clusters. E.g., k-means and k-medoids

3) Hierarchical Clustering: This approach creates a tree of clusters, known as a dendrogram. It represents the hierarchy of relationships between clusters. Agglomerative (bottom-up) and divisive (top-down) are common hierarchical clustering methods.

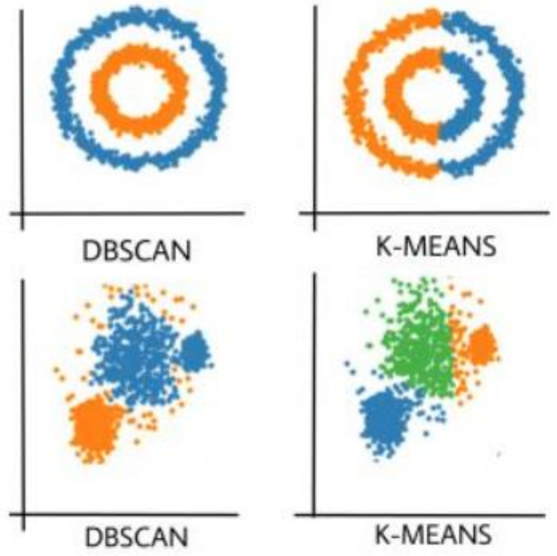


Clustering approaches

4) Density-Based Clustering: Clusters are defined as dense regions of data points separated by regions of lower point density.

E.g., DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- *While k-means tends to created circular clusters with more or less similar number of instances per cluster, DBSCAN can results in much different shapes of clusters while come clusters are denser compared to others.*

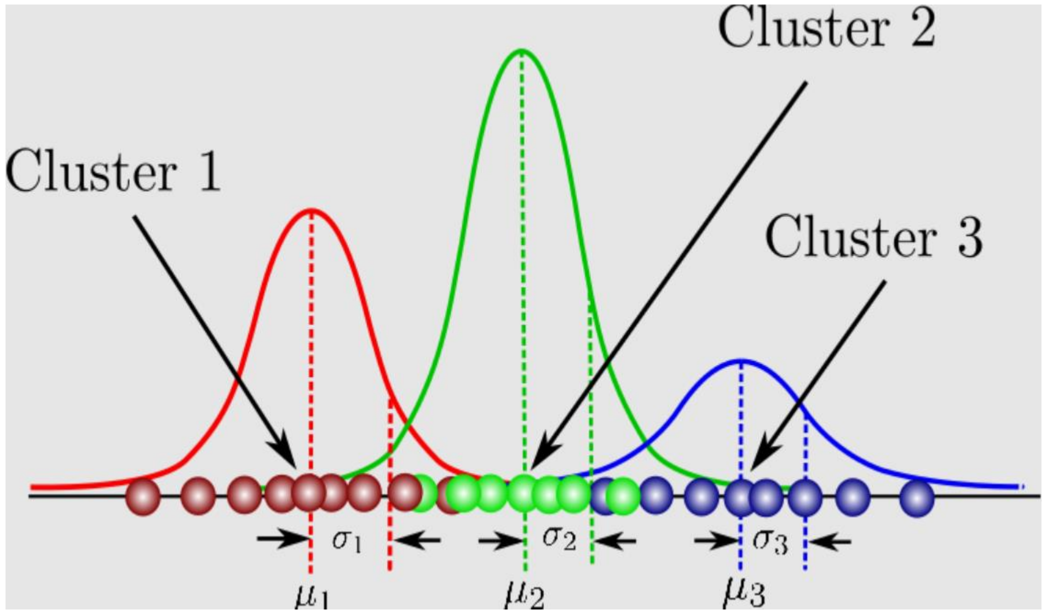


CLUSTERING APPROACHES

UNIVERSITÀ di VERONA

5) Centroid-Based Clustering: Clusters are represented by a central vector, often the mean (centroid), and data points are assigned to the cluster whose centroid is closest. E.g., k-means

6) Distribution-Based Clustering: Clusters are modeled using statistical distributions. E.g., Gaussian Mixture Models (GMM).



Gaussian Mixture Models (GMM)

In this example, we have the data and that data was fit into a GMM model composed of 3 modes, aka 3 clusters. Each has a mean and corresponding standard deviation.

Clustering approaches

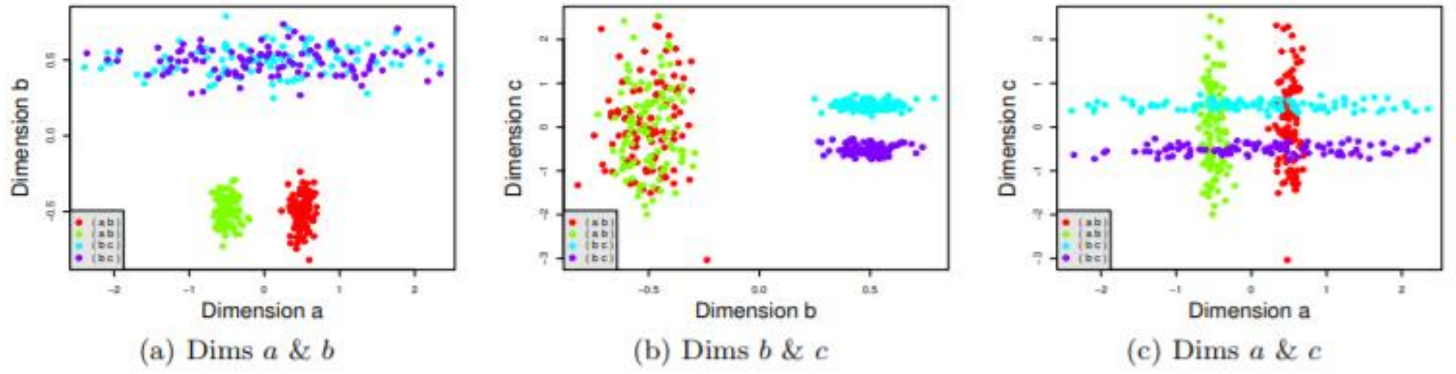
7) Graph-Based Clustering: Data points are treated as nodes in a graph, and clusters are identified as connected components. E.g., Spectral clustering

8) Subspace Clustering: Identifies clusters in subspaces of the data, considering only a subset of features for each cluster. Useful when clusters exist in different dimensions.

- Subspace clustering methods consider that not all features are equally relevant for all clusters. Different clusters may exhibit patterns in different subsets of features.

Clustering approaches

- Subspace clustering is particularly useful in handling high-dimensional data where many features may be irrelevant or redundant.
  - Focusing on relevant feature subsets can lead to more accurate and interpretable clustering results.

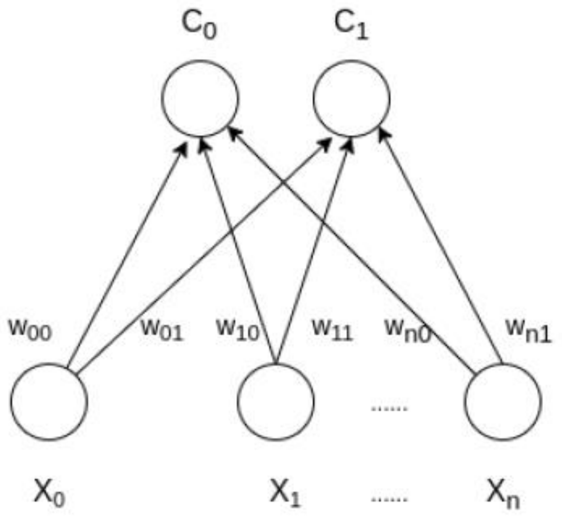


Sample data plotted in each set of two dimensions. In both (a) and (b) we can see that two clusters are properly separated, but the remaining two are mixed together. In (c) the four clusters are more visible, but still overlap each other are are impossible to completely separate.

For red and green data points feature space in dimension a is better, for violet and blue data points the feature dimension in c is better.

Clustering approaches

9) Self-Organizing Maps (SOM): A type of neural network-based clustering that projects high-dimensional data onto a lower-dimensional grid, preserving the topological relationships between data points. SOMs provide a powerful visualization tool.



K-Means

- The most well-known and popular clustering algorithm.
- We have to know a priori **how many clusters** we need (*k*).
- Fine with **numerical attributes**, but not with categorical attributes.
- **Fast!** *O(#iterations x #clusters x #instances x #dimensions)*
- K-means is a greedy algorithm that converges to a **local minimum**.
- Results can vary drastically based on **centroid selection**:
  - We can perform cluster search **multiple times** with random centroids to improve performance.

K-Means Algorithm

- Data set: $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, N observations of a random *D*-dimensional variable $\mathbf{x}$
- Goal: partition data set into ***K* clusters**, *K* fixed a priori
- A cluster is a group of data points whose **inter-point distances** are small as compared with the distances to **points outside of the cluster**
- Based on $\boldsymbol{\mu}_k$, D-dimensional vector, k=1,..,*K*, representing the **center of the clusters**

K-Means Algorithm

- For each data point $\mathbf{x}_n$ there is a set of binary indicator variables $r_{nk} \in \{0,1\}$, k=1,..,K describing which of the K clusters the data point $\mathbf{x}_n$ is assigned to
  - *Hard clustering*
- **Objective function:** $$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \left\| \mathbf{x}_n - \boldsymbol{\mu}_k \right\|^2$$

  sum of the squares of the distances of each data point to its assigned vectors $\boldsymbol{\mu}_k$

- The goal is now to find values of r_{nk} and $\boldsymbol{\mu}_k$ so as to **minimize *J***

K-Means Algorithm

- Iterative algorithm: **2 steps**, alternate optimization wrt $\boldsymbol{r}_{\mathbf{nk}}$ and $\boldsymbol{\mu}_k$
  - choose initial value for the $\boldsymbol{\mu}_k$, k=1,..,K
  - **Step 1:** minimize J wrt r_{nk} keeping $\boldsymbol{\mu}_k$ fixed
  - **Step 2:** minimize J wrt $\boldsymbol{\mu}_k$ keeping r_{nk} fixed
  - Repeat until **convergence** or a **limited number of iterations**

K-Means Algorithm

- **Determination of r_{nk}**
  - J is a linear function of r_{nk}, so closed form solution
  - optimise each $\mathbf{x}_n$ separately, choose r_{nk} to be 1 for whichever value of k gives the minimum value of

$$\left\|\mathbf{x}_n - \boldsymbol{\mu}_k\right\|^2 \qquad r_{nk} = \begin{cases} 1 & \text{if } k = \operatorname{argmin}_j \left\|\mathbf{x}_n - \boldsymbol{\mu}_j\right\|^2 \\ 0 & \text{otherwise} \end{cases}$$

K-Means Algorithm

- **Determination of μ_k**
  - J is a quadratic function of $\boldsymbol{\mu}_k$, so solution is given by setting its derivative (wrt $\boldsymbol{\mu}_k$) to zero

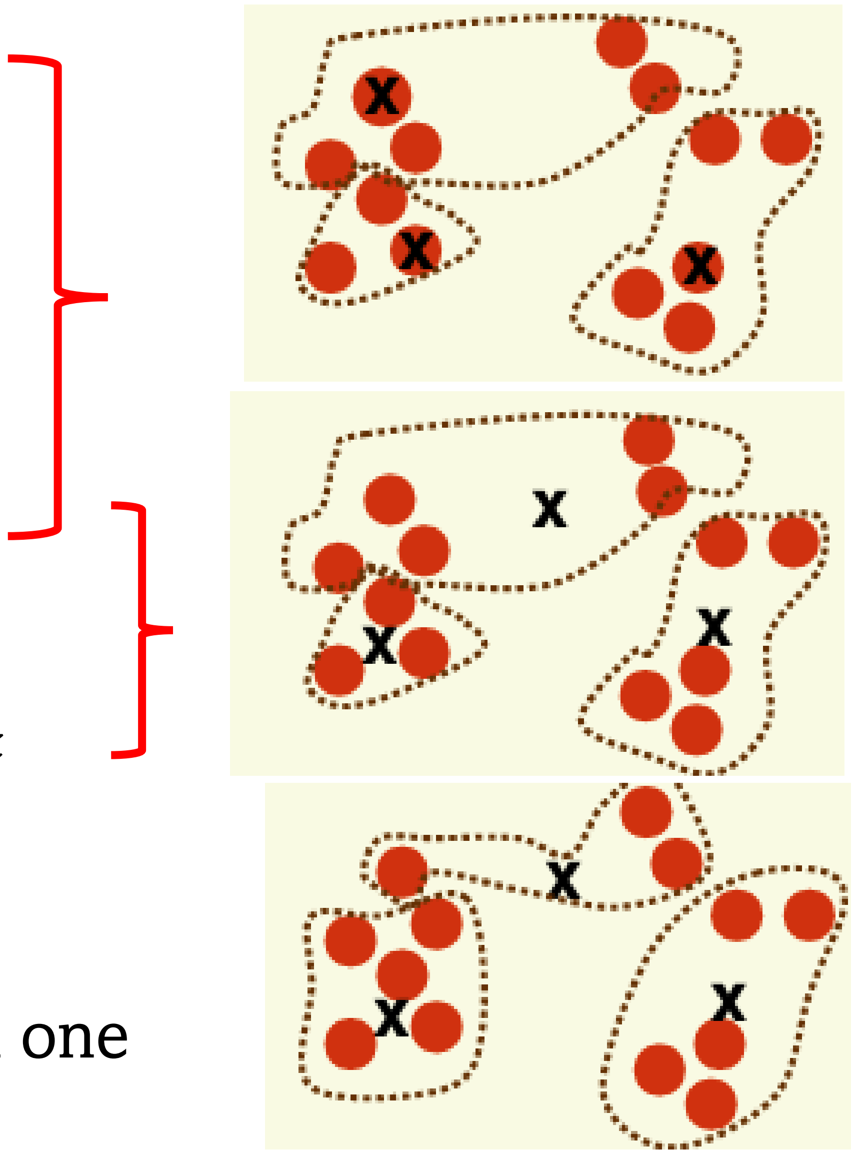
$$2\sum_{n=1}^{N} r_{nk}(\mathbf{x}_n - \boldsymbol{\mu}_k) = 0 \Rightarrow \boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\mathbf{x}_n}{\sum_n r_{nk}}$$

  where the denominator is equal to the number of points assigned to cluster k → K-means

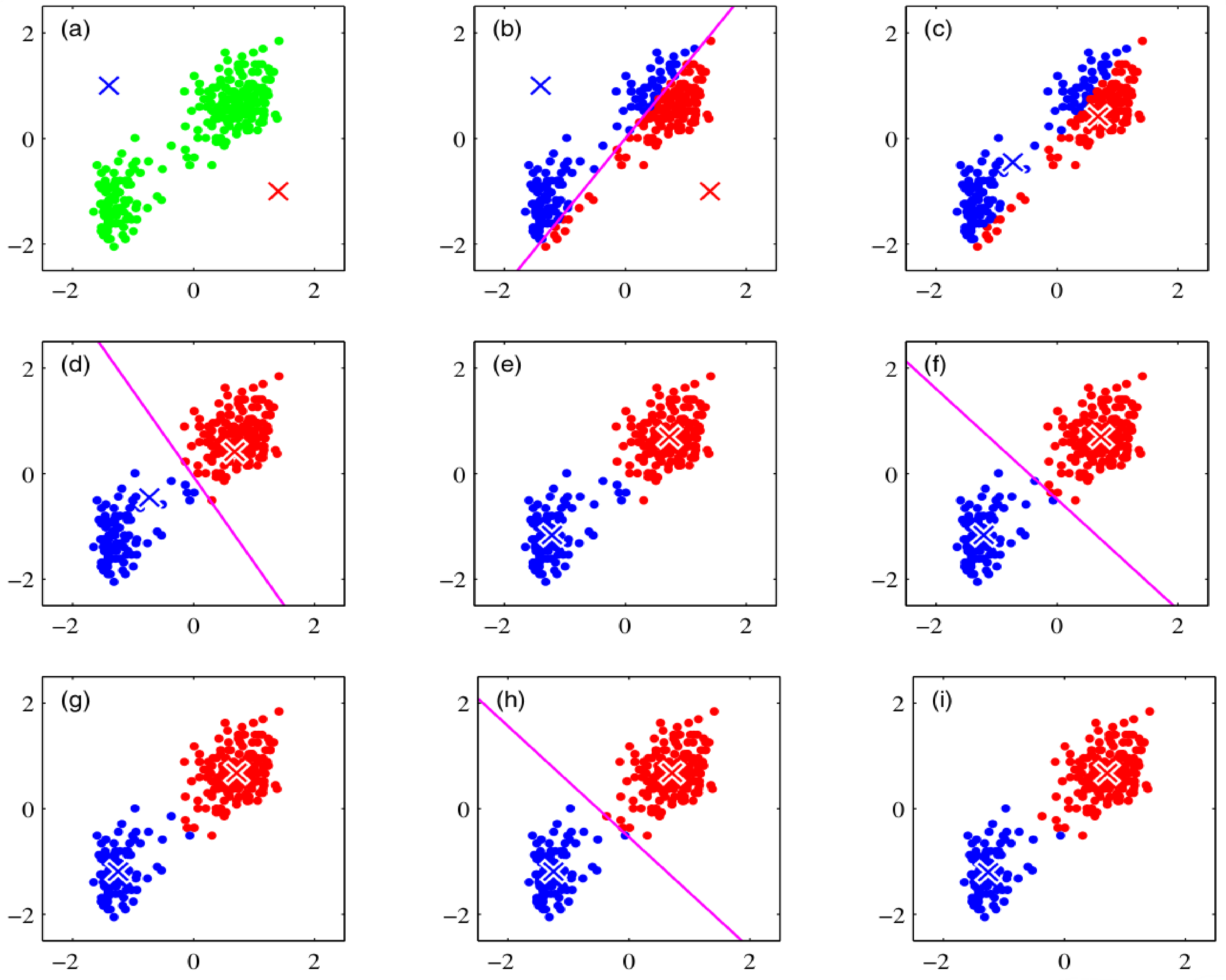
- Stop when there is no further change or after some maximum number of iterations
- May converge to a *local* (than global) minimum of J
- Issues: fast implementation, initialization of the cluster centers

K-Means Alternative Algorithm

- Initialization:
  - “randomly” select K centroids μ_k, k=1,..,K
- Recursion:
  1. For each data point $x_i \in X$
     - find the nearest centroid μ_j
     - Assign point x_i to cluster C_j
  2. For each cluster $j = 1, \dots, k$
     - Compute the centroid $\mu_j = \frac{1}{|C_j|}\sum_{x_k \in C_j} x_k$
  3. Go to 1
- Termination:
  - Convergence (i.e. all clusters unchanged in one iteration)



K-Means



K-Means Properties

- Results can vary drastically based on **centroid selection**
- Some can result in poor convergence rate or convergence to **sub-optimal clustering**
- How to manage? Common heuristics:
  - Choose random points (not examples)
  - Choose the data points least similar to any existing center (called furthest centers heuristic)
  - Try out multiple starting points (i.e., applying k-means several times)
  - Initialize the cluster centers with the results of another clustering method

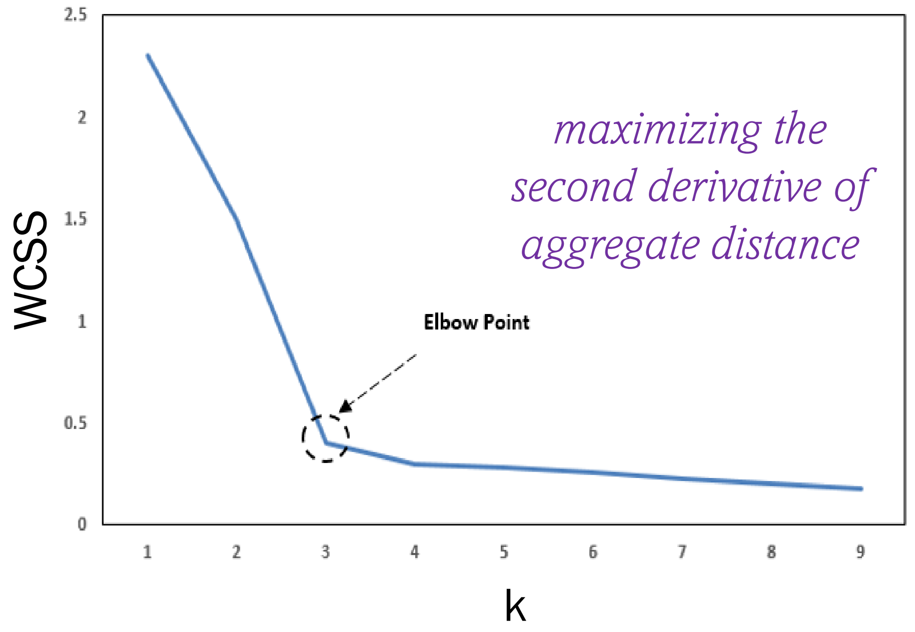
K-Means: number of clusters

- Knowledge about the data and the problem.
  - E.g., digit clustering: 0…9, K=10
- Data visualization (especially in cases with low cardinality)
- Run k-means several times with different k values, and evaluate each of them with some metrics
- **Elbow rule:** plotting the explained variation as a function of the number of clusters and looking for an "elbow" point in the graph.
  - *The explained variation is usually measured as the sum of squared distances from each point to its assigned cluster center.*

K-Means: Elbow rule

$$WCSS(k) = \sum_{i=1}^{k} \sum_{j=1}^{n_i} \|x_{ij} - c_i\|^2$$

k is the number of clusters.
n_i is the number of data points in the i-th cluster.
x_{ij} is the j-th data point in the i-th cluster.
c_i is the centroid of the i-th cluster.



- The "elbow" point is the value of ***k*** where adding more clusters provides diminishing returns in terms of reducing the sum of squared distances. It's the point where the **rate of decrease sharply changes**, forming an "elbow" shape in the graph.
- # of clusters at the elbow point is often considered as the **optimal choice** because it represents a **balance between model simplicity** and **capturing meaningful patterns** in the data.

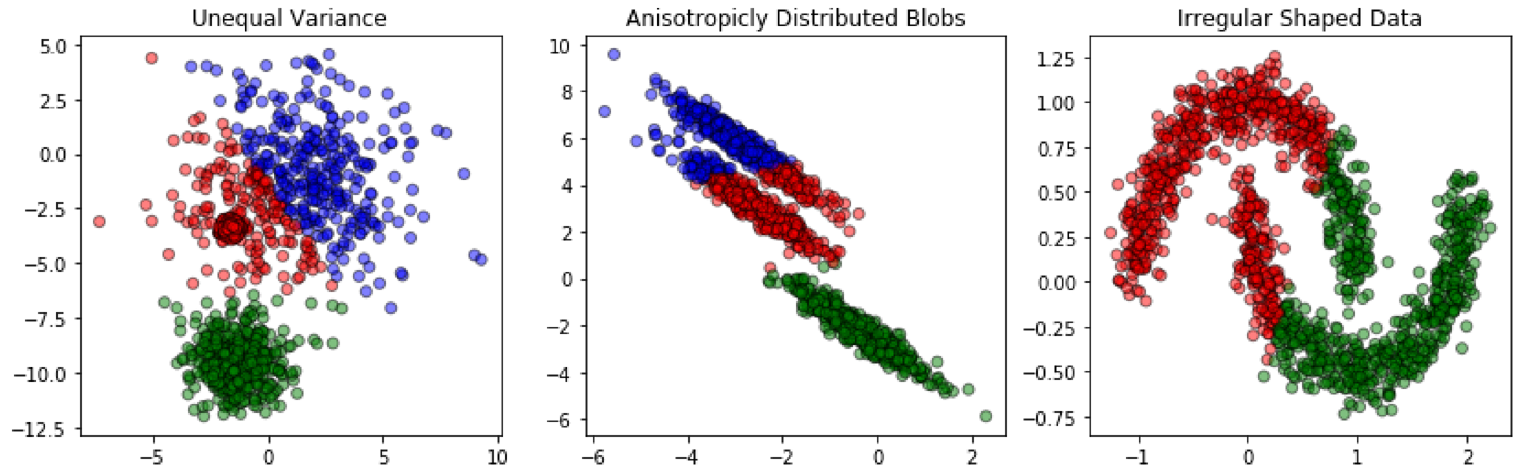
K-Means: Drawbacks

The K-means algorithm may not perform well when

a) clusters are **not spherical** and/or **varying sizes**.

b) if clusters have **different densities** (e.g., high-density cluster means many data points are closely packed together, low-density cluster means the data points are more spread out).

c)Outliers can significantly influence the centroid calculation, leading to suboptimal cluster assignments.

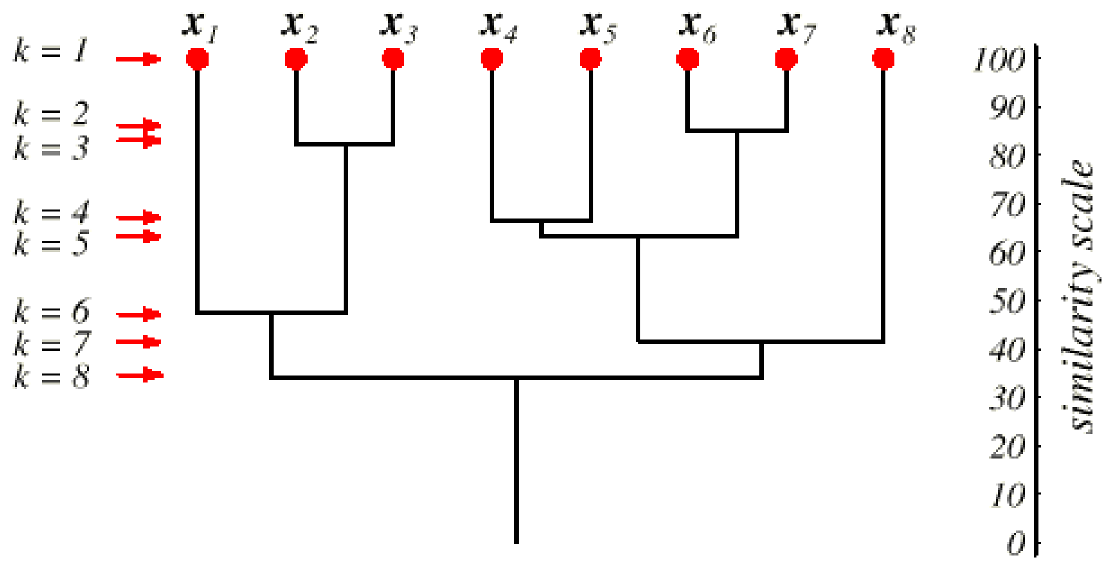


Hierarchical Clustering

- K-means is a flat clustering method, but
  - For some data, hierarchical clustering is more appropriate than flat clustering, e.g., when you have taxonomy in the data
    - Plant ➔ Seed producing (apple, rose) | spose producing (mushroom, mold)

Hierarchical Clustering

- Represented through a dendrogram
  - Binary tree-like diagram
  - Shows the similarities of grouped clusters
  - Level ***k*** corresponds to partitioning with ***n-k+1*** clusters
  - If you need ***k*** clusters, then take clustering from level ***n-k+1***
  - If samples are in the same cluster at level ***k***, they are in the same cluster at higher levels.



Hierarchical Clustering

- 2 types
  - **Agglomerative**: start with each point in a different cluster (singleton clusters) and iteratively merge them to bigger groups (*bottom-up*)
  - **Divisive**: start with all data points belonging to one single cluster and split them iteratively in smaller groups (*top-down*)

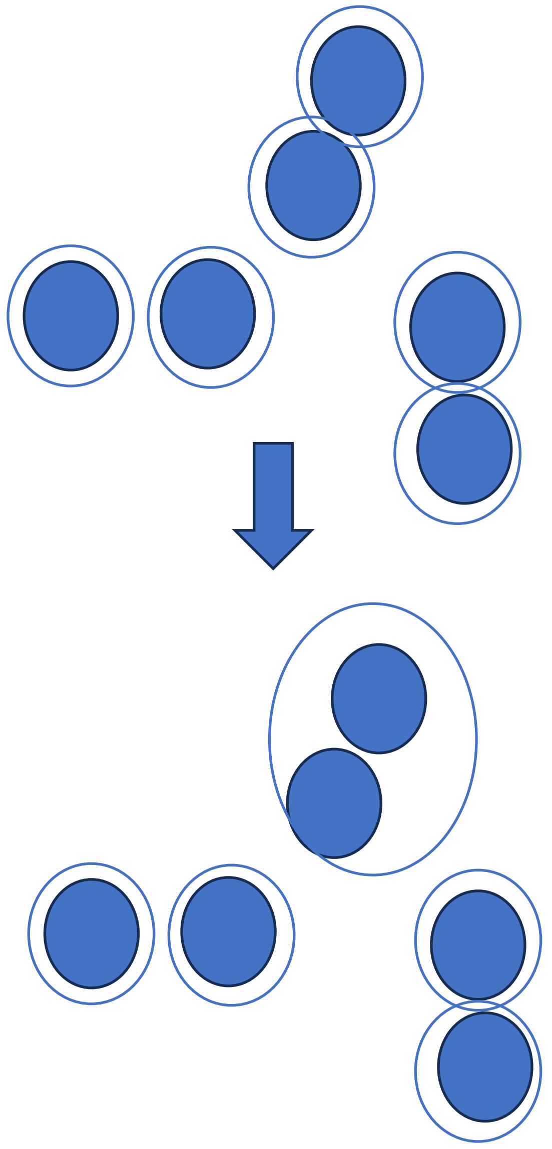
Agglomerative Hierarchical Clustering

- Given ***Data points*** $\boldsymbol{x_1, x_2, \ldots, x_n}$, $\boldsymbol{k}$ is the index of the current level where it ranges from ***1*** to ***n-1***.

- Initialize ***n*** **clusters**, each containing one data point.

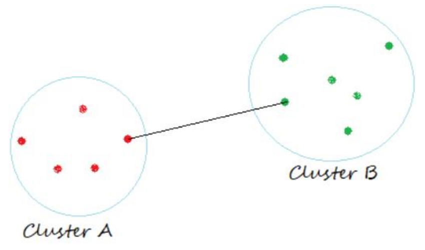
- Compute the pairwise **distance matrix D** between all data points.

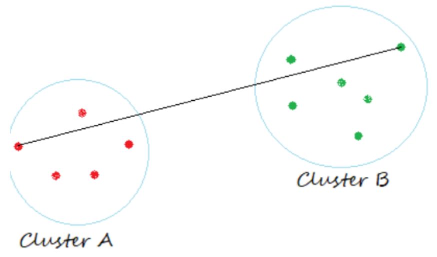
```
For k = 1 to n-1:
    - Identify the two closest clusters $C_i$ and $C_j$ based
      on the distance matrix D.
    - Merge clusters $C_i$ and $C_j$ into a single cluster $C_{ij}$.
    - Update the distance matrix D to reflect the
      distances between the new cluster $C_{ij}$ and the
      remaining clusters using chosen linkage criterion.
Until c clusters
```

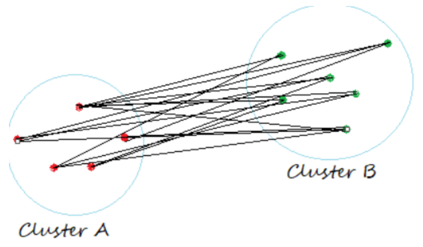


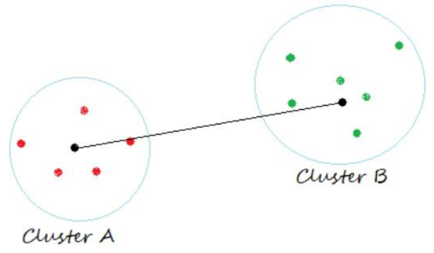
Agglomerative Hierarchical Clustering

- 4 ways to calculate cluster distance
  - Minimum distance $$d_{\min}(D_i, D_j) = \min_{\mathbf{x}\in D_i, \mathbf{x}'\in D_j} \|\mathbf{x} - \mathbf{x}'\|$$
  - Maximum distance $$d_{\max}(D_i, D_j) = \max_{\mathbf{x}\in D_i, \mathbf{x}'\in D_j} \|\mathbf{x} - \mathbf{x}'\|$$
  - Average distance $$d_{avg}(D_i, D_j) = \frac{1}{n_i n_j} \sum_{\mathbf{x}\in D_i} \sum_{\mathbf{x}'\in D_j} \|\mathbf{x} - \mathbf{x}'\|$$
  - Mean distance $$d_{mean}(D_i, D_j) = \|\mathbf{m}_i - \mathbf{m}_j\|$$









They all behave quite similarly when the clusters are hyperspherical and well separated.

Agglomerative Hierarchical Clustering

- Single Linkage or Nearest Neighbor refers to using minimum distance criterion.

$$d_{\min}(D_i, D_j) = \min_{\mathbf{x}\in D_i, \mathbf{x}'\in D_j} \|\mathbf{x} - \mathbf{x}'\|$$

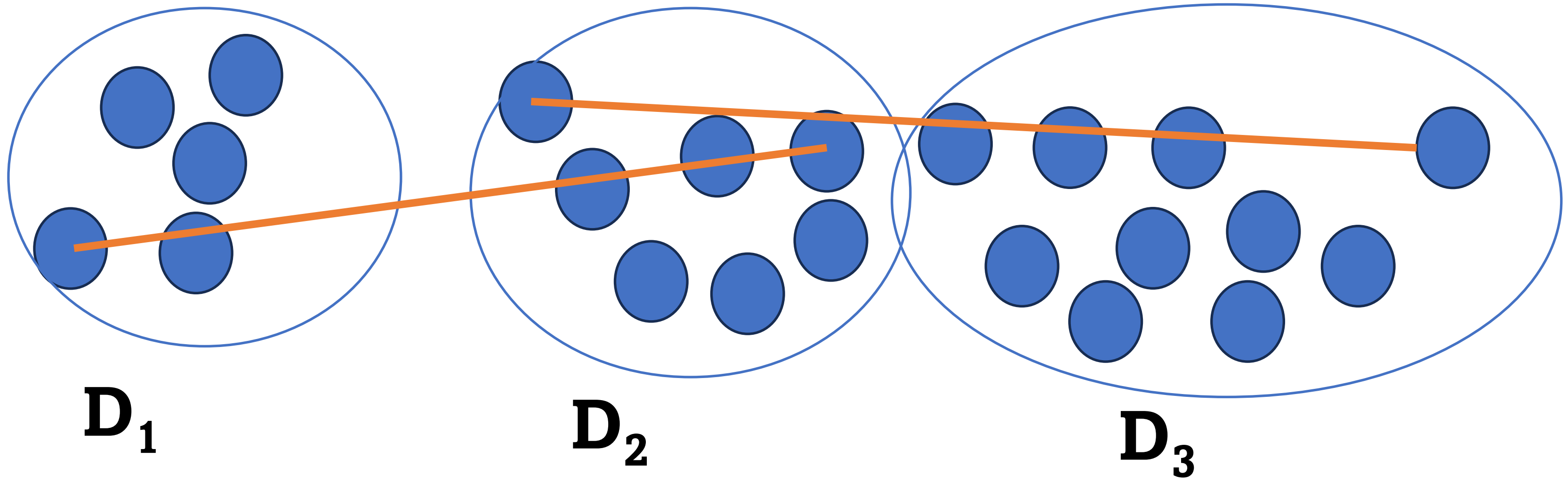
  - Generates minimum spanning tree
  - Encourages growth of elongated clusters (shaped like ellipsoids)
  - Disadvantage: very sensitive to noise / outliers

Agglomerative Hierarchical Clustering

- Complete Linkage or Farthest Neighbor refers to using maximum distance criterion.

$$d_{\max}(D_i, D_j) = \max_{\mathbf{x} \in D_i, \mathbf{x}' \in D_j} \|\mathbf{x} - \mathbf{x}'\|$$

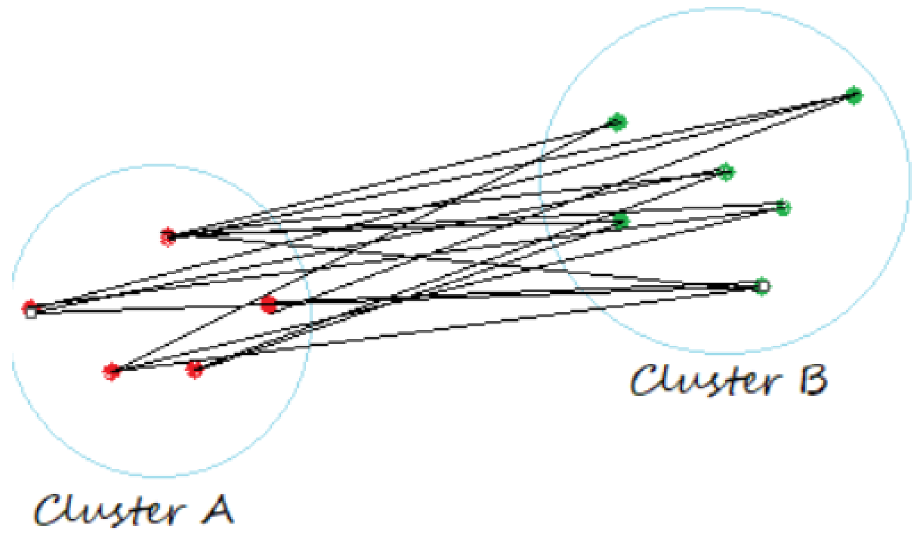
  - Results in compact clusters
  - Does not work well if elongated clusters present

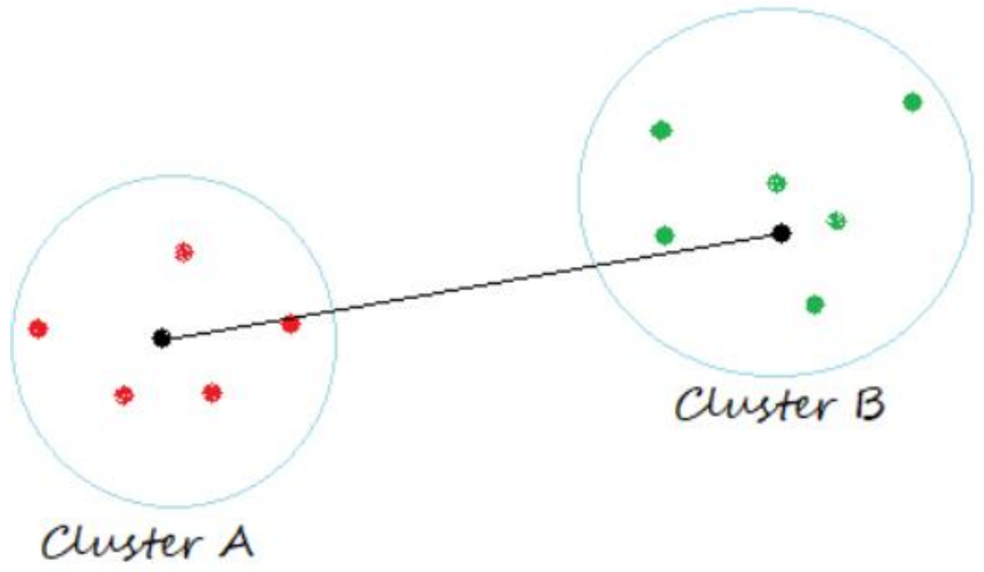


***dmax*(D$_1$, D$_2$) < *dmax*(D$_2$, D$_3$) thus, D$_1$, D$_2$ are merged**

Agglomerative Hierarchical Clustering

- It is more robust under the average or the mean cluster distance
- Mean distance is cheaper





Divisive Hierarchical Clustering

1. Split into clusters using any flat-clustering method, e.g., K-means.
2. Choose the best cluster among the clusters to split further,

   Choose the one that has the largest Sum of Squared Error (SSE)

$$SSE_k = \sum_{i=1}^{c} \sum_{x_j \in C_i} ||x_j - c_i||^2$$

- $||x_j - c_i||^2$ is the squared Euclidean distance between data point x_j and the centroid c_i of cluster C_i.
- c is the number of clusters at level k.
- k is the level in the hierarchy.

3. Repeat steps 2 and 3 until a single cluster is formed.

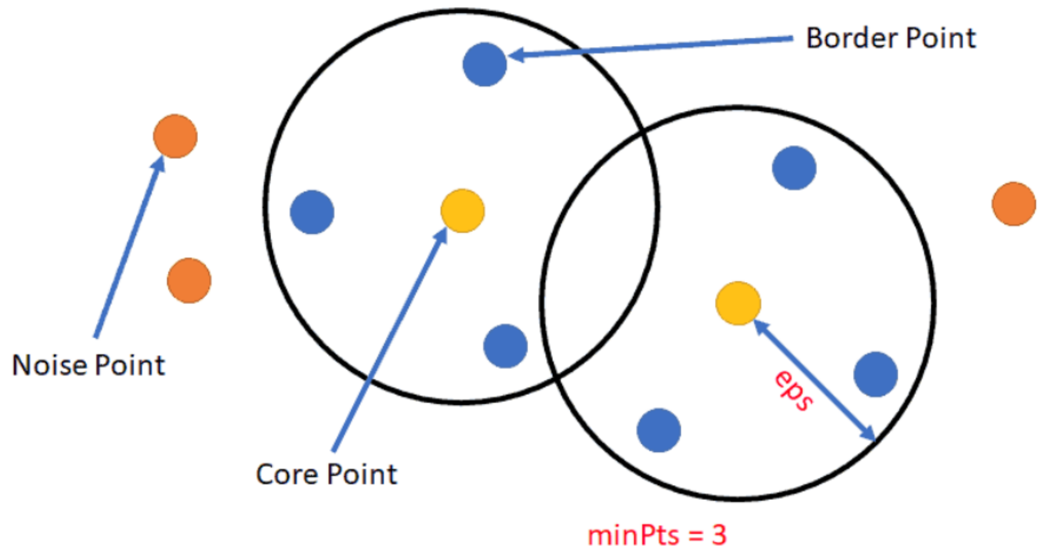
Hierarchical Clustering

- It has **high time** and **space computational complexity**.
  - For computing proximity matrix, the time complexity is $O(N^2)$, since it takes N steps to search, the total time complexity is $O(N^3)$
- There is **no objective function** for hierarchical clustering.
- Due to high time complexity, it **cannot** be used for **large datasets**.
- It is sensitive to **noise** and **outliers** since we use distance metrics.
- It has **difficulty** handling **large clusters**.

DBSCAN: Density-Based Spatial Clustering of Applications with Noise

- A density-based clustering algorithm that finds clusters by grouping data points that are closely packed together, based on a specified **distance** and **density threshold**.
- Particularly useful for identifying clusters of arbitrary shapes and for detecting noise (outliers) in the data.

DBSCAN



Two parameters:
•**eps (Epsilon, density threshold):** The radius defining the neighborhood around a data point.
•**MinPts (Distance threshold):** The minimum number of data points required to form a dense region (core point).

There are three types of points:
Core Points: the point has at least a specified number of points (**MinPts**) within a specified radius (**eps**).
Border Points: The point that is within the neighborhood (**eps**) of a core point but does not have enough neighbors to be considered a core point itself.
Noise Points: A point is a noise point (or outlier) if it is neither a core point nor a border point.

DBSCAN - Algorithm

1. For each (unvisited) data point, determine its neighborhood based on **eps**.

2. If the number of neighbors is greater than or equal to **MinPts**, create a new cluster and add the data point and its neighbors to the cluster.

3. Expand the cluster by recursively checking the neighbors' neighbors, following the same **MinPts**.

4. Repeat steps 1 to 3 for all unvisited data points.

DBSCAN - LIMITATIONS

- Sensitive to the choice of parameter.
  - Selecting appropriate values for these parameters can be challenging and may require domain knowledge or trial and error.

- Might suffer with clusters of varying densities or with high-dimensional data.

- Performance may degrade as the size of the dataset increases.

DBSCAN - Advantages

- Can find clusters of arbitrary shapes, unlike K-means, which tends to work best with spherical and equally sized clusters.

- Can identify outliers or anomalies.

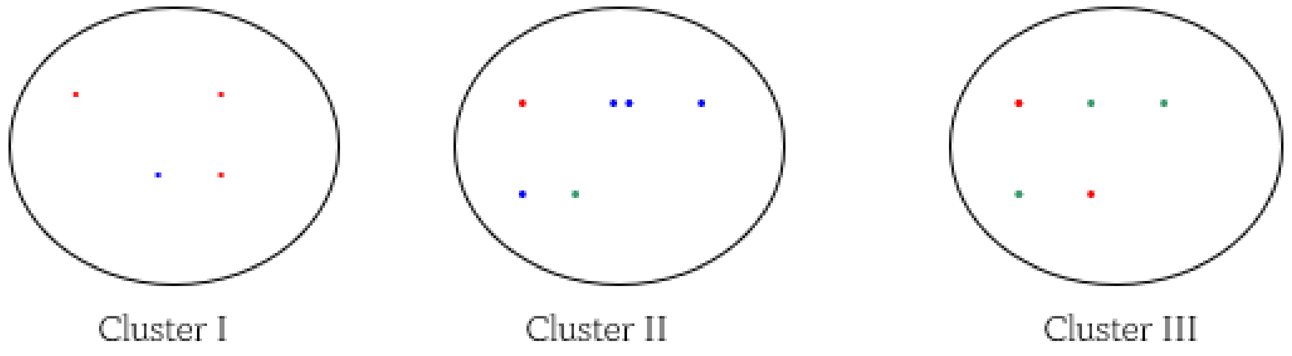
- No need to specify the number of clusters a priori.

Clustering evaluation

- Extrinsic measures *(help us to solve another problem, e.g., a classification problem)*
  - Represent images with cluster features
  - Train different classifiers for each cluster
  - For example, we use **purity** as the measure
- Intrinsic measures *(useful in and of itself)*
  - What is a good clustering? A good clustering will produce high-quality clusters in which the **intra-cluster similarity is high**, and the **inter-cluster similarity is low**.
  - Clusters correspond to classes (digits $\rightarrow$ 10 clusters): Align, evaluate as a normal classifier

Extrinsic measures

- **Purity:** The proportion of the dominant class in the cluster



Cluster I: Purity = (max(3, 1, 0)) / 4 = 3/4
Cluster II: Purity = (max(1, 4, 1)) / 6 = 4/6
Cluster III: Purity = (max(2, 0, 3)) / 5 = 3/5

Cluster average: $$\frac{\frac{3}{4}+\frac{4}{6}+\frac{3}{5}}{3} = 0.672$$

Weighted average: $$\frac{4*\frac{3}{4}+6*\frac{4}{6}+5*\frac{3}{5}}{15} = \frac{3+4+3}{15} = 0.667$$

- Good for comparing two algorithms, but not understanding how well a single algorithm is doing.
- Why? Increasing the number of clusters increases the purity

Intrinsic measures

- *Davies Bouldin Index:* Measures the compactness and separation between clusters.
  - Calculated based on the distances between **cluster centroids** and the **average distance** of **each point** in a cluster to **its centroid**.
- A lower DBI indicates better clustering, where clusters are more compact and well-separated
- A higher DBI suggests poorer clustering with more overlap between clusters or clusters being too spread out.

Davies Bouldin Index

- For a set of clusters $C=\{C1,C2,\ldots,Cn\}$, the DBI is computed as:

$$DBI = \frac{1}{n}\sum_{i=1}^{n} \max_{j\neq i}\left(\frac{\sigma_i+\sigma_j}{d(c_i,c_j)}\right)$$

- n is the number of clusters.
- ci and cj are the centroids of clusters Ci and Cj, respectively.
- σi and σj represent the average distance between points in clusters Ci and Cj to their respective centroids.
- $d(ci,cj)$ is the distance between centroids ci and cj.

Intrinsic measures

- *Silhouette Index:* A measure of how well-separated clusters are, as well as how similar each data point is to its own cluster compared to other clusters.
  - The silhouette score ranges from -1 to $+1$, where a **high value** indicates that the object is **well-matched** to its **own cluster** and **poorly matched** to **neighboring clusters**.
  - If most objects have a **high value**, then the clustering configuration is **appropriate**.
  - If many points have **a low or negative value**, then the clustering configuration may have **too many** or **too few clusters**.

Silhouette Index

- *Let* ***a*****(*i*)**, the average distance from data point ***i*** to **other points in the same cluster**, when **|*Ci*|** represents the number of data points in cluster ***Ci*** and ***d*****(*i,j*)** is the distance between data points ***i*** and ***j***.

$$a(i) = \frac{1}{|C_i|-1} \sum_{j \in C_i, j \neq i} d(i,j)$$

- ***b(i)*** the smallest average distance from data point ***i*** to points in a different cluster, minimized over clusters to which ***i*** does not belong, when **|*Ck*|** represents the number of data points in cluster ***Ck***, ***d*****(*i,j*)** is the distance between data points ***i*** and ***j*** and the minimum is taken over all clusters ***Ck*** where ***i*** does not belong.

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$$

Silhouette Index

- While *a(i)* and *b(i)* measure the intra-cluster cohesion and inter-cluster separation, respectively, for each data point *i*:

- The silhouette score for an individual data point ***i*** is calculated as:

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

- The silhouette score for a clustering is the a**verage of the silhouette scores of all data points**:

$$\text{Silhouette Score} = \frac{1}{N} \sum_{i=1}^{N} s(i)$$

where ***N*** is the total number of data points.

Intrinsic measures

Dunn Index: Measures the compactness of clusters (intra-cluster similarity) and the separation between clusters (inter-cluster dissimilarity).

$$\text{Minimum Inter-Cluster Distance} = \min_{i \neq j} d(c_i, c_j)$$

where *ci* and *cj* are centroids of different clusters, *d*(*ci*,*cj*) represents the distance between centroids *ci* and *cj*, which could be any distance metric, such as Euclidean distance.

$$\text{Maximum Intra-Cluster Distance} = \max_i \left(\max_{x,y \in C_i, x \neq y} d(x, y) \right)$$

where Ci represents a cluster, *x* and *y* are data points within cluster *Ci* and *d*(*x*,*y*) represents the distance between data points *x* and *y* within the same cluster *Ci*.

Dunn Index

$$\text{Dunn Index} = \frac{\text{Minimum inter-cluster distance}}{\text{Maximum intra-cluster distance}}$$

- Similar to other clustering evaluation metrics, the Dunn Index should be used in conjunction with domain knowledge and other validation measures for a comprehensive assessment of clustering quality.

Cigdem Beyan
cigdem.beyan@univr.it
https://cbeyan.github.io/