Machine Learning (CE 40477)

Fall 2024

Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

October 15, 2024

Unsupervised Learning

- **Unsupervised Learning** involves analyzing unlabeled data to uncover hidden patterns or structures within the data

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 features.
- **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

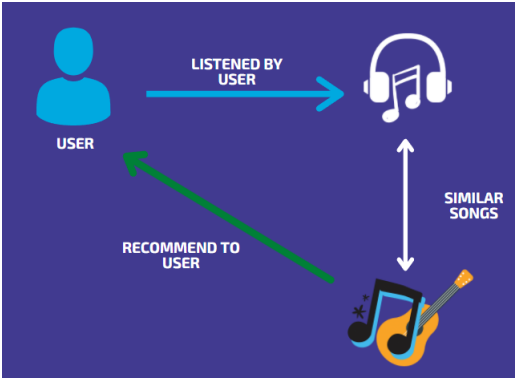
- Clustering organizes data points into groups of similar objects.
- Data points in a cluster are more similar to each other than to those in other clusters.
- The notion of similarity depends on the task at hand (e.g., purchase behavior in market segmentation).

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 in Action: Music Recommendation Systems

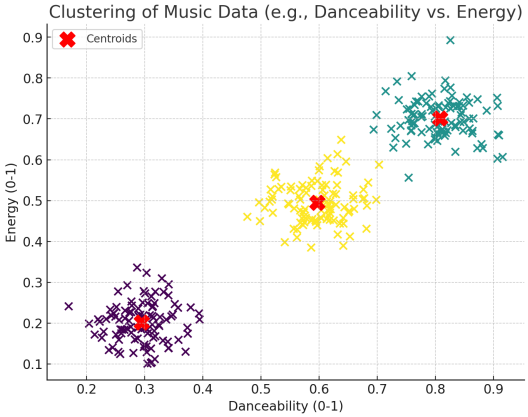
- Music recommendation systems cluster songs based on similarity.



Adopted from machinelearninggeek.com

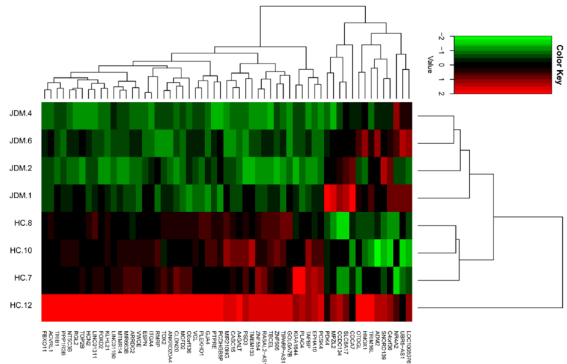
Clustering in Action: Music Recommendation Systems

- 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.



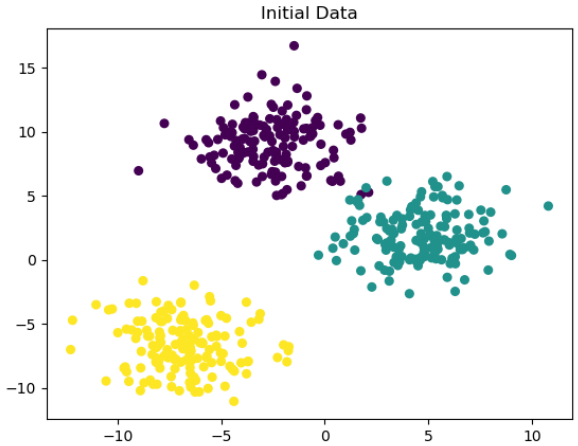
Two Beginning Questions

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

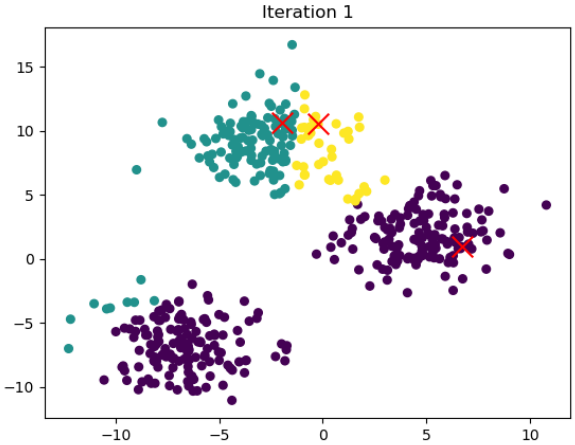
K-Means overview

- 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 process repeats until the assignments no longer change

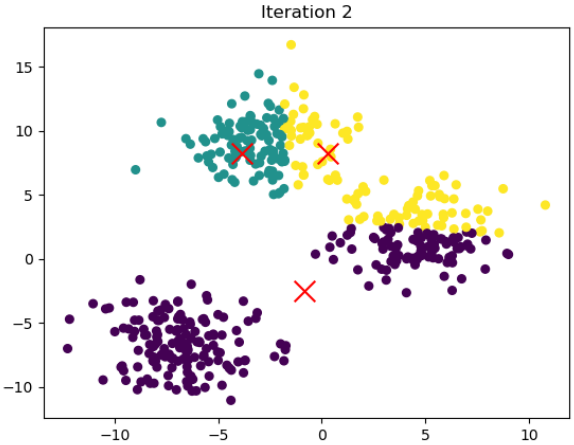
K-Means in action



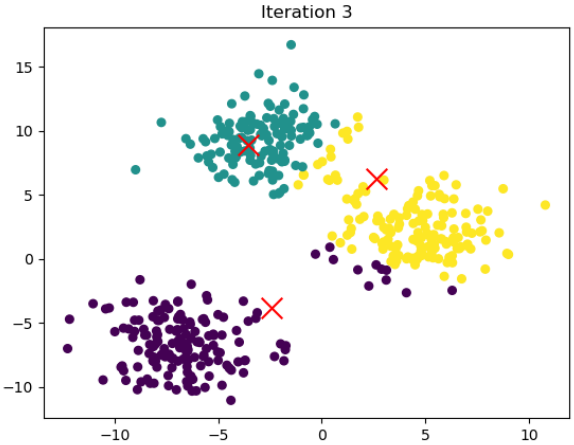
K-Means in action



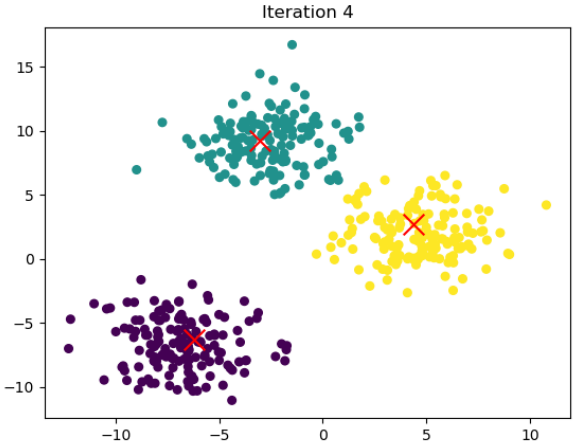
K-Means in action (cont.)



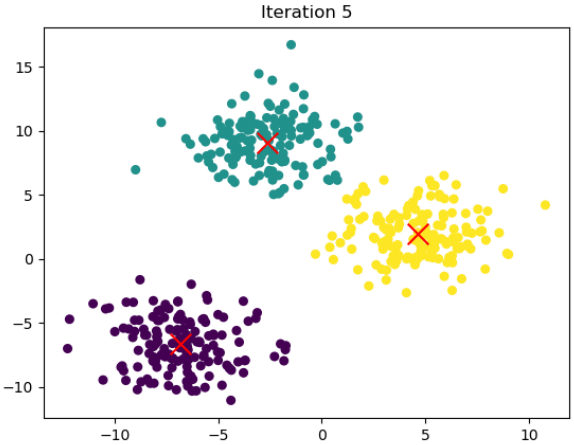
K-Means in action (cont.)



K-Means in action (cont.)



K-Means in action (cont.)



Algorithm

Algorithm 1 K-means Clustering

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

Problem definition

- Formally: We have $X_{\text{train}} = \{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\} \subseteq \mathbb{R}^d$
- K is the number of clusters.
- We are learning:
  1. A function or mapping $f : \mathbb{R}^d \to \{1, 2, \ldots, K\}$ that assigns a cluster to each data point.
  2. A set of K prototypes $\mu = \{\mu_1, \mu_2, \ldots, \mu_K\} \subseteq \mathbb{R}^d$ as the cluster representatives, called **centeroids**.

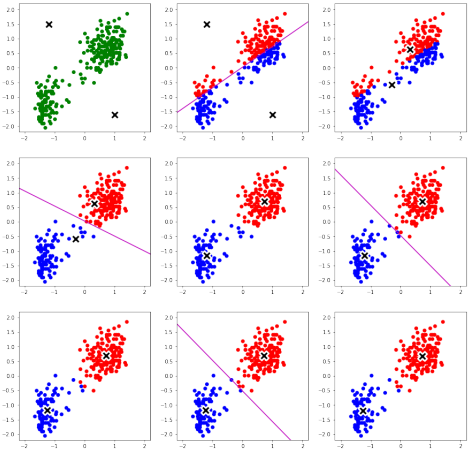
Objective Function

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

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

- Choose f and $\mu = \{\mu_1, \mu_2, \ldots, \mu_K\}$ to minimize this.
- This problem is NP-hard. K-Means is a heuristic solution, which is NOT guaranteed to find optimal solution.

K-Means Process Example



Adopted from mlbhanuyerra.github.io

Convergence

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

Convergence (cont.)

- We initially assigne each sample to the nearest centroid.

$$f(x) := argmin_j ||x - \mu_j||^2$$

.

- Keep each sample's assignment fixed until a closer centriod 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 (cont.)

- In Updating step, with $f(x)$ fixed, J is a quadratic function of μ_j (like SSE) and by taking derivative we can minimize it as:

$$\frac{\partial J}{\partial \mu_j} = 0 \Longrightarrow \sum_{x^{(i)} \in C_j} 2\left(x^{(i)} - \mu_j\right) = 0$$

- This means we should **update** each μ_j as the mean of cluster C_j:

$$\mu_j = \frac{\sum_{x^{(i)} \in C_j} x^{(i)}}{|C_j|}$$

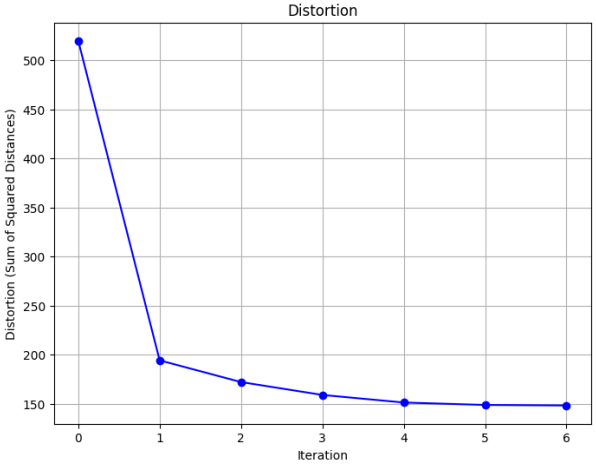
Convergence (cont.)

- For each cluster, the mean of its samples minimizes squared distances.
- For C_j if μ'_j was the old centroid we have: $\sum_{x^{(i)} \in C_j} ||x^{(i)} - \mu'_j||^2 \geq \sum_{x^{(i)} \in C_j} ||x^{(i)} - \mu_j||$. So $J_{\text{new}} \leq J_{\text{old}}$.

Convergence (cont.)

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

K-Means Convergence (cont.)



Strengths

- Simple: easy to understand and to 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.

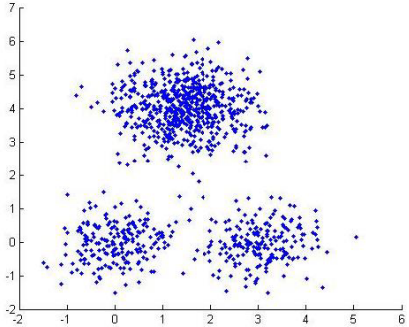
Initialization

- K-Means always converges. What could go wrong ?
- K-Means algorithm is a **heuristic**
- It requires initial centroids, and the choice is important as it could affect the t in $O(tkn)$.

Local Optimum

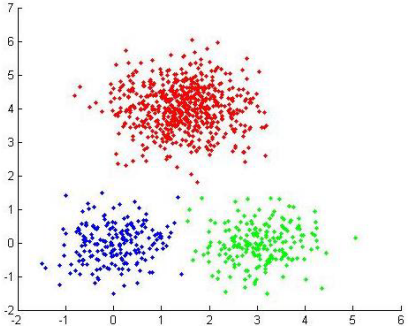
- The algorithm finds a local minimum but there is no guarantee to find global minimum.
- Its result is highly affected by the initialization.
- Some suggestions are:
  - Multiple runs with random initial centroids, then select the "best" result.
  - Initialization heuristics (K-Means++ , Furthest Traversal).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

Local Optimum

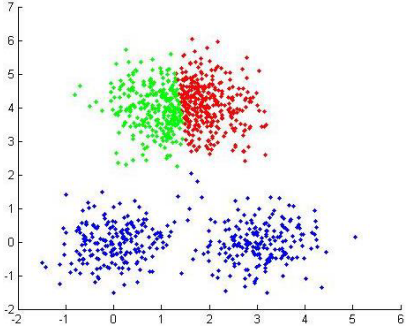


=

Local optimum (cont.)



Optimal clustering

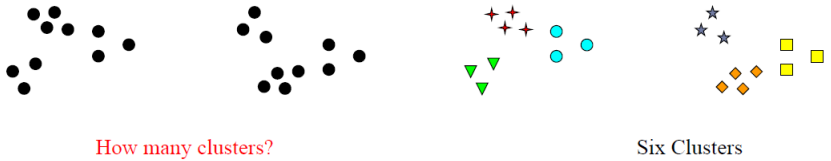


Possible clustering

Definition of Mean

- We assume $x^{(i)} \in \mathbb{R}^d$, which is not always the case. K-Means requires a space where sample **mean** is defined.
  - Categorical data.
  - A suggested solution: K-Mode - the centroid is the most frequent category (the mode) in each cluster.
  - Closest centroid is found by the Hamming Distance.

How many clusters?



Adopted from

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

How many clusters? (cont.)

- Number of clusters is usually given in advance in the problem of clustering. However; finding the right number of clusters is also a problem.
- 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):

$$WCSS = \sum_{i=1}^{K} \sum_{x \in C_i} ||x - \mu_i||^2$$

Clustering Evaluation

- **Inter-cluster separation (isolation)**: How different the data points are between clusters.
  - Single-link (Minimum Distance):
  - Measures the **minimum distance** between any two points from different clusters.

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

  - Complete-link (Maximum Distance):
  - Measures the maximum distance between any two points from different clusters.

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

Clustering Evaluation

- **Inter-cluster separation (isolation)**: How different the data points are between clusters.
  - Centroid (Wards Method):
  - Measures the distance between the centroids of two clusters.

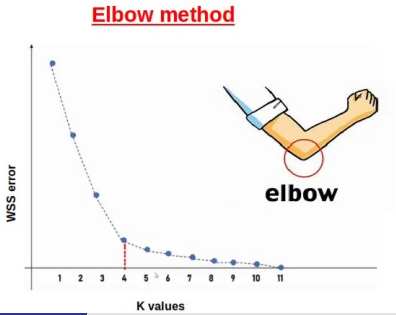
$$d_{\text{centroid}}(C_i, C_j) = d(\mu_i, \mu_j)$$

  - Average-link:
  - Measures the average distance between all pairs of points from different clusters.

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

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 (resembles 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? (cont.)

- 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.

Data Distribution

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

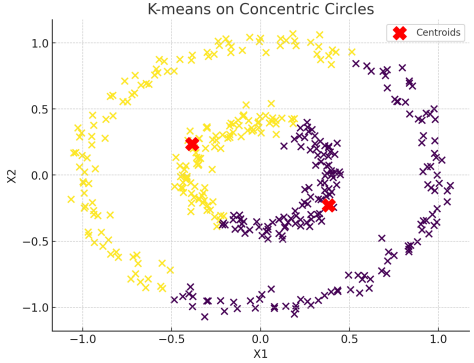


Figure 1: example when k-means wont work

Hard vs Soft Clustering

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

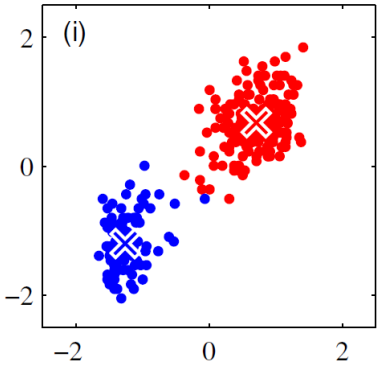


Figure adapted from Machine Learning and Pattern Recognition, Bishop

Hard vs Soft Clustering (cont.)

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

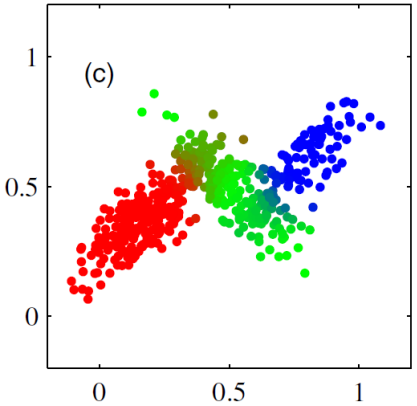
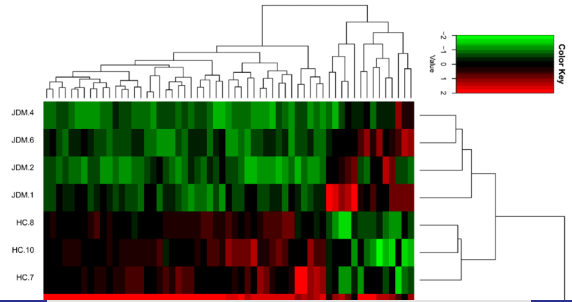


Figure adapted from Machine Learning and Pattern Recognition, Bishop

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.

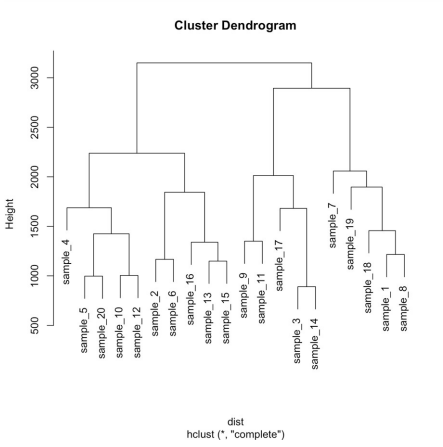


Agglomerative Clustering Algorithm

- Start with each point as its own cluster.
- Merge the "closest" clusters.
- Repeat until one cluster remains or desired number is reached.
- Closest cluster can be determined using inter-cluster separation measures

Dendrogram and Cutting

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



Adopted from r-graph-gallery.com

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.

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.

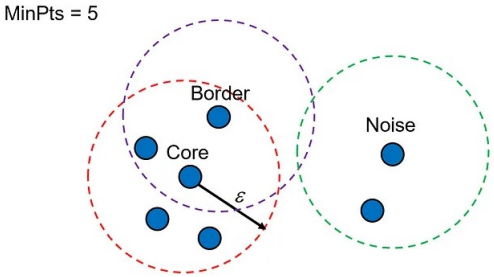
Parameters:

- ϵ (epsilon): Maximum distance for neighbors.
- `minPts`: Minimum points to form a dense region.

Core Concepts in DBSCAN

DBSCAN defines three types of points:

- **Core Point**: A point with at least `minPts` neighbors within distance ϵ.
- **Border Point**: A point within ϵ of a core point but with fewer than `minPts` neighbors.
- **Noise**: Points that are neither core points nor border points.



Adopted from ai.plainenglish.io

Core Concepts in DBSCAN (cont.)

Definitions:

- A point x_i is a core point if:

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

- A point is a border point if it is within distance ϵ of a core point, but not itself a core point.

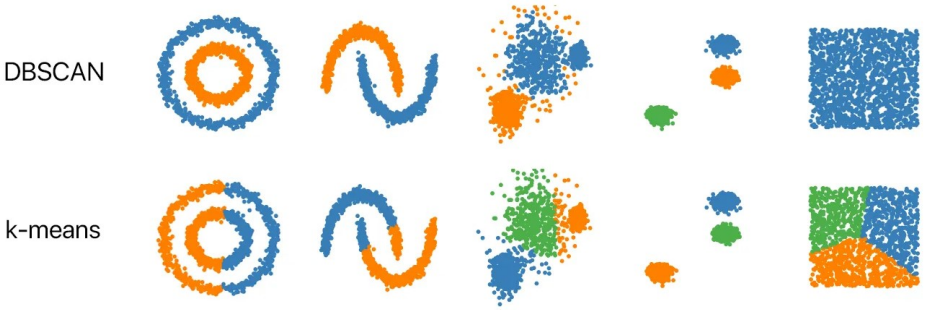
DBSCAN Algorithm Steps

Algorithm Steps:

1. For each unvisited point x_i:
   - Mark x_i as visited.
   - Find all points within distance ϵ (neighborhood).
2. If x_i is a core point:
   - Create a new cluster and expand it by recursively adding all reachable core and border points.
3. 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.



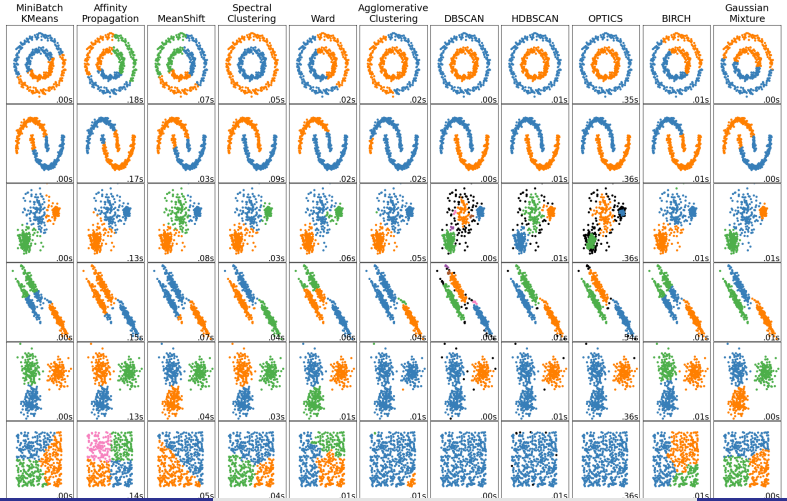
Adopted from

Limitations of DBSCAN

- DBSCAN struggles with datasets of varying densities.
- Sensitive to the selection of parameters ϵ 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.



Adopted from

Contributions

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