

MO433 - Unsupervised Learning

Data Clustering

Alexandre Xavier Falcão

Institute of Computing - UNICAMP

afalcao@ic.unicamp.br

Data Clustering

Assuming suitable object features, clustering techniques aim to group data points into clusters such that:

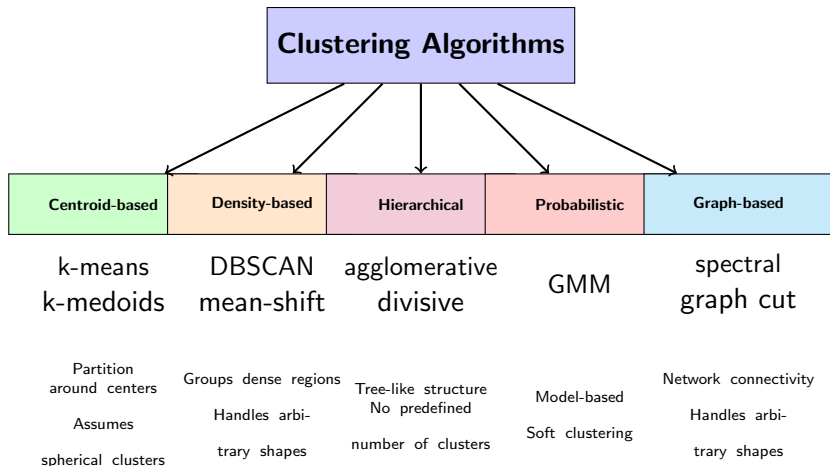
- ▶ Objects within the same cluster are similar to each other.
- ▶ Objects in different clusters are dissimilar to each other.

Characteristics:

- ▶ Discovers hidden structure in unlabeled data.
- ▶ “Good” clustering depends on the application.
- ▶ Results vary with choice of metric.

Applications: Image segmentation, gene analysis, document organization, anomaly detection, data compression.

Clustering Taxonomy



Each approach makes different assumptions about cluster structure, density, and geometry.

Agenda

Methods:

- ▶ **K-Means** – Centroid-based partitioning.
- ▶ **DBSCAN** – Density-based clustering with noise detection.
- ▶ **Agglomerative** – Hierarchical bottom-up clustering.
- ▶ **Gaussian Mixture Models (GMM)** – Probabilistic clustering.
- ▶ **Spectral** – Graph-based eigenvalue clustering.

Metrics:

- ▶ **ARI** (Adjusted Rand Index) – Clustering similarity.
- ▶ **NMI** (Normalized Mutual Information) – Information theory.
- ▶ **Silhouette Score** – Within-cluster cohesion vs separation.

Applied to: DINO-extracted features from the corel dataset.

K-Means Clustering

Objective Function:

$$J = \sum_{i=1}^n \sum_{j=1}^k w_{ij} ||x_i - \mu_j||^2$$

Where:

- ▶ $w_{ij} = 1$ if x_i belongs to cluster j , else 0.
- ▶ μ_j is the centroid of cluster j .

Centroid Update:

$$\mu_j = \frac{\sum_{i=1}^n w_{ij} x_i}{\sum_{i=1}^n w_{ij}}$$

Characteristics:

- ▶ **Assumes:** Spherical clusters of similar size.
- ▶ **Distance:** Typically Euclidean.
- ▶ **Output:** Hard clustering (each point belongs to one cluster).
- ▶ **Complexity:** $O(n \cdot k \cdot i \cdot d)$ for i = iterations, d = dimensions.

K-Means Algorithm

Algorithm 1 K-Means

Require: Dataset X , number of clusters k

Ensure: Cluster assignments, centroids

- 1: Initialize k centroids $\mu_1, \mu_2, \dots, \mu_k$ randomly
 - 2: **repeat**
 - 3: Assign each point x_i to nearest centroid:
 - 4: $c_i = \arg \min_j ||x_i - \mu_j||^2$
 - 5: Update centroids:
 - 6: $\mu_j = \text{mean}(\text{all points assigned to cluster } j)$
 - 7: **until** convergence
 - 8: **return** cluster assignments and centroids
-

Advantages: Simple and fast, works well with globular clusters, and guaranteed convergence.

Disadvantages: Must specify k , sensitive to initialization, and struggles with non-spherical clusters.

DBSCAN Clustering

ε -neighborhood of point p :

$$N_{\varepsilon}(p) = \{q \in D \mid \text{dist}(p, q) \leq \varepsilon\}$$

Core point: $|N_{\varepsilon}(p)| \geq \text{MinPts}$.

Directly density-reachable: $q \in N_{\varepsilon}(p)$ and p is core.

Density-reachable: Chain of directly density-reachable points.

Density-connected: \exists point o such that both p and q are density-reachable from o .

Characteristics:

- ▶ **Parameters:** ε (eps) and MinPts.
- ▶ **Distance:** Any metric (Euclidean, cosine, etc.).
- ▶ **Output:** Clusters + noise points (-1 label).
- ▶ **Complexity:** $O(n \log n)$ with spatial indexing, $O(n^2)$ worst case.

DBSCAN Algorithm - Part 1

Algorithm 2 DBSCAN - Main Loop

Require: Dataset X , ε (radius), MinPts (minimum points)

Ensure: Cluster labels (including noise as -1)

- 1: Initialize all points as unvisited
 - 2: **for** each unvisited point p **do**
 - 3: Mark p as visited
 - 4: Find neighbors $N = \{q \mid \text{dist}(p, q) \leq \varepsilon\}$
 - 5: **if** $|N| < \text{MinPts}$ **then**
 - 6: Mark p as noise
 - 7: **else**
 - 8: Create new cluster C
 - 9: Add p to C
 - 10: **ExpandCluster**($p, N, C, \varepsilon, \text{MinPts}$) {See next slide}
 - 11: **end if**
 - 12: **end for**
-

DBSCAN Algorithm - Part 2

Algorithm 3 DBSCAN - ExpandCluster Function

Require: Core point p , neighbors N , cluster C , ε , MinPts

Ensure: Expanded cluster C

```
1: for each point  $q$  in  $N$  do
2:   if  $q$  is unvisited then
3:     Mark  $q$  as visited
4:     Find  $q$ 's neighbors  $N' = \{r \mid \text{dist}(q, r) \leq \varepsilon\}$ 
5:     if  $|N'| \geq \text{MinPts}$  then
6:        $N = N \cup N'$  {Expand search region}
7:     end if
8:   end if
9:   if  $q$  is not in any cluster then
10:    Add  $q$  to cluster  $C$ 
11:   end if
12: end for
```

Agglomerative Clustering

Union-Find Data Structure:

Core Operations:

- ▶ **Find(x)**: Returns root of set containing x (with path compression).
- ▶ **Union(x,y)**: Merges sets containing x and y (with union by rank).

Path Compression:

$$\text{Find}(x) = \begin{cases} x & \text{if } \text{parent}[x] = x \\ \text{parent}[x] = \text{Find}(\text{parent}[x]) & \text{otherwise} \end{cases}$$

Linkage Criteria:

- ▶ **Single**: $d(C_i, C_j) = \min_{x \in C_i, y \in C_j} \|x - y\|$.
- ▶ **Complete**: $d(C_i, C_j) = \max_{x \in C_i, y \in C_j} \|x - y\|$.
- ▶ **Average**: $d(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i, y \in C_j} \|x - y\|$.
- ▶ **Ward**: $d(C_i, C_j) = \sqrt{\frac{2|C_i||C_j|}{|C_i|+|C_j|}} \|\mu_i - \mu_j\|$.

Agglomerative Algorithm - Part 1

Algorithm 4 Agglomerative Clustering - Initialization

Require: Dataset X , linkage criterion, target clusters k

Ensure: Cluster hierarchy and assignments

- 1: Initialize: Each point as its own cluster
 - 2: **for** $i = 1$ to n **do**
 - 3: $\text{parent}[i] = i, \text{rank}[i] = 0$
 - 4: **end for**
 - 5: Compute all pairwise distances and sort by distance
-

Agglomerative Algorithm - Part 2

Algorithm 5 Union-Find Helper Functions:

Find(x):

- 1: **if** $\text{parent}[x] \neq x$ **then**
- 2: $\text{parent}[x] = \text{Find}(\text{parent}[x])$ {Path compression}
- 3: **end if**
- 4: **return** $\text{parent}[x]$

Union(x, y):

- 1: $\text{root}_x, \text{root}_y = \text{Find}(x), \text{Find}(y)$
- 2: **if** $\text{rank}[\text{root}_x] < \text{rank}[\text{root}_y]$ **then**
- 3: $\text{parent}[\text{root}_x] = \text{root}_y$
- 4: **else if** $\text{rank}[\text{root}_x] > \text{rank}[\text{root}_y]$ **then**
- 5: $\text{parent}[\text{root}_y] = \text{root}_x$
- 6: **else**
- 7: $\text{parent}[\text{root}_y] = \text{root}_x$
- 8: $\text{rank}[\text{root}_x] = \text{rank}[\text{root}_x] + 1$
- 9: **end if**

Agglomerative Algorithm - Part 3

Algorithm 6 Agglomerative Clustering - Main Merging Loop

Require: Sorted edge list, initialized Union-Find structure

```
1: for each edge  $(i, j)$  in sorted order do
2:    $root_i = \text{Find}(i)$ 
3:    $root_j = \text{Find}(j)$ 
4:   if  $root_i \neq root_j$  then
5:     Compute linkage distance  $d(C_i, C_j)$ 
6:     Record merge:  $(root_i, root_j, \text{distance})$ 
7:      $\text{Union}(root_i, root_j)$ 
8:     if number of clusters =  $k$  then
9:       stop
10:    end if
11:  end if
12: end for
```

Gaussian Mixture Models (GMM)

Likelihood Function:

$$p(x) = \sum_{k=1}^K \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

Gaussian Distribution:

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

Posterior Probability (Responsibility):

$$\gamma_{ik} = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(x_i|\mu_j, \Sigma_j)}$$

Concepts:

- ▶ **Parameters:** $\{\pi_k, \mu_k, \Sigma_k\}$ for each component k .
- ▶ **Soft Assignments:** Each point has probability of belonging to each cluster.
- ▶ **EM Training:** Iteratively optimizes likelihood (covered in previous lecture).

GMM Model Selection

Model Selection:

- ▶ **AIC:** $AIC = -2\ln(L) + 2p$ (penalizes complexity).
- ▶ **BIC:** $BIC = -2\ln(L) + p\ln(n)$ (stronger penalty).

L is the maximum likelihood value, p is the number of parameters, and n is the number of samples.

GMM Model Selection

Model Selection:

- ▶ **AIC:** $AIC = -2 \ln(L) + 2p$ (penalizes complexity).
- ▶ **BIC:** $BIC = -2 \ln(L) + p \ln(n)$ (stronger penalty).

L is the maximum likelihood value, p is the number of parameters, and n is the number of samples. **Characteristics:**

- ▶ **Model:** Mixture of Gaussian distributions.
- ▶ **Output:** Soft clustering (probabilistic assignments).
- ▶ **Shapes:** Can model elliptical clusters.
- ▶ **Complexity:** $O(n \cdot k \cdot d^2 \cdot i)$ per iteration.

GMM Model Selection

Model Selection:

- ▶ **AIC:** $AIC = -2\ln(L) + 2p$ (penalizes complexity).
- ▶ **BIC:** $BIC = -2\ln(L) + p\ln(n)$ (stronger penalty).

L is the maximum likelihood value, p is the number of parameters, and n is the number of samples. **Characteristics:**

- ▶ **Model:** Mixture of Gaussian distributions.
- ▶ **Output:** Soft clustering (probabilistic assignments).
- ▶ **Shapes:** Can model elliptical clusters.
- ▶ **Complexity:** $O(n \cdot k \cdot d^2 \cdot i)$ per iteration.

Advantages:

- ▶ Provides cluster probabilities.
- ▶ Can model elliptical clusters.
- ▶ Principled probabilistic framework.

GMM Model Selection

Model Selection:

- ▶ **AIC:** $AIC = -2 \ln(L) + 2p$ (penalizes complexity).
- ▶ **BIC:** $BIC = -2 \ln(L) + p \ln(n)$ (stronger penalty).

L is the maximum likelihood value, p is the number of parameters, and n is the number of samples. **Characteristics:**

- ▶ **Model:** Mixture of Gaussian distributions.
- ▶ **Output:** Soft clustering (probabilistic assignments).
- ▶ **Shapes:** Can model elliptical clusters.
- ▶ **Complexity:** $O(n \cdot k \cdot d^2 \cdot i)$ per iteration.

Advantages:

- ▶ Provides cluster probabilities.
- ▶ Can model elliptical clusters.
- ▶ Principled probabilistic framework.

Disadvantages:

- ▶ Must specify number of components (k).
- ▶ Assumes Gaussian distributions.
- ▶ Can converge to local optima.

Spectral Clustering: Graph Laplacian mathematics

Adjacency Matrix: $W_{ij} = \text{similarity}(x_i, x_j)$

Degree Matrix: $D_{ii} = \sum_{j=1}^n W_{ij}$ (diagonal matrix)

Unnormalized Laplacian:

$$L = D - W$$

Normalized Laplacians (other options):

- ▶ **Symmetric:** $L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$.
- ▶ **Random Walk:** $L_{rw} = D^{-1} L = I - D^{-1} W$.

Solves a eigenvalue problem: $Lv = \lambda v$.

- ▶ Using L_{sym} : For symmetric W , the symmetry ensures that all eigenvalues are real and eigenvectors are orthogonal.
- ▶ Using L_{rw} : Not symmetric, but its entries (i, j) represent the transition probabilities of moving from node i to j (a random walk on the graph to study diffusion processes).

Spectral Clustering Algorithm

Algorithm 7 Spectral Clustering

Require: Dataset X , number of clusters k , similarity function

Ensure: Cluster assignments

- 1: Construct similarity graph: $W[i, j] = \text{similarity}(x_i, x_j)$
 - 2: Compute degree matrix: $D[i, i] = \sum_j W[i, j]$
 - 3: Compute Laplacian: $L = D - W$ (or normalized version)
 - 4: Eigendecomposition:
 - 5: Find k smallest eigenvalues $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_k$
 - 6: Get corresponding eigenvectors v_1, v_2, \dots, v_k
 - 7: Form matrix $Y = [v_1 \ v_2 \ \dots \ v_k] \in \mathbb{R}^{n \times k}$
 - 8: Normalize rows: $Y[i, :] = Y[i, :] / \|Y[i, :]\|_2$
 - 9: Apply k-means to rows of Y
 - 10: **return** cluster assignments
-

Spectral Clustering Algorithm

Similarity Functions:

- ▶ **RBF Kernel:** $W_{ij} = \exp(-\gamma ||x_i - x_j||^2)$.
- ▶ **k-NN:** $W_{ij} = \text{similarity}(x_i, x_j)$ if $x_j \in \text{kNN}(x_i)$, else 0.
- ▶ **ε -neighborhood:** $W_{ij} = 1$ if $||x_i - x_j|| < \varepsilon$, else 0.

Evaluation Metrics

Adjusted Rand Index (ARI):

Contingency Table: n_{ij} = number of points in true cluster i and predicted cluster j .

$$ARI = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}}{\frac{1}{2} \left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2} \right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}}$$

where: $a_i = \sum_j n_{ij}$, $b_j = \sum_i n_{ij}$.

- **Range:** $[-1, 1]$, where 1 = perfect agreement, 0 = random assignment.
- **Advantage:** Chance-corrected, handles different cluster sizes.

Evaluation Metrics

Adjusted Rand Index (ARI):

Contingency Table: n_{ij} = number of points in true cluster i and predicted cluster j .

$$ARI = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}}{\frac{1}{2} \left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2} \right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}}$$

where: $a_i = \sum_j n_{ij}$, $b_j = \sum_i n_{ij}$.

- ▶ **Range:** $[-1, 1]$, where 1 = perfect agreement, 0 = random assignment.
- ▶ **Advantage:** Chance-corrected, handles different cluster sizes.

Normalized Mutual Information (NMI):

Notation:

- ▶ U = true clustering (ground truth labels).
- ▶ V = predicted clustering (algorithm output).
- ▶ U_i = the i -th cluster in the true clustering.
- ▶ V_j = the j -th cluster in the predicted clustering.

NMI Formula

Joint Probability: $P(i, j) = \frac{|U_i \cap V_j|}{N}$ (fraction of points in both true cluster i and predicted cluster j).

Marginal Probabilities: $P(i) = \frac{|U_i|}{N}$, $P(j) = \frac{|V_j|}{N}$.

Mutual Information:

$$MI(U, V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} P(i, j) \log \left(\frac{P(i, j)}{P(i)P(j)} \right)$$

Normalized Version:

$$NMI(U, V) = \frac{2 \times MI(U, V)}{H(U) + H(V)}$$

Entropy: $H(U) = - \sum_{i=1}^{|U|} P(i) \log P(i)$.

- ▶ **Range:** $[0, 1]$, where 1 = perfect agreement, 0 = independent.
- ▶ **Advantage:** Information-theoretic, symmetric measure.

Silhouette Score

For each point i :

- ▶ $a(i)$ = average distance to points in same cluster.
- ▶ $b(i)$ = average distance to points in nearest different cluster.

Silhouette coefficient:

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

Overall Silhouette Score:

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

Silhouette Score

Detailed Formulation:

- ▶ $a(i) = \frac{1}{|C_I|-1} \sum_{j \in C_I, j \neq i} d(i, j)$ where C_I is cluster containing point i .
- ▶ $b(i) = \min_{k \neq I} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j)$ where C_k are other clusters.

Interpretation:

- ▶ $s(i) \approx 1$: point is well-clustered (far from other clusters).
- ▶ $s(i) \approx 0$: point is on cluster boundary.
- ▶ $s(i) \approx -1$: point is mis-clustered (closer to other clusters).

Comparison Summary

Algorithm	Type	Shapes	Parameters	Noise	Scale
K-Means	Centroid	Spherical	k	No	High
DBSCAN	Density	Arbitrary	ϵ , MinPts	Yes	Medium
Agglomerative	Hierarchical	Arbitrary	k , linkage	No	Low
GMM	Probabilistic	Elliptical	k , covariance	No	Medium
Spectral	Graph-based	Complex	k , similarity	No	Low

Distance Metrics Used:

- ▶ **K-Means:** Euclidean (typically).
- ▶ **DBSCAN:** Any metric (cosine, Euclidean, etc.).
- ▶ **Agglomerative:** Euclidean, Manhattan, cosine.
- ▶ **GMM:** Implicitly Euclidean (in feature space).
- ▶ **Spectral:** Via similarity graph construction.

Choosing the Right Algorithm

Decision Framework: see `code1_clustering_comparison.py`

Know number of clusters?

- ▶ **Yes:** K-Means, GMM, Agglomerative, Spectral.
- ▶ **No:** DBSCAN, Agglomerative (with dendrogram analysis).

Cluster shapes expected?

- ▶ **Spherical:** K-Means.
- ▶ **Elliptical:** GMM.
- ▶ **Arbitrary:** DBSCAN, Spectral.
- ▶ **Any:** Agglomerative.

Dataset size?

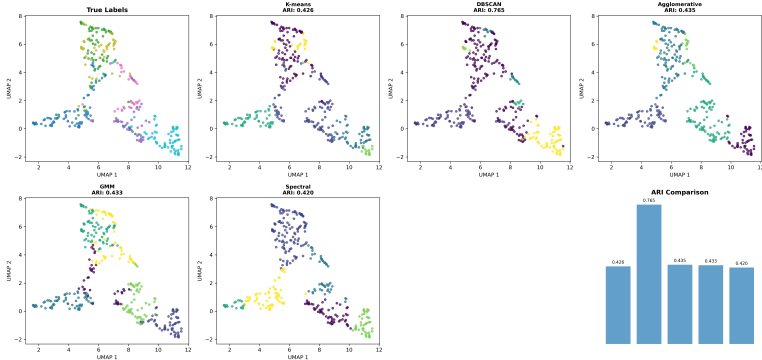
- ▶ **Large** ($> 10k$): K-Means, DBSCAN.
- ▶ **Medium** ($1k - 10k$): All methods.
- ▶ **Small** ($< 1k$): All methods.

Need outlier detection?

- ▶ **Yes:** DBSCAN.
- ▶ **No:** Other methods.

Clustering comparison results

Clustering Algorithm Comparison on DINO Features (Euclidean Distance, except DBSCAN & GMM with Cosine)



Practical Tips

Data Preprocessing:

- ▶ **Standardization:** Important for distance-based methods.
- ▶ **Normalization:** Consider for high-dimensional data.
- ▶ **Feature Selection:** Remove irrelevant features.
- ▶ **Dimensionality Reduction:** PCA/UMAP before clustering.

Practical Tips

Data Preprocessing:

- ▶ **Standardization**: Important for distance-based methods.
- ▶ **Normalization**: Consider for high-dimensional data.
- ▶ **Feature Selection**: Remove irrelevant features.
- ▶ **Dimensionality Reduction**: PCA/UMAP before clustering.

Parameter Tuning:

- ▶ **K-Means**: Try multiple k values, use elbow method.
- ▶ **DBSCAN**: Use k-distance plots for ϵ selection.
- ▶ **Agglomerative**: Analyze dendrogram for optimal cuts.
- ▶ **GMM**: Cross-validation for component selection.
- ▶ **Spectral**: Experiment with similarity graph construction.

Practical Tips

Data Preprocessing:

- ▶ **Standardization:** Important for distance-based methods.
- ▶ **Normalization:** Consider for high-dimensional data.
- ▶ **Feature Selection:** Remove irrelevant features.
- ▶ **Dimensionality Reduction:** PCA/UMAP before clustering.

Parameter Tuning:

- ▶ **K-Means:** Try multiple k values, use elbow method.
- ▶ **DBSCAN:** Use k-distance plots for ε selection.
- ▶ **Agglomerative:** Analyze dendrogram for optimal cuts.
- ▶ **GMM:** Cross-validation for component selection.
- ▶ **Spectral:** Experiment with similarity graph construction.

Best Practices:

- ▶ Try multiple algorithms.
- ▶ Validate results thoroughly.
- ▶ Consider ensemble methods.
- ▶ Document parameter choices.
- ▶ Visualize results when possible.