Lecture 8: Clustering Techniques

Luu Minh Sao Khue

Distance- and density-based clustering — K-Means, K-Medoids, Hierarchical, and DBSCAN. Cluster validity metrics (Silhouette, Davies–Bouldin). Apply clustering on real data and visualize clusters.

1. Learning Objectives

After this lecture, you will be able to:

- Explain the concept and intuition of clustering as an unsupervised task.
- Formulate distance-based clustering algorithms mathematically.
- Implement K-Means, K-Medoids, Hierarchical, and DBSCAN clustering.
- Evaluate clusters using internal metrics (Silhouette, Davies–Bouldin).
- Compare clustering methods and visualize clusters in Python.

2. Intuition

Clustering groups similar observations together without labels. It discovers patterns, segments customers, or groups medical profiles by similarity. The idea: objects within a cluster are more similar to each other than to objects in other clusters.

Each algorithm defines similarity differently:

- **K-Means** — minimizes Euclidean distance to cluster centers (means).
- **K-Medoids** — minimizes distance to real representative points (medoids).
- **Hierarchical** — builds tree-like structure of clusters.
- **DBSCAN** — groups dense regions, ignoring noise.

3. Distance Metrics

The most common measure of dissimilarity between two points $x_i = (x_{i1}, \dots, x_{ip})$ and $x_j = (x_{j1}, \dots, x_{jp})$ is:

$$d(x_i, x_j) = \begin{cases} \sqrt{\sum_k (x_{ik} - x_{jk})^2}, & \text{Euclidean distance,} \\ \sum_k |x_{ik} - x_{jk}|, & \text{Manhattan distance.} \end{cases}$$

Other metrics include cosine distance for text and correlation distance for time series.

4. K-Means Clustering

4.1 Objective Function

Given n samples $\{x_1, \ldots, x_n\}$, K-Means partitions them into K clusters $C_1, \ldots, C_K$ minimizing within-cluster variance:

$$J = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2,$$

where μ_k is the mean (centroid) of cluster C_k:

$$\mu_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i.$$

4.2 Algorithm

1. Initialize K centroids (randomly or by K-Means++).
2. Assign each sample to its nearest centroid.
3. Recompute centroids as cluster means.
4. Repeat until assignments stop changing or convergence threshold met.

4.3 Complexity and Limitations

- Complexity: $O(nKtp)$ where t is iterations, p features.
- Sensitive to initialization and scale.
- Works best for spherical clusters with similar variance.

5. K-Medoids Clustering (PAM)

K-Medoids is similar to K-Means but uses *actual data points* (medoids) instead of mean vectors. The objective:

$$J = \sum_{k=1}^{K} \sum_{x_i \in C_k} d(x_i, m_k),$$

where m_k is the medoid minimizing the total distance to other points in cluster C_k.

5.1 Advantages

- Robust to outliers (since medoids are real observations).
- Works with arbitrary distance metrics.

5.2 Drawbacks

- Computationally more expensive ($O(K(n-K)^2)$).
- Harder to scale to large datasets.

6. Hierarchical Clustering

6.1 Concept

Builds a tree (dendrogram) representing nested clusters. Two approaches:

- **Agglomerative:** Start with each point as a cluster and merge iteratively.
- **Divisive:** Start with one cluster and split iteratively.

6.2 Linkage Criteria

Distance between clusters A and B:

$$d(A, B) = \begin{cases} \min_{i \in A, j \in B} d(x_i, x_j), & \text{Single linkage,} \\ \max_{i \in A, j \in B} d(x_i, x_j), & \text{Complete linkage,} \\ \frac{1}{|A||B|} \sum_{i \in A} \sum_{j \in B} d(x_i, x_j), & \text{Average linkage.} \end{cases}$$

6.3 Properties

- No need to specify K in advance (can cut tree at any level).
- Sensitive to distance metric and linkage choice.
- Complexity: $O(n^2 \log n)$ (memory-intensive for large n).

7. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

DBSCAN defines clusters as high-density regions separated by low-density regions.

7.1 Parameters

- ε: radius of neighborhood.
- *minPts*: minimum number of points in a dense region.

7.2 Definitions

- A point x_i is a **core point** if it has at least *minPts* neighbors within ε.
- A point x_j is **directly density-reachable** from x_i if x_j lies within ε of a core point.
- **Clusters** are maximal sets of density-connected points.

7.3 Advantages and Limitations

- Finds arbitrarily shaped clusters.
- Robust to noise and outliers.
- Fails when densities vary strongly across clusters.

8. Cluster Validity Metrics

8.1 Silhouette Coefficient

For each sample i:

$$a_i = \text{mean intra-cluster distance}, \qquad b_i = \text{mean nearest-cluster distance.}$$

Silhouette score:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)} \in [-1, 1].$$

Overall mean silhouette close to 1 means well-separated clusters.

8.2 Davies–Bouldin Index (DBI)

$$\text{DBI} = \frac{1}{K} \sum_{i=1}^{K} \max_{j \neq i} \frac{\sigma_i + \sigma_j}{d(\mu_i, \mu_j)},$$

where σ_i is the average distance of points in cluster i to its centroid. Lower DBI is better.

8.3 Other Metrics

- Calinski–Harabasz index (higher better).
- Dunn index (higher better).

9. Comparison of Clustering Methods

Algorithm	Assumes Shape	Handles Noise	Scalability	Need K?
K-Means	Spherical	No	High	Yes
K-Medoids	Arbitrary	Moderate	Low	Yes
Hierarchical	Arbitrary	No	Medium	No
DBSCAN	Arbitrary	Yes	Medium	No

When to use what:

- Use **K-Means** for large, spherical clusters.
- Use **K-Medoids** for robust clustering with outliers.
- Use **Hierarchical** for dendrogram analysis or small datasets.
- Use **DBSCAN** for noisy data or irregular cluster shapes.

10. Python Example: Customer Segmentation

```python
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from sklearn.metrics import silhouette_score, davies_bouldin_score
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs

# ---- Generate sample data ----
X, _ = make_blobs(n_samples=500, centers=4, cluster_std=0.6, random_state=42)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# ---- K-Means ----
kmeans = KMeans(n_clusters=4, random_state=42).fit(X_scaled)
km_labels = kmeans.labels_
km_sil = silhouette_score(X_scaled, km_labels)
km_dbi = davies_bouldin_score(X_scaled, km_labels)

# ---- Hierarchical ----
hier = AgglomerativeClustering(n_clusters=4, linkage='ward').fit(X_scaled)
h_labels = hier.labels_
h_sil = silhouette_score(X_scaled, h_labels)
h_dbi = davies_bouldin_score(X_scaled, h_labels)

# ---- DBSCAN ----
db = DBSCAN(eps=0.8, min_samples=5).fit(X_scaled)
db_labels = db.labels_
mask = db_labels != -1   # ignore noise
db_sil = silhouette_score(X_scaled[mask], db_labels[mask])
db_dbi = davies_bouldin_score(X_scaled[mask], db_labels[mask])

# ---- Compare ----
print(f"K-Means: Silhouette={km_sil:.3f}, DBI={km_dbi:.3f}")
print(f"Hierarchical: Silhouette={h_sil:.3f}, DBI={h_dbi:.3f}")
print(f"DBSCAN: Silhouette={db_sil:.3f}, DBI={db_dbi:.3f}")

# ---- Visualization ----
fig, axs = plt.subplots(1, 3, figsize=(12, 4))
for ax, labels, title in zip(
    axs, [km_labels, h_labels, db_labels],
    ['K-Means', 'Hierarchical', 'DBSCAN']
):
    scatter = ax.scatter(X_scaled[:,0], X_scaled[:,1], c=labels, cmap='tab10')
    ax.set_title(title)
plt.tight_layout()
plt.show()
```

11. Practical Tips

- Always scale features before distance-based clustering.
- Use the elbow method or silhouette to choose K in K-Means.
- Remove or normalize outliers before K-Means.
- For DBSCAN, tune ε via k-distance plot.
- Visualize results (2D projection via PCA if data are high-dimensional).

12. Summary

We learned how clustering groups unlabeled data by similarity.

- **K-Means:** minimizes within-cluster variance; fast but sensitive to outliers.
- **K-Medoids:** uses medoids for robustness.
- **Hierarchical:** builds nested clusters, visualized by dendrograms.
- **DBSCAN:** density-based, detects noise and arbitrary shapes.
- **Validation:** Silhouette (higher better), Davies–Bouldin (lower better).
- **Comparison:** Choose based on shape, noise, and scalability.

13. Exercises

1. Derive the K-Means objective and explain convergence condition.
2. Implement K-Medoids using pairwise distance matrix.
3. Apply hierarchical clustering on a real dataset (e.g., Iris or customer spending data) and plot dendrogram.
4. Experiment with DBSCAN by varying ε and minPts.
5. Compute silhouette and Davies–Bouldin scores for all methods and compare.
6. Explain when DBSCAN outperforms K-Means conceptually.