A Comprehensive Course on Clustering Methods

Jawad ALAOUI M'HAMMEDI

November 2024

Contents

1 Introduction to Clustering

Clustering is a fundamental unsupervised learning technique. Unlike supervised learning methods such as classification or regression that rely on labeled examples, clustering aims to discover inherent groupings or structure in unlabeled data. Given a dataset $\{x_1, x_2, \ldots, x_n\}$ where each $x_i \in \mathbb{R}^p$ represents an observation, clustering algorithms attempt to partition these observations into K groups (clusters) such that:

- Observations within a cluster are more similar to each other than to observations in other clusters.
- The similarity or dissimilarity is often quantified by a distance measure $d(x_i, x_j)$, commonly Euclidean distance.

Clustering serves multiple purposes:

- **Understanding Data Structure:** By grouping similar items together, we can reveal underlying patterns or structures that may be hard to identify manually.
- **Preprocessing Step:** Clustering can be used to compress high-volume data or to reduce dimensionality. For example, applying K-means to a massive dataset can produce a smaller, representative set of cluster centroids for subsequent machine learning tasks.
- **Market Segmentation and Other Applications:** In marketing, clustering can identify different segments of customers. In bioinformatics, it can cluster genes or proteins to discover functional categories.
- **Pattern Discovery:** In large text corpora, clustering can identify recurring topics; in image processing, it can help discover visually similar patterns or compress images.

Compared to other approaches:

- **Vs. Classification:** Classification requires known labels. Clustering does not; it tries to discover structure from unlabeled data.
- **Vs. Regression:** Regression predicts a continuous outcome, while clustering focuses on grouping data without a specific response variable.

2 Overview of Common Clustering Methods

Clustering is not a single method but rather a set of diverse approaches. We will cover:

- **Hierarchical Clustering (H-Clustering):** Builds a nested hierarchy of clusters, typically represented by a dendrogram.
- **K-means and K-medoids:** Partition the data into K groups, often optimizing a criterion related to intra-cluster variance.
- **Gaussian Mixture Models with Expectation-Maximization (EM):** Model data as a mixture of Gaussian distributions and use EM to estimate parameters.
- **Density-Based Clustering (DBSCAN):** Discovers clusters based on regions of high density separated by low-density areas, automatically detecting noise and outliers.
- **OPTICS:** A density-based method related to DBSCAN that handles varying densities and provides a reachability plot to visualize cluster structure.

Each method has different assumptions, computational costs, and suitability for various data types and problem settings.

3 Hierarchical Clustering (H-Clustering)

Hierarchical clustering produces a hierarchy of clusters. In the **agglomerative** approach, we start with each observation in its own singleton cluster and repeatedly merge the two most similar clusters until all points belong to a single cluster. The process results in a tree-like structure called a *dendrogram*, which can be cut at various levels to obtain different cluster solutions. In the **divisive** approach, we start with all points in one cluster and recursively split clusters.

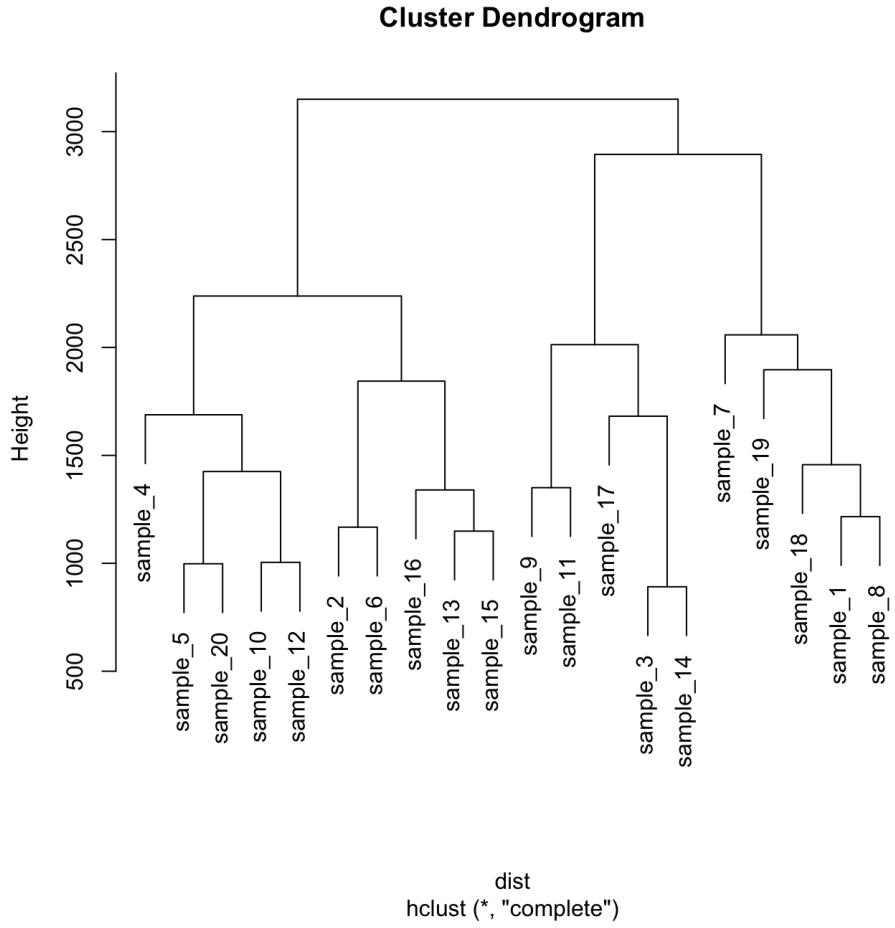


Figure 1: An example of a dendrogram

3.1 Distance Measures and Linkage Criteria

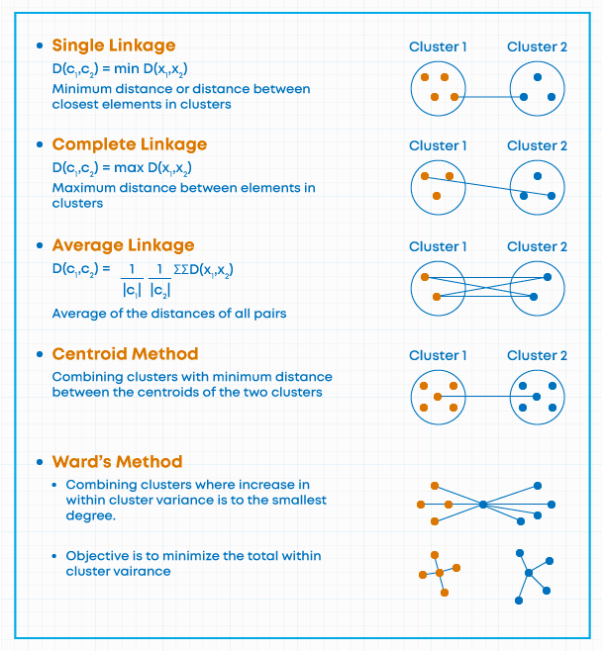
For hierarchical clustering, we must define:

- A **distance measure** between individual points, commonly Euclidean distance.
- A **linkage criterion** to define the distance between clusters.

Common linkage methods include:

Single linkage $d(C_a, C_b) = \min_{x_i \in C_a, x_j \in C_b} d(x_i, x_j)$. This can produce "stringy" clusters and is sensitive to noise, as it chains points together.

Complete linkage $d(C_a, C_b) = \max_{x_i \in C_a, x_j \in C_b} d(x_i, x_j)$. This encourages more compact clusters, less susceptible to chaining.



Average linkage $d(C_a, C_b) = \frac{1}{|C_a||C_b|}\sum_{x_i \in C_a}\sum_{x_j \in C_b} d(x_i, x_j)$. Produces intermediate results, balancing single and complete linkage tendencies.

Ward's method Minimizes the increase in within-cluster variance when merging. It's popular for producing compact and spherical clusters, especially when using Euclidean distances.

3.2 Complexity and Practical Use

Hierarchical clustering typically requires $O(n^2)$ distance computations and can be computationally expensive for very large datasets. For extremely large data, a common practice is to first perform K-means to reduce the number of points and then apply hierarchical clustering to the resulting centroids.

3.3 High-Level Remarks

- Dendrograms are a powerful visualization tool: by examining the vertical axis, you can choose a partition.
- Scaling variables may be necessary if they are on different units or scales.
- Hierarchical clustering doesn't require specifying the number of clusters upfront; you choose it after examining the dendrogram.

4 K-means and K-medoids Clustering

4.1 K-means Clustering

K-means seeks K clusters that minimize the within-cluster sum of squares:

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

where μ_k is the centroid of cluster k. The algorithm is:

1. Initialize K cluster centers (often chosen randomly).
2. Assign each point to the cluster with the nearest center.
3. Update each cluster's center as the mean of its assigned points.
4. Repeat assignments and updates until convergence.

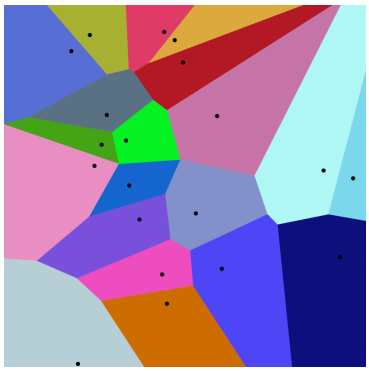


Figure 2: Voronoi diagram illustrating cluster assignment regions in K-means clustering.

Complexity: Typically $O(t \cdot K \cdot n)$ where t is the number of iterations, making it efficient for large n. **Drawbacks:**

- Sensitive to outliers because means shift significantly if extreme values are present.
- Only finds convex (often spherical) clusters.
- Requires specifying K in advance.
- Can get stuck in local minima; often run multiple times with different random starts or use k-means++ initialization to improve results.

4.2 K-medoids (PAM)

K-medoids is a related approach that picks actual data points as cluster centers (medoids) instead of means. This makes the method more robust to outliers because medoids must be actual points. The Partitioning Around Medoids (PAM) algorithm:

1. Initialize K medoids (data points).
2. Assign each point to the closest medoid.
3. For each medoid and each non-medoid point, try swapping them and keep the swap if it reduces the total cost.
4. Repeat until no improvement.

This method is often more computationally intensive than K-means but handles arbitrary distance metrics and outliers better.

4.3 Choosing K

Determining K is often done via:

- **Elbow method:** Plot within-cluster sum of squares against K and look for a bend.
- **Silhouette score:** Measures how similar an object is to its own cluster compared to other clusters.
- **Gap statistic:** Compares the within-cluster dispersion with that expected under a reference null distribution.
- **Information criteria (BIC, AIC):** For model-based clustering (e.g. Gaussian mixtures).

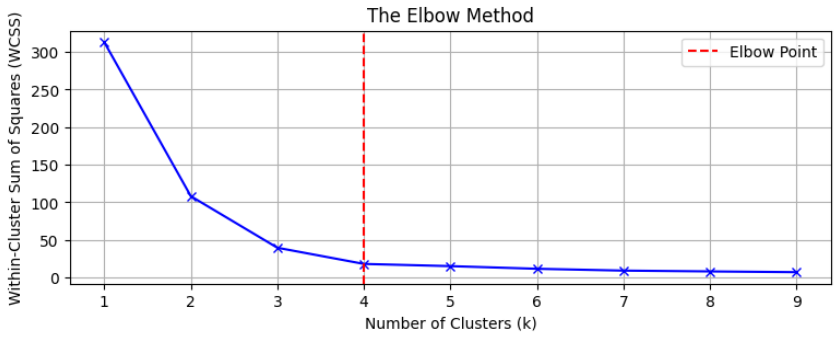


Figure 3: An example of the "Elbow" method for choosing K in K-means. The chosen K is often at the "bend" in the curve.

5 Gaussian Mixture Models with Expectation-Maximization

Instead of deterministic boundaries as in K-means, Gaussian Mixture Models (GMM) assume data is generated by a mixture of K Gaussian distributions:

$$p(x) = \sum_{j=1}^{K} \tau_j \mathcal{N}(x; \mu_j, \Sigma_j)$$

where τ_j are mixture weights, and μ_j, Σ_j are parameters of each Gaussian.

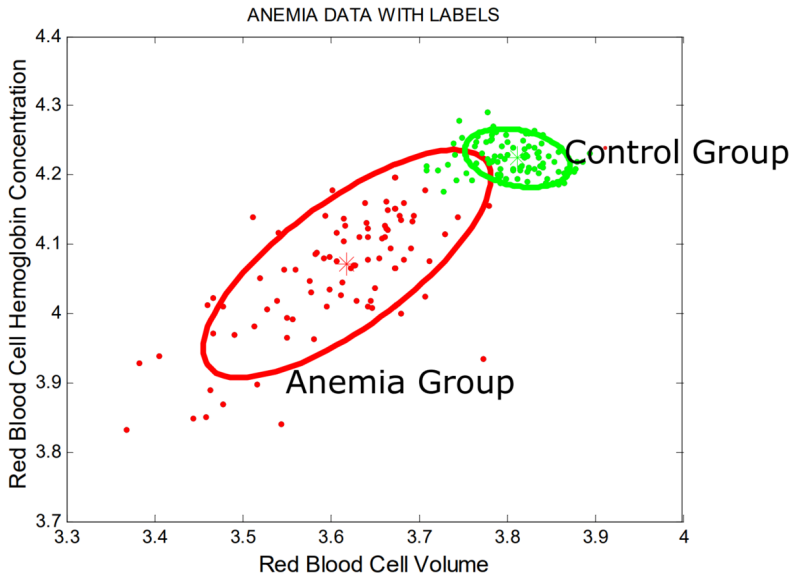


Figure 4: A Gaussian Mixture Model representation with two components in one dimension. Each Gaussian is shown along with the mixture distribution.

5.1 The EM Algorithm

The Expectation-Maximization algorithm finds parameters that maximize the data likelihood:

1. **E-step (Expectation Step):**

   Let $X_i \in \mathbb{R}^p$ be the ith data point for $i = 1, \ldots, n$, and suppose our mixture model consists of K Gaussian components. Each component $j \in \{1, \ldots, K\}$ is characterized by a mean vector $\mu_j \in \mathbb{R}^p$, a covariance matrix $\Sigma_j \in \mathbb{R}^{p \times p}$, and a mixing weight τ_j such that $\sum_{j=1}^{K} \tau_j = 1$. The probability density of component j is given by $f(x; \mu_j, \Sigma_j)$, the multivariate normal density.

   The E-step uses the current parameter estimates $\theta^{(t)} = \{\tau_j^{(t)}, \mu_j^{(t)}, \Sigma_j^{(t)}\}_{j=1}^{K}$ to compute the *posterior probabilities* (also called *responsibilities*) that each data point x_i was generated by each component j. These responsibilities $T_{j,i}^{(t)}$ represent a soft assignment of points to clusters, allowing partial membership rather than a strict one-to-one assignment.

   Formally, we compute:

$$T_{j,i}^{(t)} = P(Z_i = j \mid X_i = x_i; \theta^{(t)}) = \frac{\tau_j^{(t)} f(x_i; \mu_j^{(t)}, \Sigma_j^{(t)})}{\sum_{l=1}^{K} \tau_l^{(t)} f(x_i; \mu_l^{(t)}, \Sigma_l^{(t)})},$$

   where Z_i is a latent variable indicating the component from which x_i originates. Intuitively, $T_{j,i}^{(t)}$ measures how strongly point x_i is associated with component j under the current parameter estimates. The numerator is the likelihood of x_i coming from component j, weighted by $\tau_j^{(t)}$, and the denominator ensures the probabilities sum to one across all j.

   After the E-step, we have for each i and j a probability $T_{j,i}^{(t)}$ that x_i belongs to cluster j. This gives a soft clustering: each point is partially "owned" by every cluster according to these probabilities.

2. **M-step (Maximization Step):**

In the M-step, we use the responsibilities computed in the E-step as weights to re-estimate the parameters $(\tau_j, \mu_j, \Sigma_j)$ so as to maximize the expected complete-data log-likelihood. Essentially, we pretend each point x_i is fractionally assigned to component j with weight $T_{j,i}^{(t)}$, and then update parameters accordingly.

The updated parameters are computed as:

$$\tau_j^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n} T_{j,i}^{(t)}, \quad \mu_j^{(t+1)} = \frac{\sum_{i=1}^{n} T_{j,i}^{(t)} x_i}{\sum_{i=1}^{n} T_{j,i}^{(t)}}, \quad \Sigma_j^{(t+1)} = \frac{\sum_{i=1}^{n} T_{j,i}^{(t)} (x_i - \mu_j^{(t+1)})(x_i - \mu_j^{(t+1)})^\top}{\sum_{i=1}^{n} T_{j,i}^{(t)}}.$$

Here, $\tau_j^{(t+1)}$ is the new mixing weight for component j, computed as the average responsibility for that component across all points. The mean $\mu_j^{(t+1)}$ is a weighted average of the data points, with weights given by the responsibilities, and $\Sigma_j^{(t+1)}$ is the weighted covariance matrix of the points that component j is responsible for.

Thus, the M-step updates the mixture parameters to better fit the data given the current soft assignments. By alternating between the E-step and M-step, the EM algorithm iteratively improves the parameter estimates, leading to a locally optimal solution for the Gaussian mixture model.

Iterate until convergence. The EM algorithm provides a soft clustering: each point has a probability of belonging to each cluster.

5.2 Selecting the Number of Components

Use criteria like BIC or AIC since GMM is a probabilistic model. A higher BIC/AIC suggests less support for that model. Choose the K that minimizes these criteria.

Advantages: More flexible cluster shapes, probabilistic interpretation.

Drawbacks: Can converge to local maxima, requires careful initialization, and must choose K.

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

DBSCAN is a density-based clustering algorithm that discovers clusters of arbitrary shape based on local density criteria rather than distance from a global centroid. Instead of forcing a pre-specified number of clusters, DBSCAN identifies regions of high density that are separated by regions of low density.

The algorithm requires two main parameters: ε (epsilon) and `minPts`:

- ε defines the maximum radius of the neighborhood around a point.
- `minPts` specifies the minimum number of points required to form a dense region.

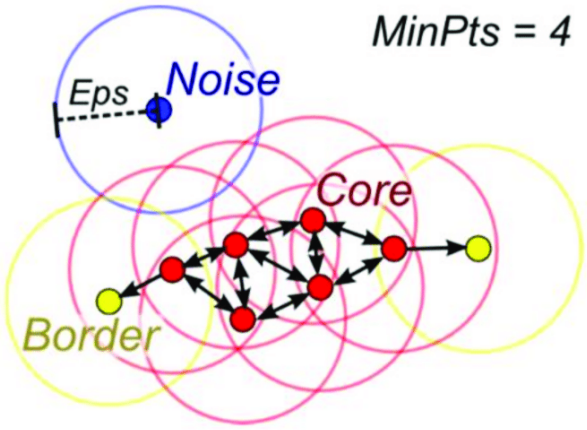


Figure 5: An illustration of DBSCAN identifying clusters (in color) and noise points (marked as outliers).

Key Concepts:

Core points: A point p is a *core point* if at least `minPts` points (including p) lie within distance ε of it. This set of points within ε of p is often called the ε-neighborhood of p. Core points are central to dense regions and are crucial to forming clusters.

Directly reachable: A point q is directly reachable from p if p is a core point and q lies in the ε-neighborhood of p. Note that direct reachability can only be established via a core point.

Reachable: A point q is reachable from p if there exists a sequence of points $p_1, p_2, \ldots, p_n$ such that $p_1 = p$, $p_n = q$, and each p_{i+1} is directly reachable from p_i. This implies that to travel from p to q within the cluster structure, all intermediate points on the path (except possibly q) must be core points. Thus, reachability extends the notion of connectivity through dense regions.

Outliers (Noise points): Points that are not reachable from any other point (under the given ε and `minPts`) are considered outliers. These are points lying in low-density areas where no sufficient number of neighbors exist.

Cluster Formation: A cluster in DBSCAN is formed around a core point. Once a core point p is identified, all points reachable from p (including core points and non-core points) belong to the same cluster. Each cluster therefore contains at least one core point. Non-core points within the ε-neighborhood of core or other reachable points belong to the cluster but cannot expand the cluster further since they are not themselves core points. These non-core points form the *edge* of the cluster, making clusters potentially complex shapes.

Properties of DBSCAN Clusters:

1. All points within a cluster are *density-connected*, meaning each point can be reached from any other via a chain of core points.

2. If a point is density-reachable from a member of a cluster, it is part of that cluster.

Notably, DBSCAN is not entirely deterministic because the order in which points are processed can affect the assignment of border points (points reachable from multiple clusters). However, core points and the overall structure remain stable.

Distance and Parameter Choices: DBSCAN typically uses Euclidean distance, though other metrics are possible. Choosing ε and `minPts` is nontrivial:

- ε**:** One approach is to examine a k-distance plot (where $k =$ `minPts`) and look for a prominent "knee" or "elbow" in the plot. This indicates a natural distance threshold separating dense and sparse regions. Setting ε too small may result in many points not being clustered (noise), while setting it too large may merge distinct clusters into one.

- `minPts`**:** A guideline is to set `minPts` greater than the number of dimensions D of the data. As a rule of thumb, `minPts` $\geq D + 1$, and often a value like 5, 10, or larger is used. Larger values of `minPts` are more robust to noise but may require a suitable ε for identifying meaningful clusters.

Complexity and Characteristics: DBSCAN runs in approximately $O(n \log n)$ time if a spatial index (like a k-d tree) is used for neighborhood queries. Without such indices, it can be $O(n^2)$ for distance computations. DBSCAN excels at finding arbitrarily shaped clusters and is robust to outliers, but it struggles with varying density levels across the dataset. If clusters differ greatly in density, a single ε-`minPts` combination may not separate them well.

Key Benefits of DBSCAN:

- No need to specify the number of clusters K in advance.
- Can find clusters of any shape, including non-convex patterns.
- Identifies outliers naturally as points not assigned to any cluster.

Limitations:

- Sensitive to the selection of ε and `minPts`.
- Does not handle large variations in density well.

5.4 OPTICS (Ordering Points To Identify the Clustering Structure)

OPTICS is another density-based clustering algorithm designed to overcome some limitations of DBSCAN, especially in datasets with varying densities. Instead of producing a flat set of clusters for a fixed ε, OPTICS generates an augmented ordering of the database that captures the cluster structure at multiple density levels.

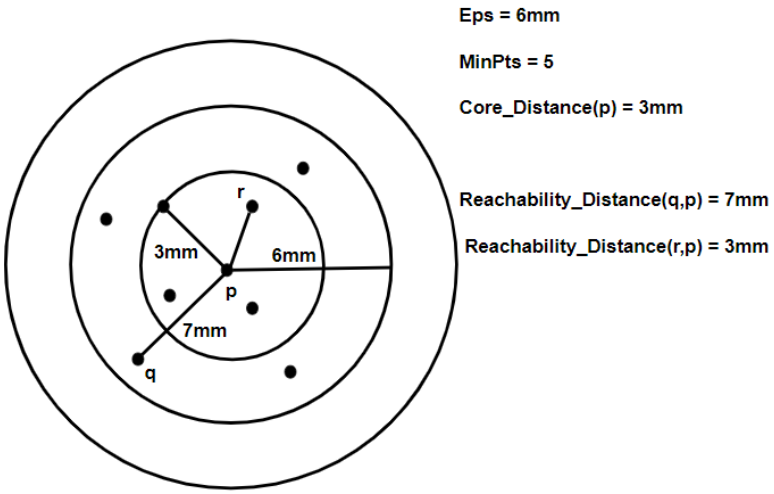


Figure 6: A reachability plot produced by OPTICS, showing cluster structures as "valleys" in the plot.

Core Ideas: OPTICS does not directly produce a clustering but yields an *ordering* of points and associated *reachability-distances*. From this ordering and these distances, one can extract cluster structures at different density thresholds.

Generating-distance (ϵ): OPTICS uses a parameter ϵ that serves as an upper bound on the neighborhood size, but does not strictly partition points based on a single ϵ. Instead, it ensures that the data is processed in a particular order.

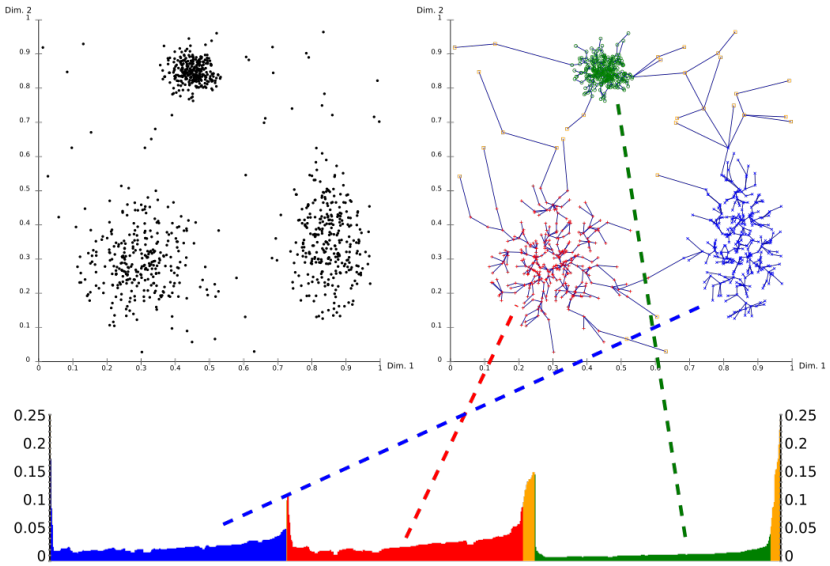
Core-distance: For a point P, the *core-distance* is defined as the minimum ϵ' such that P has at least `minPts` neighbors within ϵ'. If no such ϵ' exists (i.e., P is not a core point for any radius), then P's core-distance is undefined.

Reachability-distance: The *reachability-distance* of a point P (relative to another point O) measures how close P is to being density-reachable from O. Formally, the reachability-distance of P with respect to a core point O is the greater of O's core-distance and the actual distance between O and P. This ensures that if P is well within a dense neighborhood of O, it will have a small reachability-distance.

Algorithmic Process: OPTICS creates an ordering of points based on density connectivity. It starts from a point, finds its ϵ-neighborhood, identifies core points, and iteratively expands clusters by traversing a priority queue of points sorted by their reachability-distance. Points in denser regions (lower reachability-distances) are processed earlier, producing an ordering of points from "denser" to "sparser" regions.

Reachability Plot: A key output of OPTICS is the *reachability plot*, which is a two-dimensional plot:

- The x-axis is the order in which points are processed.
- The y-axis is the reachability-distance of each point.



Clusters appear as valleys in this plot: points in a cluster have low reachability-distances, forming a "dip" in the plot. By visually inspecting the reachability plot or by applying threshold-based extraction, multiple density-based clusterings can be obtained from a single OPTICS run.

Extracting Clusters: After producing the ordering and reachability-distances, one can:

- Select a reachability threshold to "cut" the plot horizontally, identifying clusters similar to DBSCAN at a particular density level.

- Use heuristic methods to detect "valleys" in the reachability plot automatically.
- Obtain a hierarchy of clusters at various density levels, providing richer structural information than a single run of DBSCAN.

Complexity and Considerations: With a suitable spatial index, OPTICS achieves $O(n \log n)$ complexity, similar to DBSCAN. Without such indexing, it may be $O(n^2)$. OPTICS often runs with a constant slowdown factor compared to DBSCAN. The main advantage of OPTICS is that it does not require a globally optimal ϵ, making it more flexible when the data density varies significantly.

Advantages of OPTICS:

- Handles datasets with varying densities without committing to a single ϵ.
- Produces a rich visualization (reachability plot) that allows hierarchical cluster structures to be discovered.

Limitations:

- More complex to interpret than DBSCAN, as one must analyze the reachability plot.
- Determining cluster boundaries from the plot can be subjective or require additional heuristics.

OPTICS can thus be seen as a generalization and improvement over DBSCAN when dealing with non-uniform densities, providing deeper insights into the data's density structure.

6 Python Example

Below is a Python example illustrating some clustering methods on synthetic data using scikit-learn. This code is just a basic demonstration; in practice, you would tune parameters and interpret results carefully.

```
import numpy as np
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans, DBSCAN
from sklearn.mixture import GaussianMixture
import matplotlib.pyplot as plt

# Generate synthetic dataset with 4 clusters
X, y_true = make_blobs(n_samples=300, centers=4, cluster_std=0.6,
    random_state=0)

# K-means
kmeans = KMeans(n_clusters=4, random_state=0).fit(X)
labels_km = kmeans.labels_

# Gaussian Mixture
gmm = GaussianMixture(n_components=4, random_state=0).fit(X)
labels_gmm = gmm.predict(X)

# DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5).fit(X)
labels_db = dbscan.labels_

fig, axes = plt.subplots(1,3, figsize=(12,4))
axes[0].scatter(X[:,0], X[:,1], c=labels_km, cmap='viridis')
```

```
axes[0].set_title("K-means")

axes[1].scatter(X[:,0], X[:,1], c=labels_gmm, cmap='viridis')
axes[1].set_title("Gaussian Mixture")

axes[2].scatter(X[:,0], X[:,1], c=labels_db, cmap='viridis')
axes[2].set_title("DBSCAN")

plt.show()
```

For hierarchical clustering or OPTICS, you would follow a similar pattern using `scipy.cluster.hierarchy` or `sklearn.cluster.OPTICS`.

7 Summary of Advantages and Drawbacks

Method	Advantages	Drawbacks
Hierarchical	No need to set K, visual dendrogram	$O(n^2)$, not scalable
K-means	Fast, simple	Need K, sensitive to outliers
K-medoids	More robust to outliers	More computationally expensive
Gaussian Mixture (EM)	Flexible cluster shapes, probabilistic	Need K, local maxima
DBSCAN	No need K, finds arbitrary shapes	Choosing ε hard, not for varying densities
OPTICS	Handles varying densities	Complex parameter tuning

Table 1: Advantages, drawbacks, and complexity of different clustering methods.

8 Practical Tips and Extensions

8.1 Scaling and Normalization

Data variables often have different scales. Applying standardization or normalization before clustering is common so that all features contribute equally to the distance metric.

8.2 High-Dimensional Data

In very high dimensions, distances become less meaningful due to the curse of dimensionality. Techniques like PCA or t-SNE can be applied first to reduce dimensionality, or specialized methods (like subspace clustering) can be considered.

8.3 Combining Clustering Methods

It is not uncommon to combine methods. For instance:

- Use K-means to compress a huge dataset into cluster centroids and then run hierarchical clustering on these centroids.
- Use hierarchical clustering to determine a suitable K for K-means.

8.4 Assessing Quality of Clusters

Since clustering is unsupervised, quality must be assessed by internal metrics (like silhouette scores, Calinski-Harabasz index) or external benchmarks (if some label or partial information is available).

9 Conclusion

Clustering provides a powerful, unsupervised approach to understanding data structure. Each method has its place:

- **Hierarchical clustering** is great for exploratory analysis and producing dendrograms.
- **K-means and K-medoids** offer simplicity and speed, suitable for large-scale problems where spherical clusters are acceptable.
- **Gaussian Mixture Models** allow more flexible cluster shapes and probabilistic interpretations.
- **DBSCAN and OPTICS** are strong choices for discovering arbitrarily shaped clusters and handling noise, with OPTICS offering insights into varying densities.

Selecting the best clustering method depends on the nature of the data, the presence of noise and outliers, computational constraints, and whether one seeks hierarchical structure, density-based patterns, or model-based clustering. In practice, experimentation, validation with domain knowledge, and multiple runs with different parameters are essential steps to achieve meaningful results.