Clustering

Jingyuan Sun
ChatGPT Claude

Dec, 2024

1 Introduction to Clustering

Clustering is a fundamental technique in unsupervised learning, aimed at grouping data points based on their similarity or dissimilarity. This section introduces the key concepts of clustering, starting with the distinction between supervised and unsupervised learning, followed by a detailed exploration of distance metrics.

1.1 Unsupervised vs. Supervised Learning

Definition 1 (Supervised Learning)**.** *Supervised learning is a type of machine learning where the algorithm is trained on labeled data. Given a dataset* $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$*, where* $\mathbf{x}_i \in \mathbb{R}^d$ *represents the features and* $y_i \in \mathcal{Y}$ *is the corresponding label, the objective is to learn a function* $f : \mathbb{R}^d \to \mathcal{Y}$ *that minimizes a loss function over the labeled dataset.*

Definition 2 (Unsupervised Learning)**.** *Unsupervised learning is a type of machine learning where the algorithm learns patterns, structures, or relationships from unlabeled data. Given a dataset* $\{\mathbf{x}_i\}_{i=1}^n$*, where* $\mathbf{x}_i \in \mathbb{R}^d$*, the goal is to discover underlying structures in the data, such as clusters or dimensionality reduction mappings.*

Key Differences:

- **Supervised Learning:** Requires labeled data and aims to predict outputs for unseen inputs.
- **Unsupervised Learning:** Operates on unlabeled data, focusing on pattern discovery.

Applications:

- **Unsupervised Learning:** Clustering, anomaly detection, dimensionality reduction.
- **Supervised Learning:** Classification, regression, and ranking tasks.

1.2 Distance Metrics

Distance metrics are essential in clustering algorithms, as they define the similarity or dissimilarity between data points. A distance function $d : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ must satisfy the following properties:

Definition 3 (Distance Metric)**.** *A function $d(\mathbf{x}, \mathbf{y})$ is a valid distance metric if:*

1. ***Non-negativity:*** $d(\mathbf{x}, \mathbf{y}) \geq 0, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$.
2. ***Identity of Indiscernibles:*** $d(\mathbf{x}, \mathbf{y}) = 0$ *if and only if* $\mathbf{x} = \mathbf{y}$.
3. ***Symmetry:*** $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$.
4. ***Triangle Inequality:*** $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$.

Common Distance Metrics:

- **Euclidean Distance:**

$$d_{\text{Euc}}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}.$$

  Measures the straight-line distance between two points in Euclidean space.

- **Cosine Similarity:**

$$\text{similarity}_{\cos}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}.$$

  The dissimilarity is defined as:

$$d_{\cos}(\mathbf{x}, \mathbf{y}) = 1 - \text{similarity}_{\cos}(\mathbf{x}, \mathbf{y}).$$

  Measures the angle between two vectors, commonly used in text analysis.

- **Mahalanobis Distance:**

$$d_{\text{Mah}}(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^\top \mathbf{S}^{-1} (\mathbf{x} - \mathbf{y})},$$

  where $\mathbf{S}$ is the covariance matrix of the dataset. This metric accounts for correlations between variables and is often used for identifying outliers.

Visual Examples: Figure 1 illustrates the application of these distance metrics in clustering tasks, highlighting their impact on cluster shapes and boundaries.

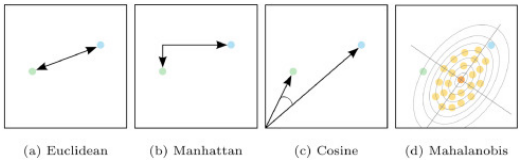

Figure 1: Examples of Euclidean, Manhattan, Cosine, and Mahalanobis distances applied to clustering.

2 Hierarchical Agglomerative Clustering (HAC)

2.1 Algorithm Description

Hierarchical Agglomerative Clustering (HAC) is an iterative clustering algorithm that builds a hierarchy of clusters. The steps of the algorithm are as follows:

1. Initialize each data point as a single-element cluster.
2. Compute the pairwise dissimilarity matrix C, where C_{ij} represents the dissimilarity between clusters i and j.
3. Repeat until only one cluster remains:
   (a) Find the two clusters A and B with the smallest dissimilarity, i.e.,
   $$(A, B) = \arg\min_{i,j} C_{ij}, \quad i \neq j.$$
   (b) Merge A and B into a single cluster.
   (c) Update the dissimilarity matrix C to reflect the dissimilarity between the new cluster and all remaining clusters based on a specified linkage method.

2.2 Linkages and Distance Measures

Linkages determine how dissimilarities between clusters are computed when merging. The following are common linkage methods:

2.2.1 Single Linkage

The dissimilarity between clusters A and B is defined as:

$$d(A, B) = \min_{x \in A, y \in B} d(x, y),$$

where $d(x, y)$ is the pairwise dissimilarity between points x and y.

Explanation: This method focuses on the closest pair of points from two clusters, resulting in elongated, chain-like clusters.

2.2.2 Complete Linkage

The dissimilarity between clusters A and B is defined as:

$$d(A, B) = \max_{x \in A, y \in B} d(x, y).$$

Explanation: This method considers the farthest pair of points between clusters, which tends to produce compact, spherical clusters.

2.2.3 Average Linkage

The dissimilarity between clusters A and B is defined as:

$$d(A, B) = \frac{1}{|A| \cdot |B|} \sum_{x \in A, y \in B} d(x, y),$$

where $|A|$ and $|B|$ are the sizes of clusters A and B, respectively.

Explanation: This method averages the distances between all pairs of points in the two clusters and often produces results between single and complete linkage.

2.2.4 Ward's Linkage

Ward's Linkage minimizes the increase in within-cluster variance after merging clusters A and B. The increase is given by:

$$\Delta(A, B) = \sum_{i \in A \cup B} \|x_i - c_{A \cup B}\|^2 - \sum_{i \in A} \|x_i - c_A\|^2 - \sum_{i \in B} \|x_i - c_B\|^2,$$

where c_A, c_B, and $c_{A \cup B}$ are the centroids of clusters A, B, and their union, respectively.

The change in variance can be rewritten as:

$$\Delta(A, B) = \frac{|A| \cdot |B|}{|A| + |B|} \|c_A - c_B\|^2.$$

Explanation: This method seeks to create compact, spherical clusters by minimizing the within-cluster variance.

2.2.5 Ward's Linkage Formula Derivation

The within-cluster variance for a cluster C is defined as:

$$\text{Var}(C) = \sum_{i \in C} \|x_i - c_C\|^2,$$

where c_C is the centroid of C. When merging clusters A and B, the new centroid $c_{A \cup B}$ is:

$$c_{A \cup B} = \frac{|A| \cdot c_A + |B| \cdot c_B}{|A| + |B|}.$$

The increase in variance is derived as:

$$\Delta(A, B) = \frac{|A| \cdot |B|}{|A| + |B|} \|c_A - c_B\|^2.$$

Explanation: The formula ensures that the merging process results in clusters with minimal increases in variance, favoring compact and spherical clusters.

2.3 Dendrogram

A dendrogram is a tree structure representing the hierarchy of cluster merges. It is constructed as follows:

1. Start with each point as a single cluster.

2. At each iteration, merge the two closest clusters and record the dissimilarity at which the merge occurred.

3. The height of each merge in the dendrogram represents the dissimilarity between the merged clusters.

4. Cutting the dendrogram at a certain height gives the desired number of clusters.

2.4 Algorithm Complexity

Time Complexity The naive implementation of HAC has a time complexity of:

$$\Theta(N^3),$$

where N is the number of data points. This is because we scan the $N \times N$ distance matrix $N - 1$ times.

Optimized implementations using priority queues can reduce the time complexity to:

$$\Theta(N^2 \log N).$$

Space Complexity The algorithm requires storing the $N \times N$ distance matrix, resulting in a space complexity of:

$$\Theta(N^2).$$

3 K-Means Clustering

3.1 Introduction to K-Means

K-Means clustering is a partition-based clustering method that divides the dataset into k distinct non-overlapping subsets. The primary goal is to minimize the total within-cluster variance by iteratively updating cluster assignments and centroids.

3.1.1 Definition

Given a dataset $X = \{x_1, x_2, \ldots, x_n\}$ with n data points in a d-dimensional space, the K-Means algorithm aims to partition X into k clusters, $C = \{C_1, C_2, \ldots, C_k\}$, such that the within-cluster sum of squares (WCSS) is minimized:

$$\min_{C_1, C_2, \ldots, C_k} \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2,$$

where μ_i is the centroid of cluster C_i.

3.1.2 Algorithm Steps

1. **Initialization:** Randomly select k initial cluster centroids.

2. **Reassignment:** Assign each data point to the cluster with the nearest centroid based on Euclidean distance.

$$C_i = \{x_j : \|x_j - \mu_i\|^2 \leq \|x_j - \mu_l\|^2, \forall l \neq i\}.$$

3. **Re-centering:** Recalculate the centroid of each cluster as the mean of the data points in that cluster:
$$\mu_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j.$$

4. **Iteration:** Repeat steps 2 and 3 until convergence, defined as no significant change in centroids or a fixed number of iterations.

3.2 Within-Cluster Variation (WCV)

The quality of clustering is often measured by the within-cluster variation (WCV), which is defined for a cluster C_k as:

$$WCV(C_k) = \frac{1}{|C_k|} \sum_{i=1}^{|C_k|} \|x_i - \mu_k\|^2.$$

The goal is to minimize the total WCV across all clusters:

$$\min_{C_1, C_2, \ldots, C_k} \sum_{k=1}^{K} WCV(C_k).$$

However, K-Means typically finds local minima rather than the global minimum.

3.3 Choosing the Number of Clusters

The optimal number of clusters k can be chosen using the **elbow method**, which involves plotting the total WCV (inertia) against different values of k. The "elbow point" represents the optimal k, where the rate of decrease in WCV significantly slows down.

3.4 Silhouette of Clustering

The silhouette method evaluates clustering quality by measuring how similar each data point is to its own cluster compared to other clusters. The silhouette coefficient s_i for a point i is defined as:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)},$$

where:

- a_i: Average distance between i and other points in the same cluster.
- b_i: Minimum average distance from i to points in any other cluster.

The silhouette coefficient ranges from -1 to 1, with higher values indicating better clustering.

3.4.1 Silhouette Coefficient Derivation

- In-cluster distance:
$$a_i = \frac{1}{|C_A| - 1} \sum_{j \in C_A, i \neq j} d(i, j).$$

- Between-cluster distance:
$$b_i = \min_{B \neq A} \frac{1}{|C_B|} \sum_{j \in C_B} d(i, j).$$

3.5 Complexity of K-Means

- Time complexity: $O(Nkdi)$, where N is the number of data points, k is the number of clusters, d is the dimensionality, and i is the number of iterations.

- Space complexity: $O((N + k)d)$.

3.6 Limitations of K-Means

3.6.1 Dissimilarity

Cause: K-Means clustering is fundamentally designed for Euclidean space and uses Euclidean distance as the measure of dissimilarity. This assumption makes it less effective when the underlying dataset is better described using other distance metrics.

Situations:

- **Non-Euclidean Data:** For datasets where relationships between data points are non-linear or involve categorical data, Euclidean distance fails to capture meaningful dissimilarities.

- **High-Dimensional Data:** In high-dimensional spaces, distances between points tend to become similar (the curse of dimensionality), reducing the effectiveness of clustering.

Solutions:

- Use alternative clustering algorithms that support diverse dissimilarity measures, such as DBSCAN or spectral clustering.

- If K-Means is necessary, consider pre-processing the data using dimensionality reduction techniques (e.g., PCA) to focus on the most informative features.

3.6.2 Cluster Shapes

Cause: K-Means assumes clusters are spherical and isotropic (equal variance in all directions). This assumption breaks down for irregularly shaped clusters.

Situations:

- **Spread-Out Clusters:** When clusters are elongated or have non-linear shapes, K-Means will misclassify points that fall near the decision boundaries.

- **Nested Clusters:** For clusters with one cluster contained within another (e.g., concentric circles), K-Means fails to identify the nested structure.

Solutions:

- Use density-based algorithms like DBSCAN that can capture irregularly shaped clusters.
- Kernel K-Means is a modified version of K-Means that uses a kernel function to project data into a higher-dimensional space where clusters are linearly separable.

3.6.3 Misspecified k Value

Cause: K-Means requires the number of clusters k to be specified beforehand. A wrong choice of k can lead to overfitting or underfitting.

Situations:

- **Overfitting:** Choosing a value of k larger than the true number of clusters leads to overly fragmented clusters.
- **Underfitting:** Choosing a value of k smaller than the true number of clusters merges distinct clusters into one, losing valuable information.

Solutions:

- Use methods like the elbow method or silhouette analysis to estimate an appropriate value for k.
- Algorithms like Gaussian Mixture Models (GMM) use a probabilistic approach and can determine the number of clusters automatically through model selection criteria (e.g., AIC or BIC).

3.6.4 Cluster Boundaries

Cause: K-Means defines cluster boundaries as equidistant hyperplanes between centroids. This rigid boundary definition is problematic when clusters vary in size or density.

Situations:

- **Clusters with Different Sizes:** Large, dispersed clusters are not well-separated from smaller, compact clusters.
- **Clusters with Varying Densities:** Points in dense regions are more likely to be assigned correctly than points in sparse regions near decision boundaries.

Solutions:

- Switch to clustering algorithms that consider density (e.g., DBSCAN) or hierarchical structure (e.g., Agglomerative Clustering).
- Weight the clustering process to account for cluster size or density variations.

4 DBSCAN: Density-Based Clustering

4.1 Introduction to DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) identifies clusters based on regions densely populated with points, grouping them to form larger clusters. Unlike K-means or hierarchical clustering, DBSCAN can discover clusters of arbitrary shape, making it particularly effective on datasets with non-linear cluster structures.

Key features:

- Identifies dense regions in data space and joins them to form clusters.
- Labels points not belonging to any cluster as anomalies or noise.
- Effective for datasets with noise or irregular cluster shapes.

4.2 DBSCAN Parameters

DBSCAN requires two key parameters:

- ϵ (**Eps**): The maximum distance for points to be considered neighbors.
- **MinPts**: The minimum number of points required within an ϵ-neighborhood for a point to qualify as a *core point*.

Point Classification:

- *Core Points*: Points with at least **MinPts** neighbors within their ϵ-neighborhood.
- *Border Points*: Points within the ϵ-neighborhood of a core point but not meeting the **MinPts** criterion.
- *Noise Points*: Points neither core points nor border points.

4.3 DBSCAN Algorithm

The DBSCAN algorithm works as follows:

1. Arbitrarily select an unvisited point p.
2. Retrieve all points within the ϵ-neighborhood of p (called *regionQuery*).
3. If p is a core point:
    - Add all points in its ϵ-neighborhood to the same cluster as p.
    - Recursively apply *regionQuery* to all unvisited neighbors.
4. If p is not a core point, label it as noise (unless it belongs to an existing cluster as a border point).
5. Repeat for all unvisited points.

4.4 Anomaly Detection in DBSCAN

Points that remain unclustered after applying DBSCAN are considered anomalies or noise. These points do not belong to the ϵ-neighborhood of any core point or border point.

4.5 DBSCAN Algorithm Complexity

- **Naive Implementation:**
  - **Time Complexity:** $\mathcal{O}(N^2)$ due to exhaustive pairwise distance computations.
  - **Space Complexity:** $\mathcal{O}(N^2)$ for storing the distance matrix.
- **Optimized Implementation:**
  - **Time Complexity:** $\mathcal{O}(N \log N)$ with spatial indexing (e.g., R-trees).
  - **Space Complexity:** $\mathcal{O}(N)$ for storing spatial indices.

4.6 Density-Connected Points

- A point q is *density-reachable* from a point p if q lies within the ϵ-neighborhood of p, and p is a core point.
- A point q is *density-connected* to a point r if there exists a point p such that q and r are both density-reachable from p.

4.7 Parameter Selection for DBSCAN

4.7.1 Choosing MinPts

- Rule of thumb: $\mathbf{MinPts} = 5$ for 2-dimensional data.
- For higher dimensions, use $\mathbf{MinPts} = 1 + 2 \times$ (number of dimensions).

4.7.2 Choosing ϵ

- Plot distances to the kth nearest neighbor for all points (with $k = \mathbf{MinPts} - 1$).
- Identify the "elbow point" in the plot to select ϵ.

4.8 HDBSCAN: Hierarchical DBSCAN Algorithm

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) improves upon DBSCAN by constructing a hierarchical tree of clusters. The algorithm refines the clustering process by dynamically adjusting density thresholds, resulting in a more flexible clustering solution.

Algorithm Steps:

1. **Core Distance Calculation:**

- For each data point, compute its *core distance*, which is the distance to the kth nearest neighbor, where $k = \textbf{MinPts}$.
- The core distance defines the minimum density required for the point to act as a core point.

2. **Mutual Reachability Distance:**
   - For each pair of points p and q, compute the *mutual reachability distance*:

$$d_{\text{mutual}}(p, q) = \max\{\text{core\_dist}(p), \text{core\_dist}(q), d(p, q)\},$$

     where $d(p, q)$ is the Euclidean distance between p and q.
   - This ensures that points with higher densities are more likely to form clusters.
3. **Construct a Minimum Spanning Tree (MST):**
   - Create a graph where each point is a node, and edges between nodes are weighted by their mutual reachability distances.
   - Use a minimum spanning tree (MST) algorithm (e.g., Prim's or Kruskal's algorithm) to find the tree connecting all points with minimal total weight.
4. **Cluster Formation:**
   - Traverse the MST from the root, repeatedly removing edges with large weights to split the tree into smaller subtrees.
   - At each level, clusters are formed based on the density thresholds defined by the mutual reachability distances of removed edges.
5. **Hierarchical Clustering:**
   - Repeat the clustering process at multiple density levels, creating a hierarchy of clusters.
   - Points with lower densities are marked as noise at higher density thresholds.
6. **Optimal Cluster Selection:**
   - Evaluate clusters at different hierarchy levels using metrics such as the silhouette coefficient or the cluster stability score.
   - Select the level that best represents the underlying structure of the data.

Advantages of HDBSCAN:

- **Flexibility:** Automatically determines the optimal number of clusters.
- **Noise Handling:** Effectively separates noise from meaningful clusters.
- **Scalability:** Handles large datasets with varying densities.
- **Robustness:** Reduces reliance on strict parameter tuning.