Clustering

Hierarchical clustering

- Hierarchical clustering is a method of cluster analysis that builds a hierarchy of clusters.
- Hierarchical clustering is an unsupervised learning technique, meaning it doesn't require labeled data to train the model.
- It is useful when you want to explore different levels of similarity between data points without specifying the number of clusters in advance.

Hierarchical clustering - Types

1. **Agglomerative (Bottom-Up Approach)**
   - Starts with each data point as its own cluster.
   - Iteratively merges the closest clusters based on a similarity measure.
   - Stops when all points belong to a single cluster or a defined number of clusters is reached.
   - This clustering algorithm does not require us to prespecify the number of clusters.
   - Bottom-up algorithms treat each data as a singleton cluster at the outset and then successively agglomerate pairs of clusters until all clusters have been merged into a single cluster that contains all data.

Hierarchical clustering - Types

2. **Divisive (Top-Down Approach)**
   - Starts with all data points in one large cluster.
   - Iteratively splits the cluster into smaller clusters.
   - Stops when each data point is in its own cluster or a defined number of clusters is reached.
   - This algorithm also does not require to prespecify the number of clusters.
   - Top-down clustering requires a method for splitting a cluster that contains the whole data and proceeds by splitting clusters recursively until individual data have been split into singleton clusters.

Hierarchical clustering - Types

Divisive clustering Working:

- Treat the entire dataset as a single large cluster.
- **Split the cluster**: Divide the cluster into two smaller clusters. The division is done by finding the two most dissimilar points in the cluster and using them to separate the data into two parts.
- For each of the new clusters, repeat the splitting process:
  - Choose the cluster with the most dissimilar points.
  - Split it again into two smaller clusters.
- Continue this process until every data point is its own cluster, or the stopping condition (such as a predefined number of clusters) is met.

Hierarchical clustering

Distance Measures

To determine the similarity between clusters, different distance metrics can be used:

1. **Euclidean Distance**: Most commonly used, measures straight-line distance.
2. **Manhattan Distance**: Measures the sum of absolute differences.
3. **Cosine Similarity**: Measures the cosine of the angle between two vectors.

Hierarchical clustering (Computing Distance Matrix)

Linkage Methods

- **Single Linkage**: Distance between the closest points of two clusters. Find the minimum distance between any two points of the cluster.
- **Complete Linkage**: Distance between the farthest points of two clusters. Find the maximum distance between any two points of the cluster.
- **Average Linkage**: Mean distance between all points in both clusters. Find the average distance between every two points of the clusters.
- **Centroid Linkage**: Distance between the centroids of two clusters.
- **Ward's Method**: Minimizes the variance within clusters. The similarity of two clusters is based on the increase in sum of squared error when two clusters are merged.
  - smallest increase in the total within-cluster variance

Hierarchical clustering

Dendrogram

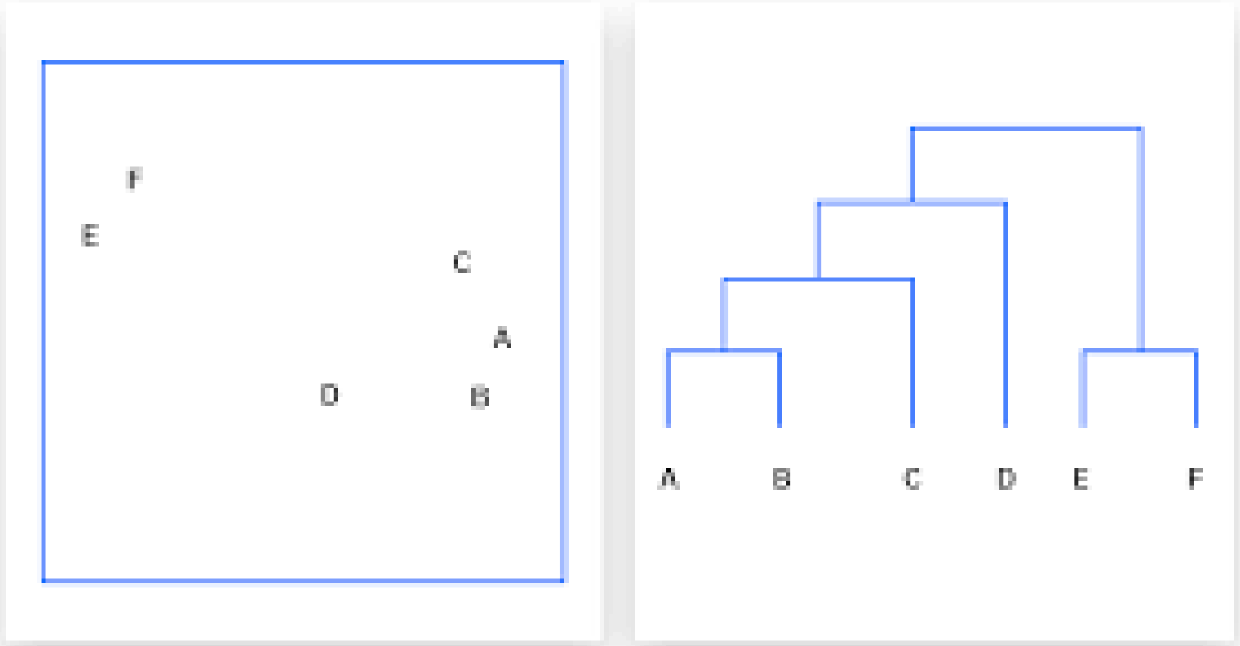
- A tree-like diagram that represents the merging process of clusters.
- Helps determine the optimal number of clusters by setting a cutoff threshold.
- shows the relationships between clusters at different levels of similarity.

How to Read a Dendrogram?

- The **x-axis** represents individual data points.
- The **y-axis** represents the distance (or dissimilarity) at which clusters are merged.
- The **height at which a horizontal line is cut** determines the number of clusters.

Hierarchical clustering

Dendrogram Example



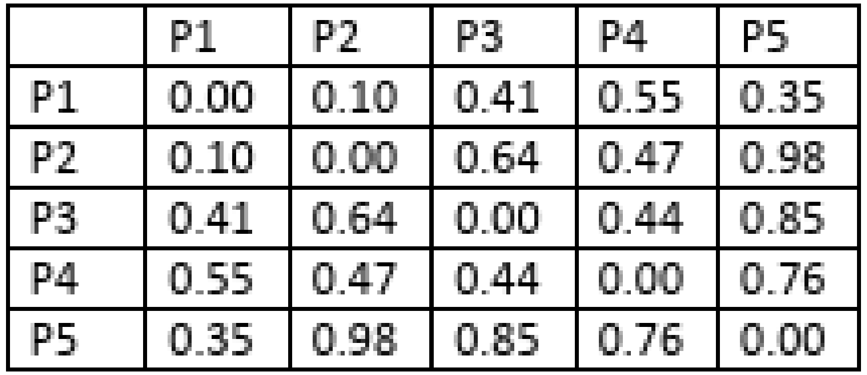
Hierarchical clustering working

- **Distance Calculation**: The algorithm first calculates the distance or dissimilarity between all data points (or clusters) using a chosen metric (e.g., Euclidean distance).
- **Merging/Splitting:**
  - Agglomerative: It then iteratively merges the closest clusters based on a linkage criterion (e.g., single linkage, complete linkage, average linkage).
  - Divisive: It iteratively splits the clusters until each data point is in its own cluster.
- Dendrogram Creation: The resulting hierarchy of clusters is represented as a dendrogram.

Hierarchical Clustering- Solved Example

- Use the distance matrix in Table1 to perform single link and complete link hierarchical clustering. Show your results by drawing a dendrogram. The dendrogram should clearly show the order in which the points are merged.

Table 1 Distance matrix

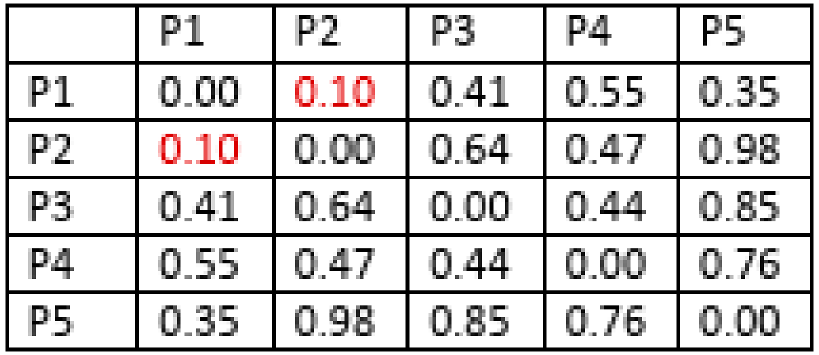


	P1	P2	P3	P4	P5
P1	0.00	0.10	0.41	0.55	0.35
P2	0.10	0.00	0.64	0.47	0.98
P3	0.41	0.64	0.00	0.44	0.85
P4	0.55	0.47	0.44	0.00	0.76
P5	0.35	0.98	0.85	0.76	0.00

Hierarchical Clustering- Solved Example

Solution: **single link**

- Using graph terminology, start with all points as singleton clusters.
- Add links between points one at a time (shortest links first).
- These single links combine the points into clusters.



	P1	P2	P3	P4	P5
P1	0.00	0.10	0.41	0.55	0.35
P2	0.10	0.00	0.64	0.47	0.98
P3	0.41	0.64	0.00	0.44	0.85
P4	0.55	0.47	0.44	0.00	0.76
P5	0.35	0.98	0.85	0.76	0.00

Combine P1 and P2:

- $\boldsymbol{dist(\{P1,P2\},\{P3\})} = min(dist(P1,P3), dist(P2,P3))$

  $= min(0.41, 0.64) = \mathbf{0.41}$

- $\boldsymbol{dist(\{P1,P2\},\{P4\})} = min(dist(P1,P4), dist(P2,P5))$

  $= min(0.55, 0.98) = \mathbf{0.55}$

- $\boldsymbol{dist(\{P1,P2\},\{P5\})} = min(dist(P1,P5), dist(P2,P5))$

  $= min(0.35, 0.98) = \mathbf{0.35}$

Hierarchical Clustering- Solved Example

	P12	P3	P4	P5
P12	0.00	0.41	0.55	0.35
P3	0.41	0.00	0.44	0.85
P4	0.55	0.44	0.00	0.76
P5	0.35	0.85	0.76	0.00

Combine P12 and P5:

- $dist(\{P12,P5\},\{P3\}) = min(dist(P12,P3),dist(P5,P3))$

  $=min(0.41,0.85) =\mathbf{0.41}$
- $dist(\{P12,P5\},\{P4\}) = min(dist(P12,P4),\ dist(P5,P4))$

  $=min(0.55,0.76) =\mathbf{0.55}$

	P125	P3	P4
P125	0.00	0.41	0.55
P3	0.41	0.00	0.44
P4	0.55	0.44	0.00

Combine P125 and P3:

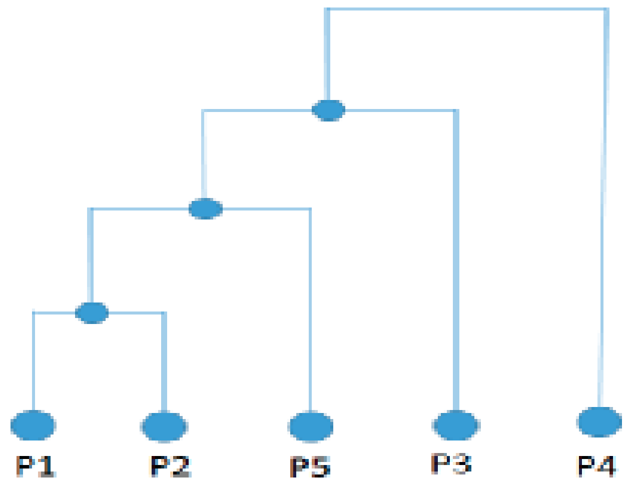
- $dist(\{P125,P3\},\{P4\}) = min(dist(P125,P4),dist(P3,P4))$

  $=min(0.55,0.44)$

  $=\mathbf{0.44}$

Hierarchical Clustering- Solved Example

	P1235	P4
P1235	0.00	0.44
P4	0.44	0.00

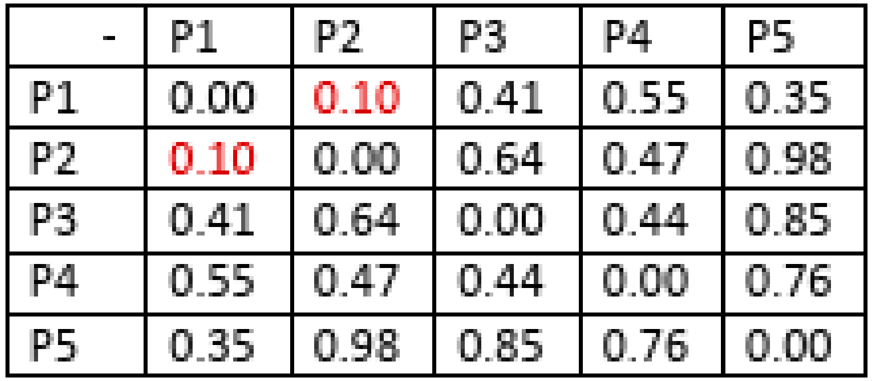


Single Link Dendrogram

Hierarchical Clustering- Solved Example

- For the **complete link or MAX version** of hierarchical clustering, the proximity of two clusters is defined as the maximum of the distance (minimum of the similarity) between any two points in the two different clusters.
- Steps:
  - Using graph terminology, start with all points as singleton clusters.
  - Add links between points one at a time (shortest links first).
  - Group points until all the points are completely linked

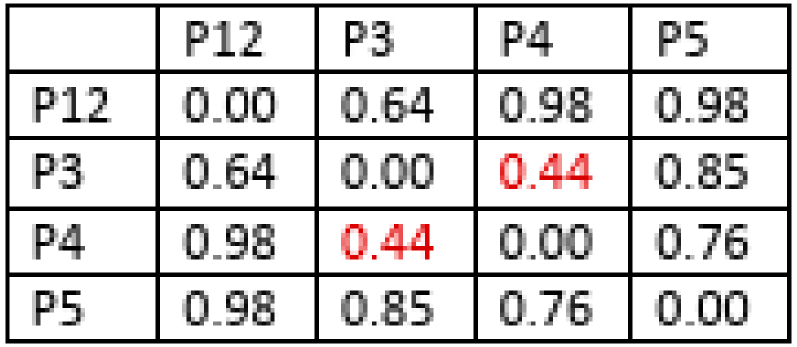
Hierarchical Clustering- Solved Example



-	P1	P2	P3	P4	P5
P1	0.00	0.10	0.41	0.55	0.35
P2	0.10	0.00	0.64	0.47	0.98
P3	0.41	0.64	0.00	0.44	0.85
P4	0.55	0.47	0.44	0.00	0.76
P5	0.35	0.98	0.85	0.76	0.00

Combine P1 and P2:

- $dist(\{P1,P2\},\{P3\}) = max(dist(P1,P3),dist(P2,P3))$
  $= max(0.41,0.64) = 0.64$
- $dist(\{P1,P2\},\{P4\}) = min(dist(P1,P4),dist(P2,P5))$
  $= min(0.55,0.98) = 0.98$
- $dist(\{P1,P2\},\{P5\}) = min(dist(P1,P5),dist(P2,P5))$
  $= min(0.35,0.98) = 0.98$



	P12	P3	P4	P5
P12	0.00	0.64	0.98	0.98
P3	0.64	0.00	0.44	0.85
P4	0.98	0.44	0.00	0.76
P5	0.98	0.85	0.76	0.00

Combine P3 and P4:

- $dist(\{P3,P4\},\{P12\}) = min(dist(P3,P12),dist(P4,P12))$
  $= max(0.64,0.98) = 0.98$
- $dist(\{P3,P4\},\{P5\}) = min(dist(P3,P5),dist(P4,P5))$
- $= max(0.85,0.76) = 0.85$

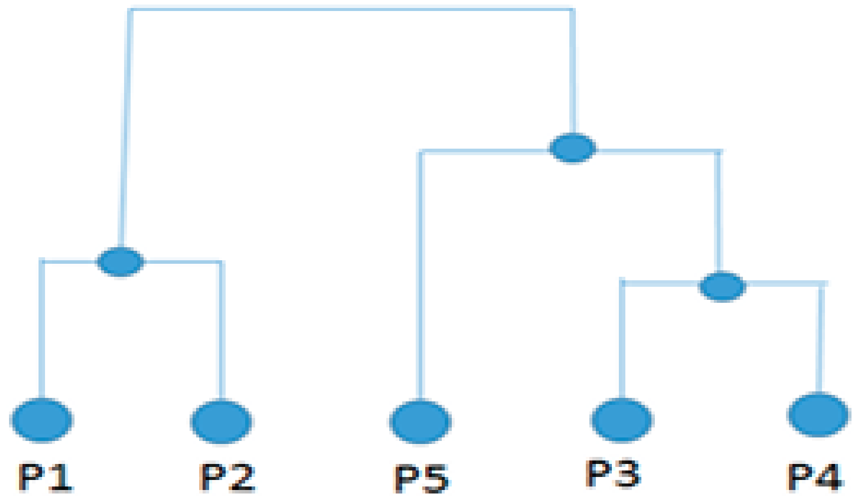
Hierarchical Clustering- Solved Example

- Combine P34 and P5:

$$dist(\{P34,P5\},\{P12\}) = max(dist(P34,P12),dist(P5,P12))$$

$$=max(0.98,0.98) = 0.98$$

	P12	P345
P12	0.00	0.98
P345	0.98	0.00



Complete Link Dendrogram

DBSCAN, or Density-Based Spatial Clustering

- DBSCAN is a Density-Based Spatial Clustering that **groups data points that are closely packed together and marks outliers as noise** based on their density in the feature space.
- It identifies clusters as dense regions in the data space, separated by areas of lower density.
- It's often used for exploratory data analysis and outlier detection.

DBSCAN

Three key concepts:

- **Core Points**: These are points that have at least a minimum number of other points (MinPts) within a specified distance (ε or epsilon).
- **Border Points**: These are points that are within the ε distance of a core point but don't have MinPts neighbors themselves.
- **Noise Points**: These are points that are neither core points nor border points. They're not close enough to any cluster to be included.

DBSCAN

- DBSCAN uses two main parameters:
- **ε (epsilon)**: The maximum distance between two points for them to be considered as neighbors.
- **MinPts**: The minimum number of points required to form a dense region.
- By adjusting these parameters, you can control how the algorithm defines clusters, allowing it to adapt to different types of datasets and clustering requirements.

DBSCAN

Key concepts: density reachability and density connectivity

Density reachability

- A point q is density-reachable from a point p if:
  1. p is a core point (has at least MinPts within ε distance)
  2. There is a chain of points p = p_1, ..., p_n = q where each p_i+1 is directly density-reachable from $p_{i.}$

  In simple r terms, you can reach q from p by stepping through core points, where each step is no larger than ε.

Density connectivity

- Two points p and q are density-connected if there exists a point o such that both p and q are density-reachable from o.
- Density connectivity is the basis for forming clusters in DBSCAN. All points in a cluster are mutually density-connected, and if a point is density-connected to any point in the cluster, it also belongs to that cluster.

DBSCAN - working

- **Identify Core Points**: For each point in the dataset, count the number of points within its **eps** neighborhood. If the count meets or exceeds **MinPts**, mark the point as a **core point**.
- **Form Clusters**: For each core point that is not already assigned to a cluster, create a new cluster. Recursively find all **density-connected points** (points within the **eps** radius of the core point) and add them to the cluster.
- **Density Connectivity**: Two points, **a** and **b**, are **density-connected** if there exists a chain of points where each point is within the **eps** radius of the next, and at least one point in the chain is a core point. This chaining process ensures that all points in a cluster are connected through a series of dense regions.
- **Label Noise Points**: After processing all points, any point that does not belong to a cluster is labeled as **noise**.

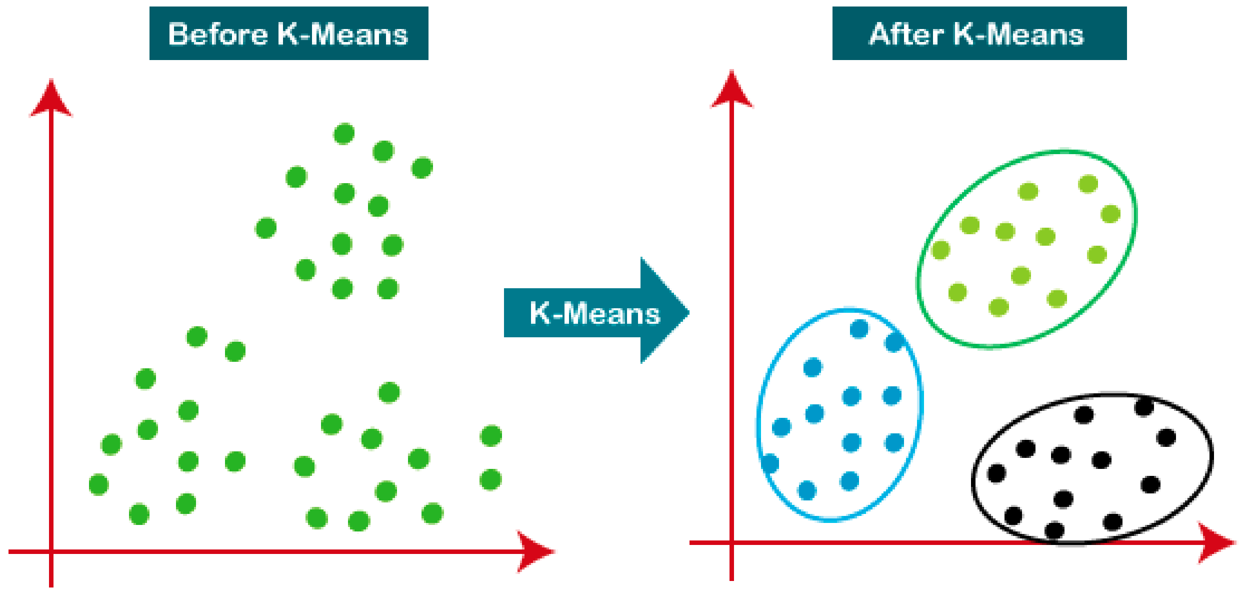
K-Means Clustering

- K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters.
- Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.
- It is an iterative algorithm that divides the unlabeled dataset into k different clusters based on its similar properties.
- It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

- It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.



K-Means Clustering

- Each data point (item) is assigned to the cluster whose current centroid (mean) is closest to it.
- The "closest" is typically determined using a distance metric like Euclidean distance.
- Example: If you have three clusters with centroids at (1,1)(1,1), (5,5)(5,5), and (10,10)(10,10), and a data point at (2,2)(2,2), it will be assigned to the cluster with the centroid at (1,1)(1,1) because that is the closest in terms of distance.
- After all points are categorized into clusters, the centroid of each cluster is recalculated. This is done by finding the average of all the data points currently assigned to that cluster.

K-Means Clustering

- Start by randomly selecting K points from the dataset. These points will act as the initial cluster centroids.
- For each data point in the dataset, calculate the distance between that point and each of the K centroids. Assign the data point to the cluster whose centroid is closest to it. This step effectively forms K clusters.
- Once all data points have been assigned to clusters, recalculate the centroids of the clusters by taking the mean of all data points assigned to each cluster.
- Repeat steps 2 and 3 until convergence. Convergence occurs when the centroids no longer change significantly or when a specified number of iterations is reached.
- Once convergence is achieved, the algorithm outputs the final cluster centroids and the assignment of each data point to a cluster.

END