# Introduction to Machine Learning

– Prof. Balaraman Ravindran | IIT Madras

## Problem Solving Session (Week-10)

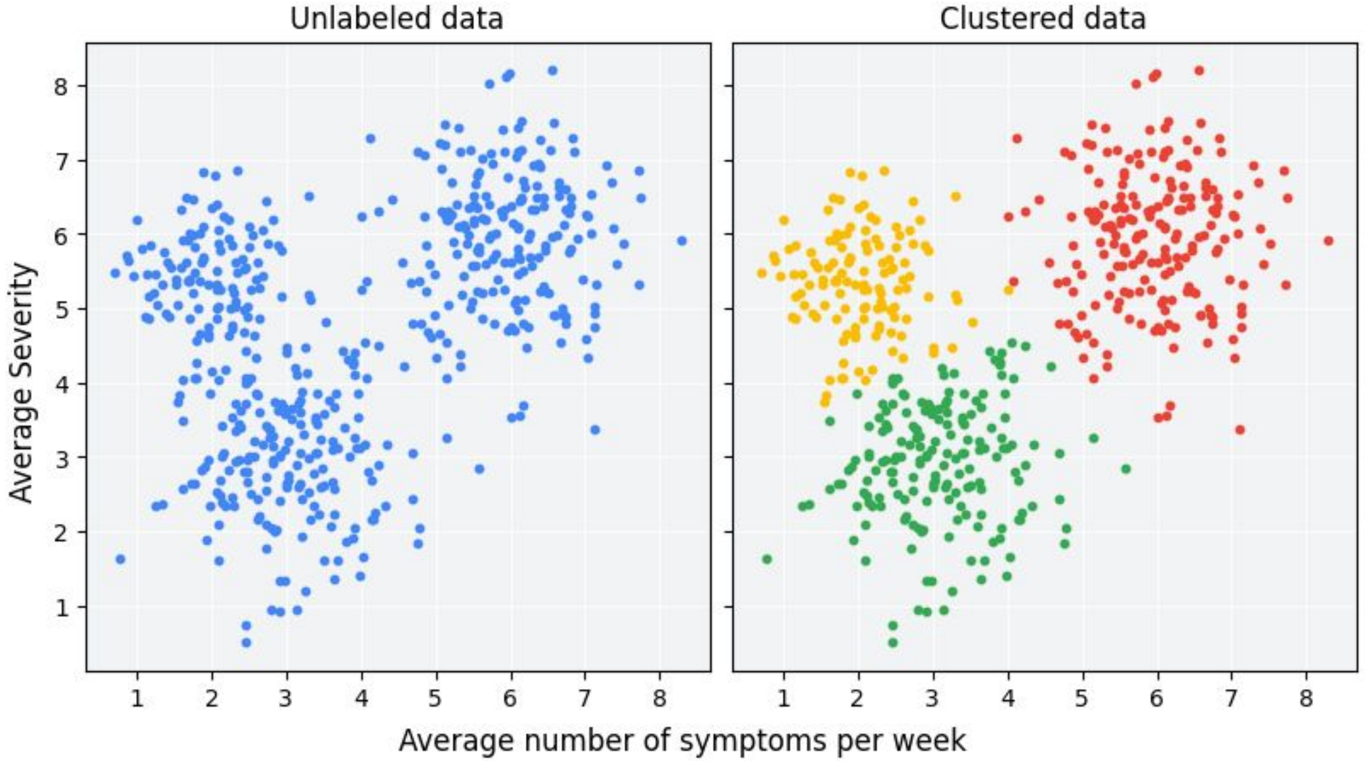
Shreya Bansal

PMRF PhD Scholar
IIT Ropar

## Week-10 Contents

1. Partitional Clustering (K-means)
2. Hierarchical Clustering (Agglomerative)
3. Birch Algorithm
4. Cure Algorithm
5. Density Based Clustering (DB-SCAN)

## What is Clustering?



## What is Clustering?

- **Definition**: Grouping similar data points together.
- **Goal:** Maximize intra-cluster similarity.

  Minimize inter-cluster similarity.
- **Formal Problem:** Partition n data points into k clusters.
- **Challenge:** Number of possible partitions is huge (combinatorial).

## Applications of Clustering

- **Data Mining**: Categorization of unlabeled data.
- **Pre-processing**: Reduce large datasets to representative points (e.g., 10M → 10k clusters).
- **Visualization**: Understand structure in high-dimensional feature spaces.
- **Anomaly Detection**: Identify outliers (e.g., fraud detection).
- **Example**:
- Customer segmentation: Group users by behavior.

## Clustering Algorithms Overview

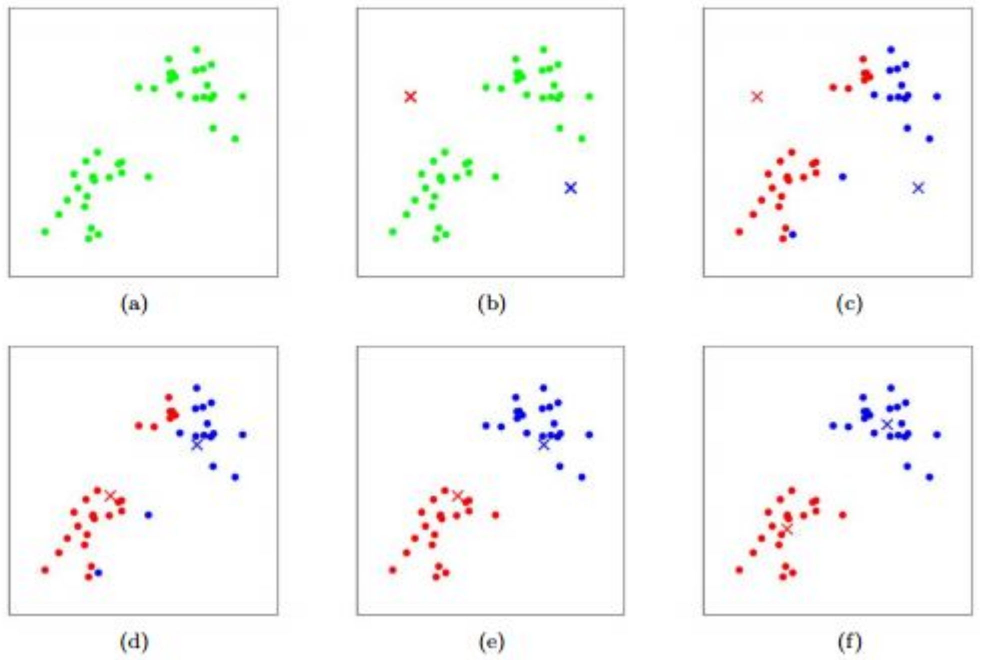
- Three Main Approaches:
  - Partitional (e.g., K-means, K-medoids).
  - Hierarchical (e.g., Agglomerative).
  - Density-Based (e.g., DBSCAN).

- Key Difference:
- Partitional methods search directly for the final k-partition.

# K-Means Clustering

Steps:

1. Initialize k centroids (randomly or heuristically).
2. Assign each point to the nearest centroid.
3. Recompute centroids as cluster means.
4. Repeat until convergence.



## K-Means Clustering

**Steps:**

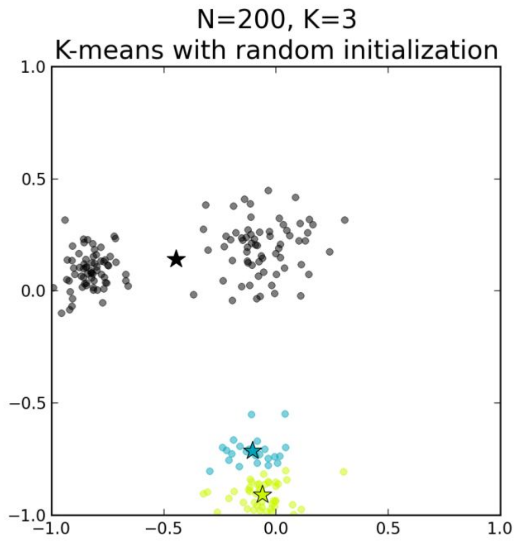
1. **Initialize k centroids (randomly or heuristically).**
2. **Assign each point to the nearest centroid.**
3. **Recompute centroids as cluster means.**
4. **Repeat until convergence.**

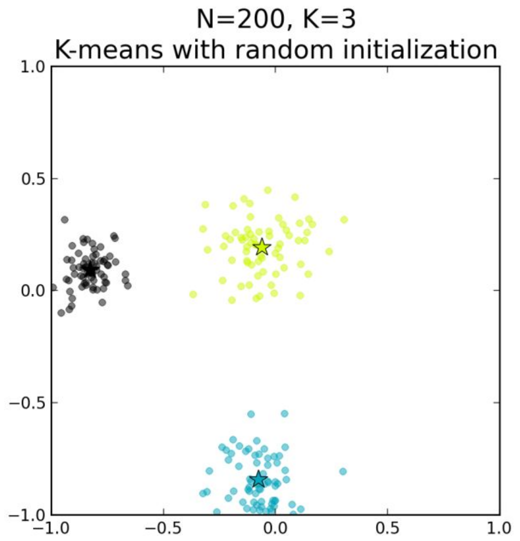
**Pros: Simple, fast.**
**Cons: Sensitive to initialization; assumes spherical clusters.**

# K-Means Initialization Matters!

- Problem: Random seeds can lead to poor local optima.

- Example:
- Bad initialization → Uneven clusters.
- Solution: K-means++ (smart seeding).



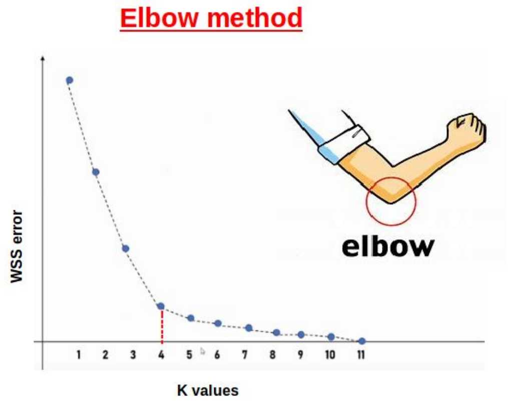


# K-Medoids and PAM

- K-Medoids:
    - Uses actual data points as centroids (medoids).
    - Robust to outliers (median-like behavior).
- PAM (Partitioning Around Medoids):
    - Swap medoids with non-medoids.
    - Keep swaps that improve cluster quality.
- Trade-off: Computationally expensive (O($n^2$))

## How to Choose k?

- Methods:
- Domain Knowledge: Predefined k (e.g., 5 customer segments).
- Elbow Method: Plot k vs. distortion (sum of squared distances).
- Pick k at the "elbow" of the curve.



## Cluster Evaluation Metrics

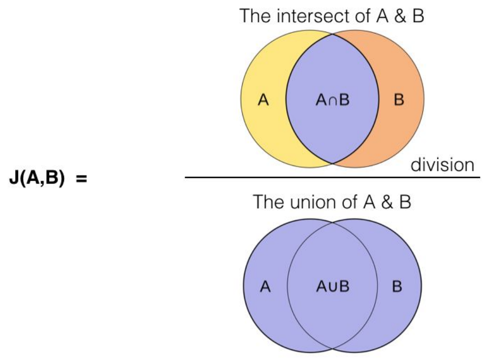
- Diameter: Max/Avg pairwise distance in a cluster.
- Radius: Avg distance to centroid.
- Purity: % of dominant class in a cluster (if labels exist).
- Rand Index: Agreement with reference clustering.

- Formula:
- Purity=(1/N) $\Sigma_{\text{clusters}} \max_{\text{class}} |\text{cluster} \cap \text{class}|$

# Limitations of Clustering

- Ill-Posed Problem: No unique "correct" clustering.
- Distance Metrics: Euclidean fails for categorical data.
- Scalability: PAM is slow for large n.

- Workarounds:
- Use Jaccard similarity for categorical data.
- Sample data before clustering.



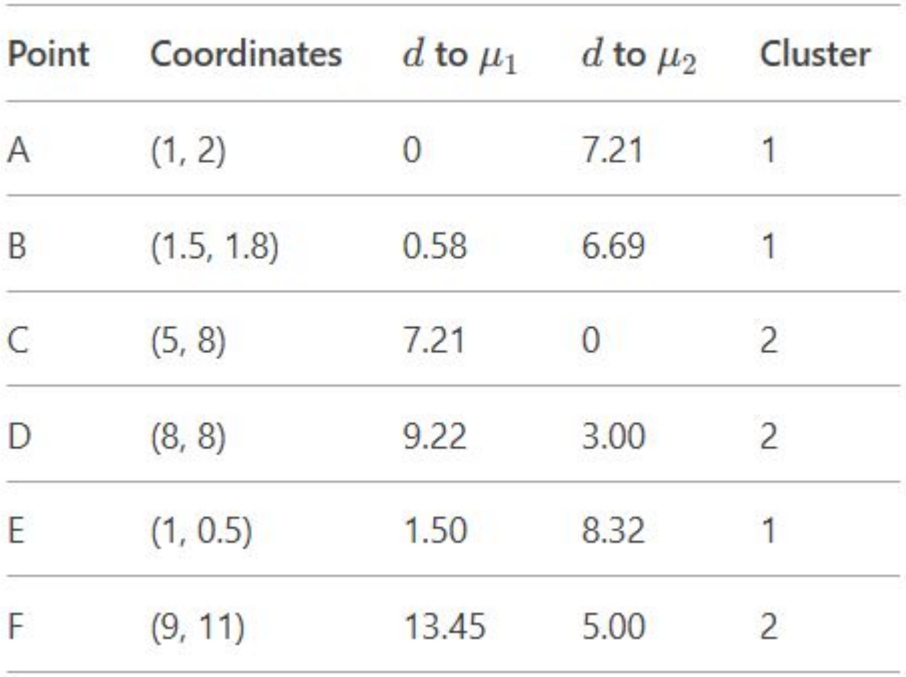
## Example - K-Means

- Given the following 2D data points, perform K-means clustering with K=2. Use Euclidean distance and show the first two iterations.

- Data Points: A(1,2), B(1.5,1.8), C(5,8), D(8,8), E(1,0.5), F(9,11)
- Initial Centroids (chosen randomly):
- $\mu_1$ =(1,2) (Point A)
- $\mu_2$ =(5,8) (Point C)

## Example - K-Means

- **Iteration 1**
- **Step 1: Assign Points to Nearest Centroid**
- **Calculate Euclidean distance (d) from each point to centroids:**



| Point | Coordinates | $d$ to $\mu_1$ | $d$ to $\mu_2$ | Cluster |
|---|---|---|---|---|
| A | (1, 2) | 0 | 7.21 | 1 |
| B | (1.5, 1.8) | 0.58 | 6.69 | 1 |
| C | (5, 8) | 7.21 | 0 | 2 |
| D | (8, 8) | 9.22 | 3.00 | 2 |
| E | (1, 0.5) | 1.50 | 8.32 | 1 |
| F | (9, 11) | 13.45 | 5.00 | 2 |

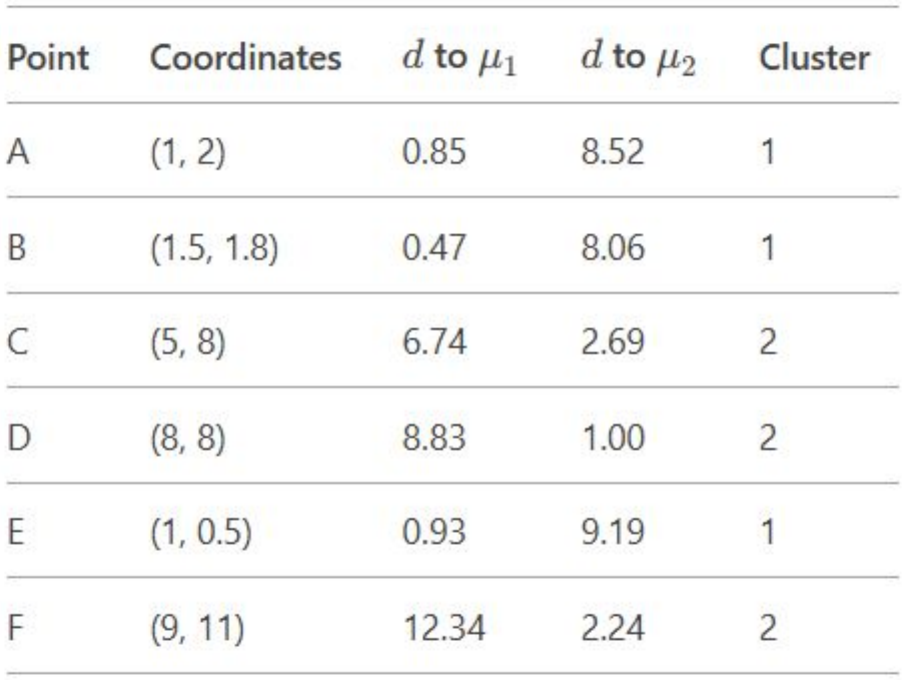
## Example - K-Means

- **Cluster Assignments: Cluster 1: A, B, E            Cluster 2: C, D, F**
- **Step 2: Update Centroids**
- **Recalculate centroids as the mean of assigned points:**

- $\mu_1 = \left(\frac{1+1.5+1}{3}, \frac{2+1.8+0.5}{3}\right) = (1.17, 1.43)$
- $\mu_2 = \left(\frac{5+8+9}{3}, \frac{8+8+11}{3}\right) = (7.33, 9.00)$

## Example - K-Means

- **Iteration 2**
- **Step 1: Reassign Points to New Centroids**



| Point | Coordinates | $d$ to $\mu_1$ | $d$ to $\mu_2$ | Cluster |
|---|---|---|---|---|
| A | (1, 2) | 0.85 | 8.52 | 1 |
| B | (1.5, 1.8) | 0.47 | 8.06 | 1 |
| C | (5, 8) | 6.74 | 2.69 | 2 |
| D | (8, 8) | 8.83 | 1.00 | 2 |
| E | (1, 0.5) | 0.93 | 9.19 | 1 |
| F | (9, 11) | 12.34 | 2.24 | 2 |

## Example - K-Means

Cluster Assignments:

Cluster 1: A, B, E

Cluster 2: C, D, F

Step 2: Check Convergence

Centroid remain unchanged (no reassignments).

Algorithm converges.

## Example - K-Means

**Final Result**

**Clusters:**

**Cluster 1 (Red): A(1,2), B(1.5,1.8), E(1,0.5)**

**Cluster 2 (Blue): C(5,8), D(8,8), F(9,11)**

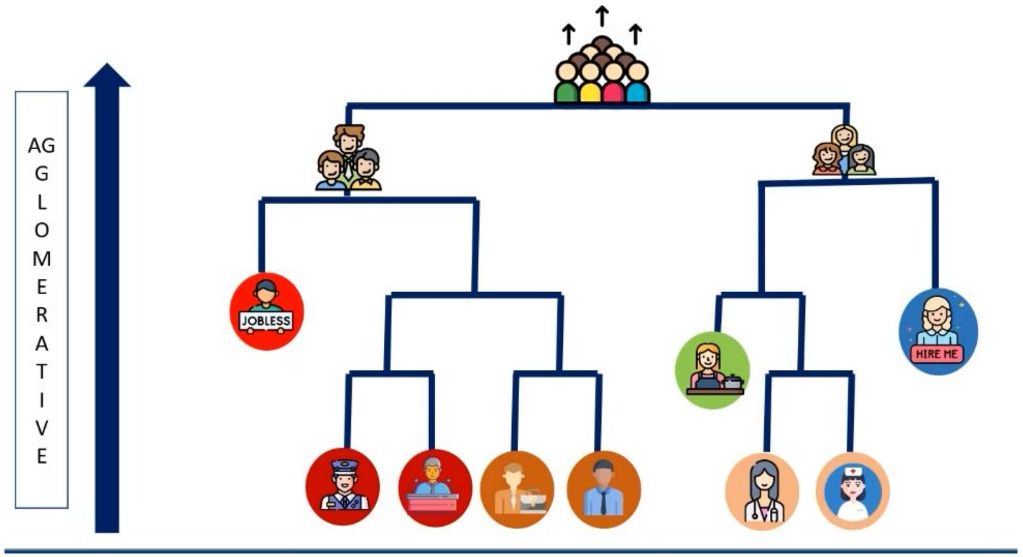
**Final Centroids:**

$\mu_1 = (1.17, 1.43)$

$\mu_2 = (7.33, 9.00)$

# Hierarchical Clustering

- **Hierarchical clustering is a method of cluster analysis that builds a hierarchy of clusters.**
- **It starts with each data point as an individual cluster and successively merges the closest clusters until a single cluster remains.**



## Steps in Hierarchical Clustering

1. **Start with each data point as a separate cluster.**
2. **Compute distances between clusters (initially between individual points).**
3. **Merge the closest clusters.**
4. **Repeat until only one cluster remains.**
5. **Generate a dendrogram to visualize the merging process.**

## Distance Measures

- There are multiple ways to measure the distance between clusters:
- Single Link Clustering:
  - Distance is defined by the closest pair of points between clusters.
  - May result in long, chain-like clusters.
- Complete Link Clustering:
  - Distance is defined by the farthest pair of points between clusters.
  - Tends to produce compact, well-separated clusters.
- Average Link Clustering:
  - Distance is the average of all pairwise distances between points in two clusters.

## Distance Measures

Measuring Distance Between Clusters

- Centroid-Based Distance:
    - Distance is measured between the centroids of two clusters.
- Radius-Based Distance:
    - Clusters are merged based on the radius of the combined cluster.
- Diameter-Based Distance:
    - Clusters are merged based on the diameter of the combined cluster.

# Distance Metrics for Data Points

- The distance measure between individual data points depends on the type of data and can be:
- Euclidean Distance
- Manhattan Distance
- Jaccard Similarity
- Cosine Similarity
- Other domain-specific distance measures

## Dendrograms

- A dendrogram is a tree-like diagram that illustrates the order and levels at which clusters were merged.
- The height at which two clusters are joined represents the distance between them.
- To determine the final clusters, the dendrogram is cut at a chosen threshold.

## Pros and Cons of Hierarchical Clustering

- Advantages of Hierarchical Clustering
- ✅ No need to specify the number of clusters K in advance.
- ✅ Provides a complete hierarchical decomposition of the dataset.
- ✅ Can visualize the clustering process using a dendrogram.

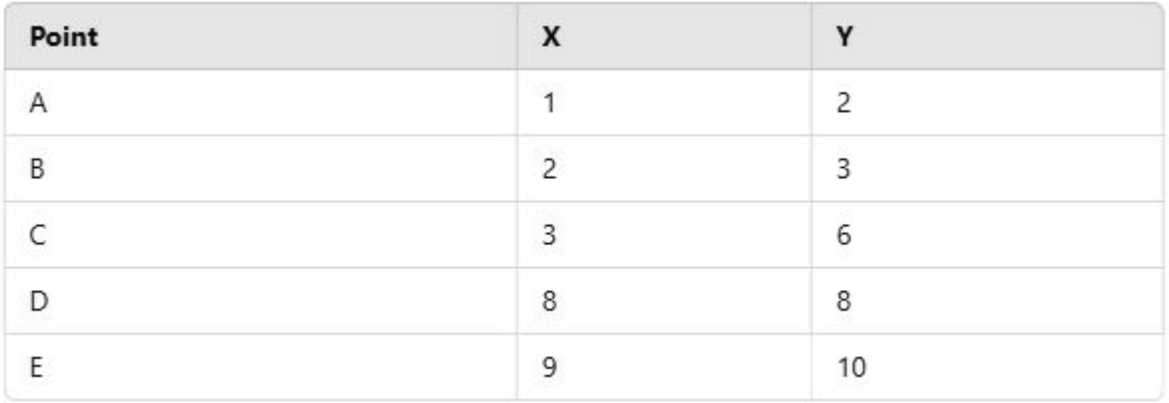
- Disadvantages of Hierarchical Clustering
- ❌ Computationally expensive for large datasets.
- ❌ Sensitive to noise and outliers.
- ❌ Merging decisions are irreversible.

## Choosing the Number of Clusters

- The knee method or thresholding the dendrogram can help determine the optimal number of clusters.
- Look for a large jump in merging distances to decide where to cut the dendrogram.

## Example : Hierarchical Clustering

- **Given the following dataset with five points:**
- **Perform hierarchical clustering using the Euclidean distance and single linkage.**

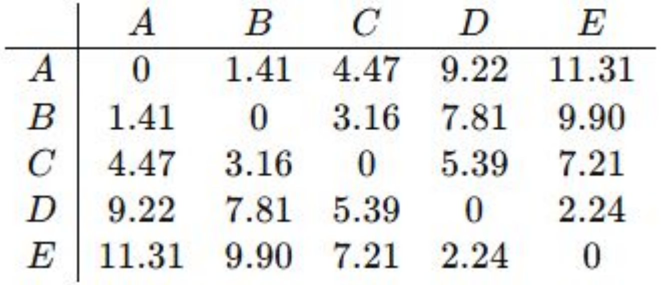


| Point | X | Y |
|---|---|---|
| A | 1 | 2 |
| B | 2 | 3 |
| C | 3 | 6 |
| D | 8 | 8 |
| E | 9 | 10 |

## Example : Hierarchical Clustering

- Solution Steps
- Compute the pairwise Euclidean distances:
- Example: Distance between A and B:
- Create an initial distance matrix (distances between all points are computed).
- Find the closest pair: A and B are closest, so merge them into one cluster.
- Recompute the distance matrix:
- Using single linkage, the distance between clusters is the minimum of all pairwise distances.
- Repeat until only one cluster remains.

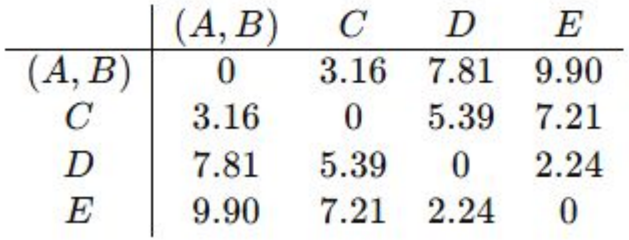
# Example : Hierarchical Clustering



|  | $A$ | $B$ | $C$ | $D$ | $E$ |
|---|---|---|---|---|---|
| $A$ | 0 | 1.41 | 4.47 | 9.22 | 11.31 |
| $B$ | 1.41 | 0 | 3.16 | 7.81 | 9.90 |
| $C$ | 4.47 | 3.16 | 0 | 5.39 | 7.21 |
| $D$ | 9.22 | 7.81 | 5.39 | 0 | 2.24 |
| $E$ | 11.31 | 9.90 | 7.21 | 2.24 | 0 |

**Step 1: (A,B)=1.41 —C1- Merge them**

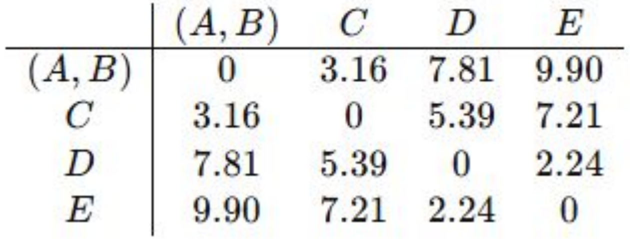
**Step2: Update distance matrix**



|  | $(A, B)$ | $C$ | $D$ | $E$ |
|---|---|---|---|---|
| $(A, B)$ | 0 | 3.16 | 7.81 | 9.90 |
| $C$ | 3.16 | 0 | 5.39 | 7.21 |
| $D$ | 7.81 | 5.39 | 0 | 2.24 |
| $E$ | 9.90 | 7.21 | 2.24 | 0 |

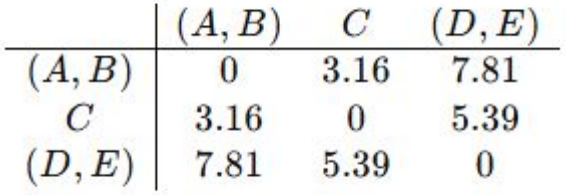
# Example : Hierarchical Clustering

Step 3: (D,E)=2.24 —C2- Merge them



|  | $(A,B)$ | $C$ | $D$ | $E$ |
|---|---|---|---|---|
| $(A,B)$ | 0 | 3.16 | 7.81 | 9.90 |
| $C$ | 3.16 | 0 | 5.39 | 7.21 |
| $D$ | 7.81 | 5.39 | 0 | 2.24 |
| $E$ | 9.90 | 7.21 | 2.24 | 0 |

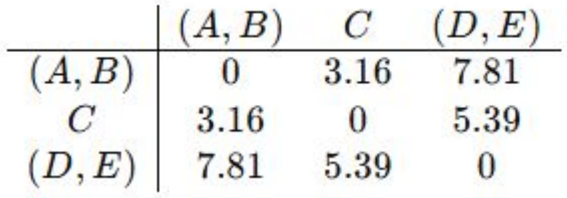
Step4: Update distance matrix



|  | $(A,B)$ | $C$ | $(D,E)$ |
|---|---|---|---|
| $(A,B)$ | 0 | 3.16 | 7.81 |
| $C$ | 3.16 | 0 | 5.39 |
| $(D,E)$ | 7.81 | 5.39 | 0 |

## Example : Hierarchical Clustering

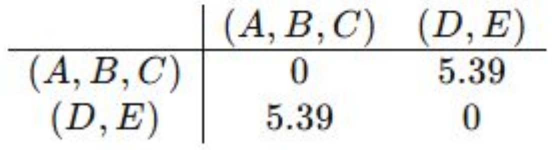
**Step 5: (C1,C)=3.16 —C3- Merge them**



| | $(A,B)$ | $C$ | $(D,E)$ |
|---|---|---|---|
| $(A,B)$ | 0 | 3.16 | 7.81 |
| $C$ | 3.16 | 0 | 5.39 |
| $(D,E)$ | 7.81 | 5.39 | 0 |

**Step 6: Update distance matrix**

**Step 7: Merge all**

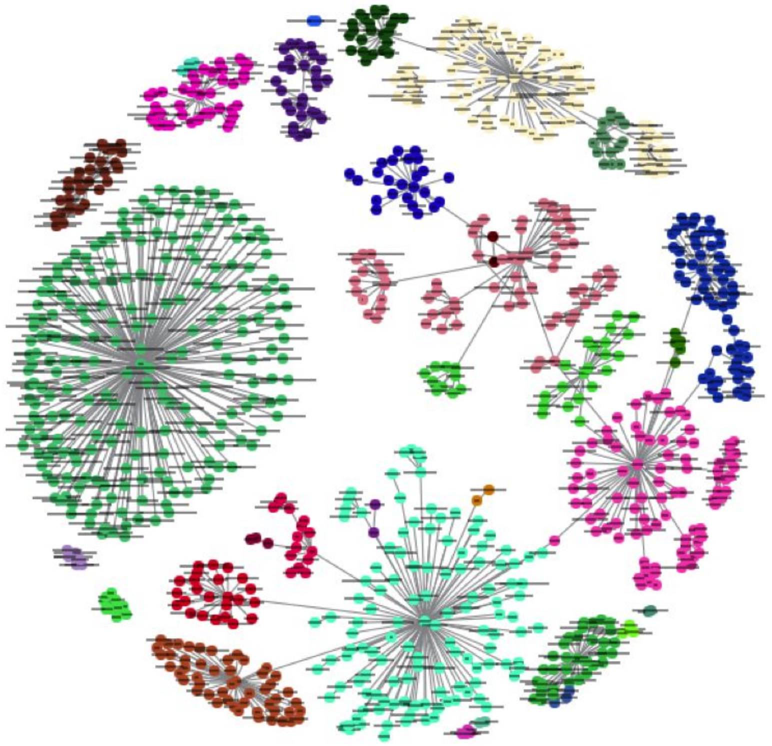


| | $(A,B,C)$ | $(D,E)$ |
|---|---|---|
| $(A,B,C)$ | 0 | 5.39 |
| $(D,E)$ | 5.39 | 0 |

```
          (C4)
      /          \
   (C3)        (C2)
  /    \        /  \
(C1)    C    D    E
 /  \
A    B
```

## BIRCH Clustering for Large Datasets

- Balanced Iterative Reducing and Clustering using Hierarchies
- Scalable clustering for massive data
- Two-phase approach: CF-Tree (Clustering Features) + Refinement
- Memory-efficient hierarchical clustering



## Why BIRCH?

- Challenges with Large Data:
  - K-means requires multiple passes (expensive for disk/network I/O).
  - Hierarchical methods are $O(n^2)$ in memory.

- BIRCH Solution:
  - Phase 1: Build a CF-Tree (summarize data into tight subclusters).
  - Phase 2: Refine clusters using the CF-Tree's summaries.

- Key Idea: Reduce n to k representatives (e.g., 1M → 10K).

## Clustering Feature (CF) Vector

- Definition: A tuple summarizing a cluster: CF=(N, LS ,SS)
    - N: Number of points in the cluster.
    - LS : Linear Sum of points (vector).
    - SS: Sum of Squares (scalar).
- Example:
- For cluster with points (1,2) and (3,4):
    - CF=(2, (1+3,2+4), (1+9+4+16) = (2,(4,6),30 )
- Properties:
- CFs are additive (merge clusters by adding CFs).
- Enable centroid (LS/N, radius, diameter calculations.

## Clustering Feature (CF) Vector

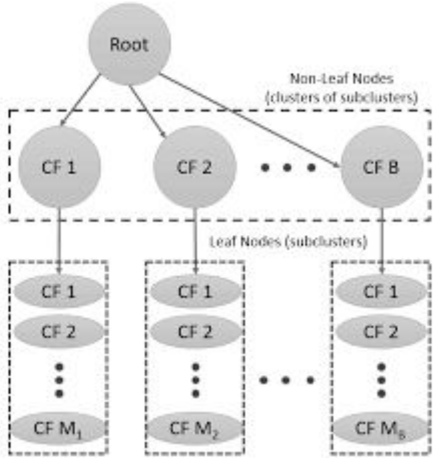
- **Enable centroid (LS/N, radius, diameter calculations.**

Cluster's centroid, $$X_0 = \frac{LS}{n}$$

Cluster's radius, $$R = \sqrt{\frac{\sum_{i=1}^{n}(x_i - X_0)^2}{n}} = \sqrt{\frac{n(SS) - 2LS^2 - n(LS)}{n^2}}$$

Cluster's diameter, $$D = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\left(x_i - x_j\right)^2}{n(n-1)}} = \sqrt{\frac{2n(SS) - 2(LS)^2}{n(n-1)}}$$

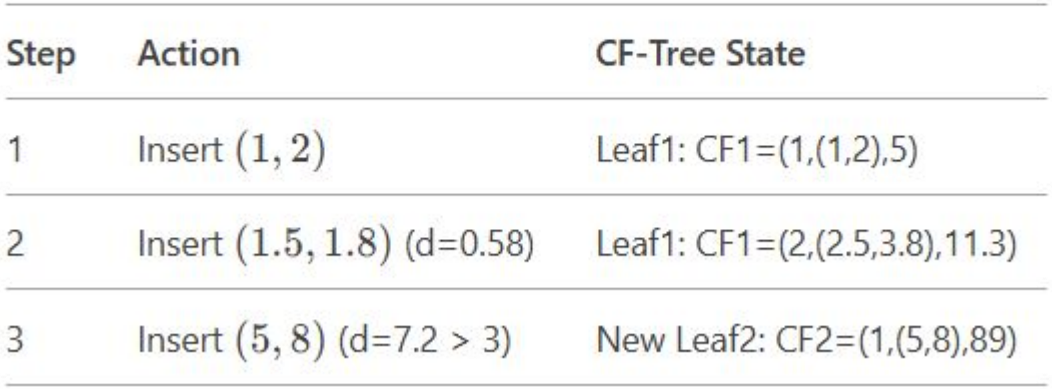
## CF-Tree Construction



- **Structure:**
  - **Leaf Nodes: Store CF entries (subclusters).**
  - **Non-Leaf Nodes: Store CFs summarizing child nodes.**
- **Algorithm Steps:**
  - **Insert a point into the closest CF in the leaf (based on centroid/diameter threshold).**
  - **If leaf exceeds max entries (e.g., 10), split leaf and propagate CFs upward.**
  - **Repeat until all points are processed.**

# Phase 1 Example (CF-Tree Build)

- **Data Points: (1,2),(1.5,1.8),(5,8),(8,8),(9,11)**
- **Threshold: Max diameter = 3.0**
- **Key Insight: Points are grouped into tight subclusters dynamically.**



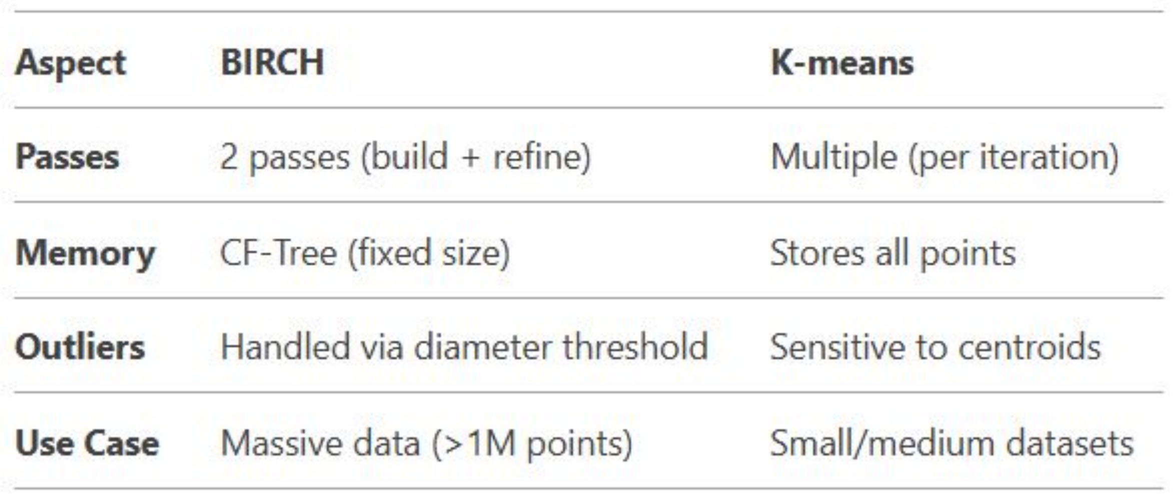
| Step | Action | CF-Tree State |
|---|---|---|
| 1 | Insert $(1,2)$ | Leaf1: CF1=(1,(1,2),5) |
| 2 | Insert $(1.5,1.8)$ (d=0.58) | Leaf1: CF1=(2,(2.5,3.8),11.3) |
| 3 | Insert $(5,8)$ (d=7.2 > 3) | New Leaf2: CF2=(1,(5,8),89) |

## Phase 2: Global Clustering

- Input: CF-Tree leaf entries (subcluster centroids).
- Process:
  - Run any clustering algorithm (e.g., K-means, hierarchical) on the centroids.
  - Assign original data points to the nearest final centroid (1 additional pass).

- Advantages:
  - Scalable: Works with summaries, not raw data.
  - Flexible: Choose any clustering method for refinement.

## BIRCH vs. K-means



| Aspect | BIRCH | K-means |
|---|---|---|
| **Passes** | 2 passes (build + refine) | Multiple (per iteration) |
| **Memory** | CF-Tree (fixed size) | Stores all points |
| **Outliers** | Handled via diameter threshold | Sensitive to centroids |
| **Use Case** | Massive data (>1M points) | Small/medium datasets |

## Limitations of BIRCH

- Order Sensitivity: Early points influence CF-Tree structure.
- Threshold Tuning: Diameter threshold impacts cluster granularity.
- Non-Spherical Clusters: Struggles with arbitrary shapes (like K-means).

- Workaround:
- Use BIRCH for initial reduction, then DBSCAN for refinement.

## CURE: Clustering Using Representatives

1. Handles non-convex clusters
2. Sampling-based approach for scalability
3. Shrinkage-based outlier robustness

## Why CURE?

- Limitations of Traditional Methods:
- K-means: Only convex clusters (spherical shapes).
- Hierarchical Methods: $O(n^2)$ complexity → infeasible for large n.

- CURE's Solution:
- Sampling: Work with a memory-friendly subset of data.
- Representative Points: Capture cluster boundaries (not just centroids).
- Shrinkage: Mitigate outlier influence.

## Key Steps of CURE

- Sampling: Randomly partition data into subsets (fit in memory).
- Initial Clustering: Run hierarchical clustering on each subset.

- Representative Points:
- For each cluster, pick m farthest points from centroid.
- Shrink them toward centroid by factor $\alpha$ (e.g., $\alpha$=0.3).
- Reassignment: Assign points to the closest representative.
- Merge: Combine subsets' representatives and recluster.

## Representative Points Selection

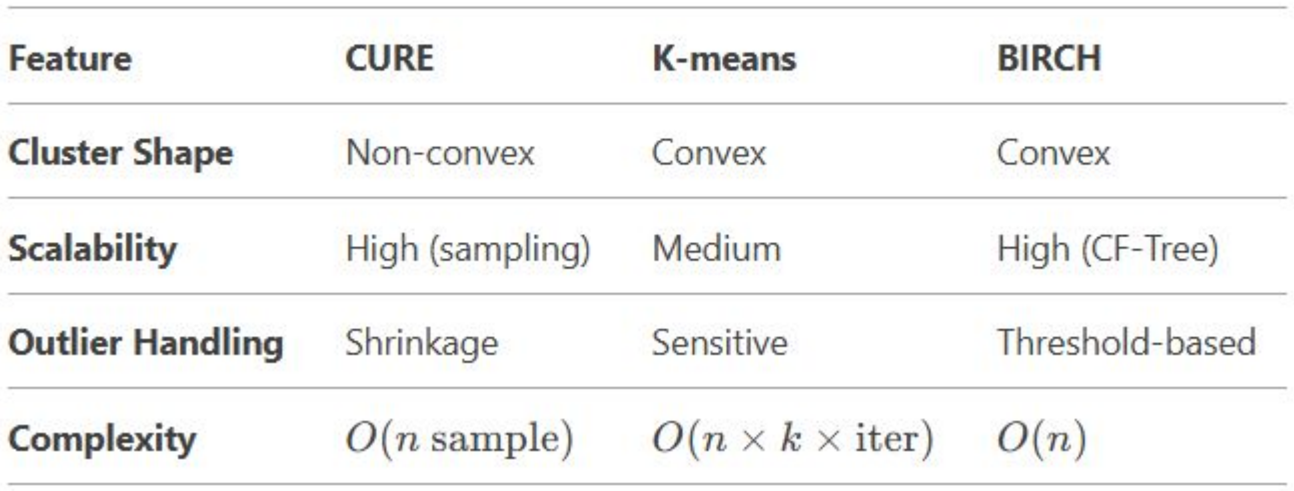
- Process:
  - Compute centroid μ of a cluster.
  - Find farthest point p1 from μ.
  - Find farthest point p2 from p1 .
  - Repeat for m points.
  - Shrink: Move each pi toward μ by $\alpha \times d(pi,\mu)$
- Example:
  - Cluster points: ((1,1),(1,2),(5,5),(6,6).
  - Centroid μ=(3.25,3.5).
  - Farthest point: p1 =(6,6).
  - Shrunk point ($\alpha$=0.2):
  - $p1' = (6-0.2\times(6-3.25),\ 6-0.2\times(6-3.5)) \approx (5.25,5.5)$

## Parallelization in CURE

- Scalability Trick:
  - Split data into k random partitions.
  - Process each partition independently (parallelizable).
  - Merge results by clustering all representatives.

- Example:
  - 1M points → 10 partitions of 100K each.
  - Each partition → 100 clusters × 4 reps = 400 points.
  - Final merge: 4K points → manageable clustering.
  - Advantage: Avoids full $O(n^2)$ computations.

# CURE vs. K-means vs. BIRCH



| Feature | CURE | K-means | BIRCH |
|---|---|---|---|
| **Cluster Shape** | Non-convex | Convex | Convex |
| **Scalability** | High (sampling) | Medium | High (CF-Tree) |
| **Outlier Handling** | Shrinkage | Sensitive | Threshold-based |
| **Complexity** | $O(n \text{ sample})$ | $O(n \times k \times \text{iter})$ | $O(n)$ |

## Parameters in CURE

- Number of Representatives (m): Typically 5–20.
- Shrinkage (α): 0.2–0.5 (balances outlier robustness and boundary accuracy).
- Sample Size: As large as memory allows.

- Trade-offs:
    - Larger m: Better boundary detection but higher overhead.
    - Smaller α: More outlier resistance but less precise boundaries.

## Limitations of CURE

- Parameter Sensitivity: Performance depends on m, $\alpha$, and sample size.
- Order Dependency: Initial sampling affects results.
- Overhead: Representative selection adds computation.

- Workaround:
- Use multiple samples and aggregate (e.g., ensemble clustering).

## DBSCAN: Density-Based Clustering

Handling Arbitrary Cluster Shapes and Noise

Key concepts: Core points, density-reachability, noise

Parameters: ε (epsilon) and minPts

Advantages over K-means and hierarchical clustering

## Why DBSCAN?

**Limitations of Traditional Methods:**

**K-means: Only convex clusters (fails on non-spherical shapes).**

**Hierarchical: Computationally expensive ($O(n^2)$)**

**DBSCAN's Solution:**

**Density-based: Clusters are dense regions separated by low-density areas.**

**Noise Handling: Automatically identifies outliers.**

## Key Definitions

ε (epsilon): Radius of neighborhood around a point.

minPts: Minimum points to define a dense region.

Core Point: A point with ≥ minPts within ε.

Border Point: Fewer than minPts but reachable from a core point.

Noise Point: Not reachable from any core point.

## Density-Reachability and Connectivity

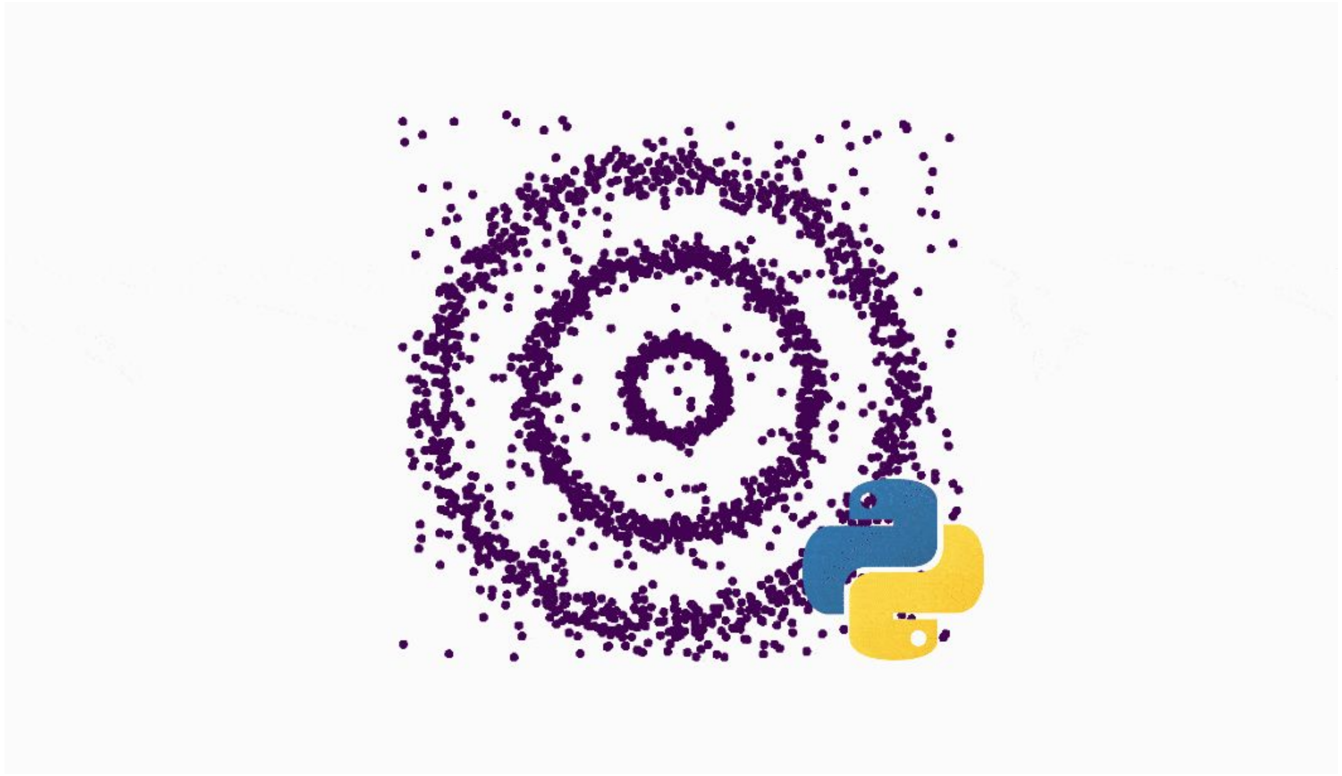
Density-Reachable: Point p is reachable from q if there’s a path of core points within ε.

Density-Connected: Points p and q are connected if they share a common core point.

Cluster Definition:

A cluster is a set of density-connected points.

## DBSCAN Algorithm Steps



## DBSCAN Algorithm Steps

- Random Start: Pick an unvisited point.
- Core Check: Count neighbors in ε-radius.
- If ≥ minPts, mark as core and expand cluster.
- Expand Cluster: Recursively add density-reachable points.
- Repeat: Until all points are visited.

## Parameter Sensitivity

- Effects of ε and minPts:
- Large ε/Small minPts: Fewer, larger clusters (may merge true clusters).
- Small ε/Large minPts: More clusters, more noise.
- 
- Rule of Thumb:
- minPts: Start with minPts≥dimensions+1
- ε: Use k-distance plot (find "knee" for optimal ε).

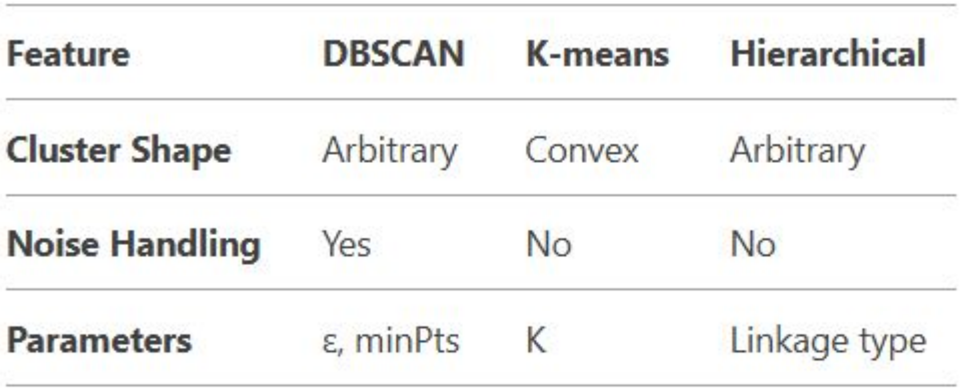
## Advantages & Limitations

- **Pros:**
- **Handles arbitrary shapes and noise.**
- **No need to specify cluster count (unlike K-means).**

- **Cons:**
- **Sensitive to ε and minPts.**
- **Struggles with varying densities.**

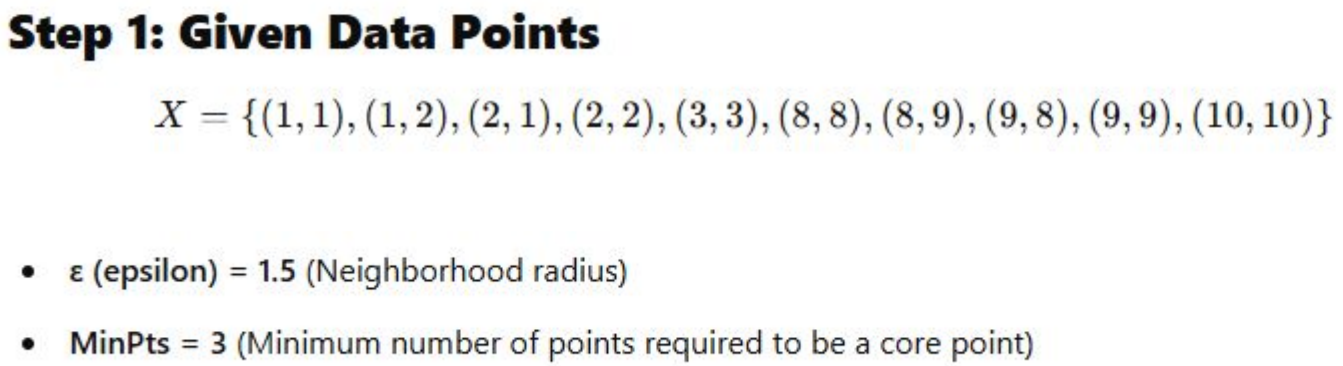
- **Alternatives:**
- **OPTICS: Handles varying ε (hierarchical density).**
- **HDBSCAN: Automates parameter selection.**

# DBSCAN vs. K-means vs. Hierarchical



| Feature | DBSCAN | K-means | Hierarchical |
|---|---|---|---|
| **Cluster Shape** | Arbitrary | Convex | Arbitrary |
| **Noise Handling** | Yes | No | No |
| **Parameters** | ε, minPts | K | Linkage type |

## Example:DBSCAN



### Step 1: Given Data Points

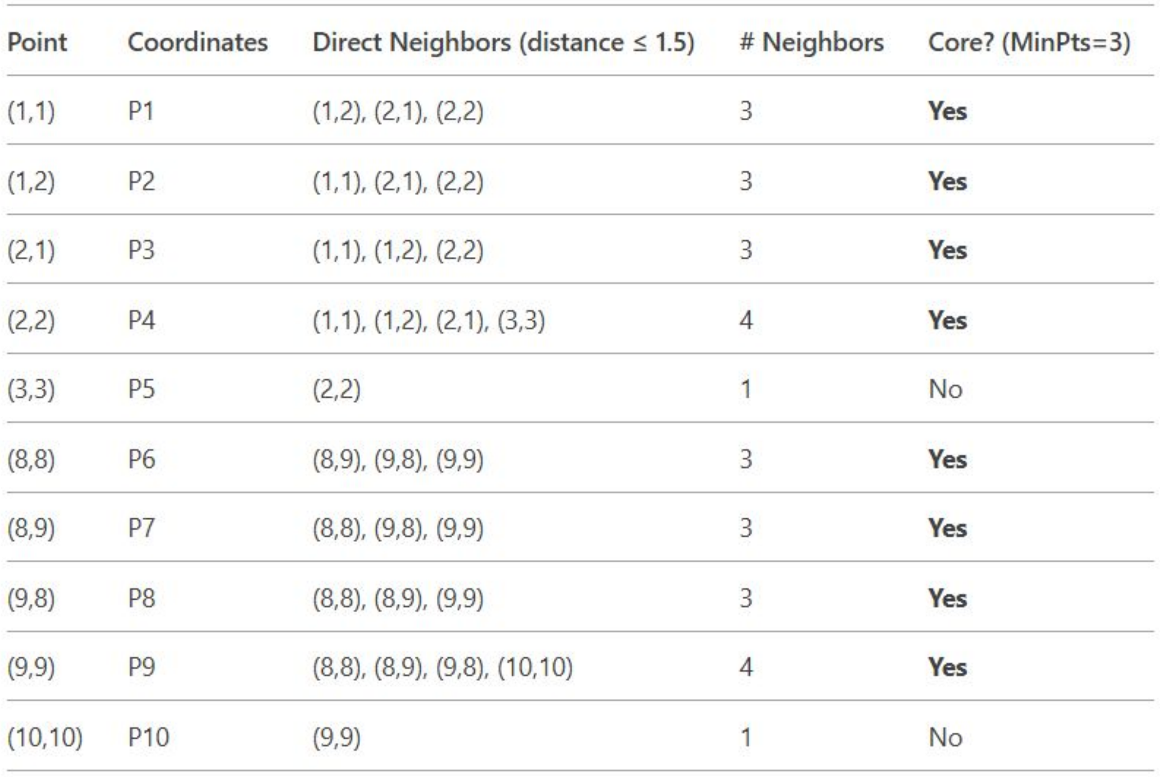
$$X = \{(1,1),(1,2),(2,1),(2,2),(3,3),(8,8),(8,9),(9,8),(9,9),(10,10)\}$$

- **ε (epsilon) = 1.5** (Neighborhood radius)
- **MinPts = 3** (Minimum number of points required to be a core point)

# Example:DBSCAN

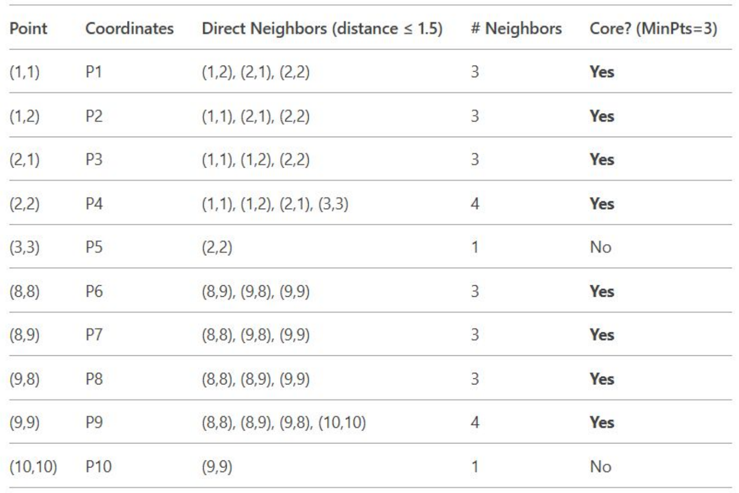
| Point | (1,1) | (1,2) | (2,1) | (2,2) | (3,3) | (8,8) | (8,9) | (9,8) | (9,9) | (10,10) |
|---|---|---|---|---|---|---|---|---|---|---|
| **(1,1)** | 0 | 1✅ | 1✅ | 1.41✅ | 2.83❌ | 9.90❌ | 10.63❌ | 10.63❌ | 11.31❌ | 12.73❌ |
| **(1,2)** | 1✅ | 0 | 1.41✅ | 1✅ | 2.23❌ | 9.22❌ | 9.90❌ | 9.90❌ | 10.63❌ | 12.08❌ |
| **(2,1)** | 1✅ | 1.41✅ | 0 | 1✅ | 2.23❌ | 9.22❌ | 9.90❌ | 9.90❌ | 10.63❌ | 12.08❌ |
| **(2,2)** | 1.41✅ | 1✅ | 1✅ | 0 | 1.41✅ | 8.48❌ | 9.22❌ | 9.22❌ | 9.90❌ | 11.31❌ |
| **(3,3)** | 2.83❌ | 2.23❌ | 2.23❌ | 1.41✅ | 0 | 7.07❌ | 7.81❌ | 7.81❌ | 8.48❌ | 9.90❌ |
| **(8,8)** | 9.90❌ | 9.22❌ | 9.22❌ | 8.48❌ | 7.07❌ | 0 | 1✅ | 1✅ | 1.41✅ | 2.83❌ |
| **(8,9)** | 10.63❌ | 9.90❌ | 9.90❌ | 9.22❌ | 7.81❌ | 1✅ | 0 | 1.41✅ | 1✅ | 2.23❌ |
| **(9,8)** | 10.63❌ | 9.90❌ | 9.90❌ | 9.22❌ | 7.81❌ | 1✅ | 1.41✅ | 0 | 1✅ | 2.23❌ |
| **(9,9)** | 11.31❌ | 10.63❌ | 10.63❌ | 9.90❌ | 8.48❌ | 1.41✅ | 1✅ | 1✅ | 0 | 1.41✅ |
| **(10,10)** | 12.73❌ | 12.08❌ | 12.08❌ | 11.31❌ | 9.90❌ | 2.83❌ | 2.23❌ | 2.23❌ | 1.41✅ | 0 |

# Example:DBSCAN

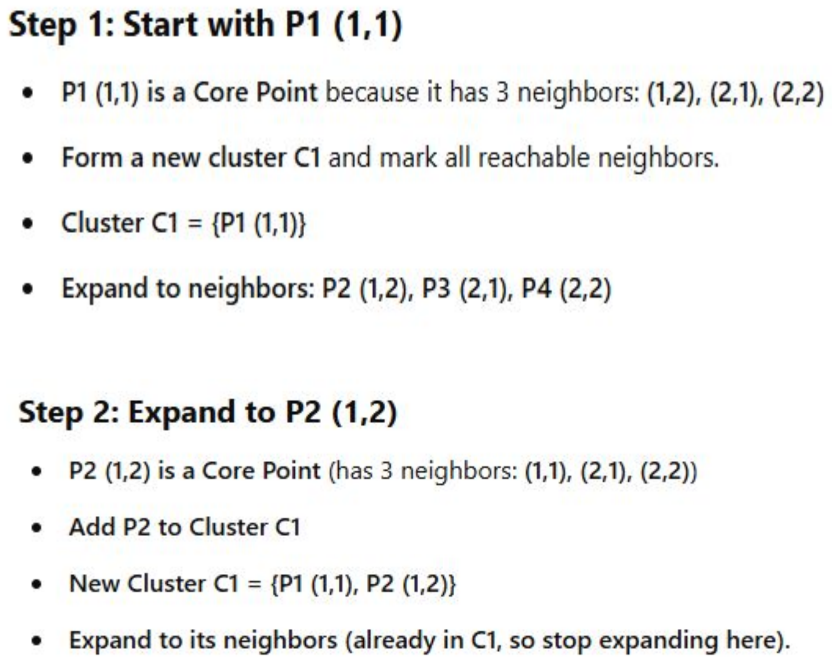


| Point | Coordinates | Direct Neighbors (distance ≤ 1.5) | # Neighbors | Core? (MinPts=3) |
|---|---|---|---|---|
| (1,1) | P1 | (1,2), (2,1), (2,2) | 3 | **Yes** |
| (1,2) | P2 | (1,1), (2,1), (2,2) | 3 | **Yes** |
| (2,1) | P3 | (1,1), (1,2), (2,2) | 3 | **Yes** |
| (2,2) | P4 | (1,1), (1,2), (2,1), (3,3) | 4 | **Yes** |
| (3,3) | P5 | (2,2) | 1 | No |
| (8,8) | P6 | (8,9), (9,8), (9,9) | 3 | **Yes** |
| (8,9) | P7 | (8,8), (9,8), (9,9) | 3 | **Yes** |
| (9,8) | P8 | (8,8), (8,9), (9,9) | 3 | **Yes** |
| (9,9) | P9 | (8,8), (8,9), (9,8), (10,10) | 4 | **Yes** |
| (10,10) | P10 | (9,9) | 1 | No |

## Example:DBSCAN



| Point | Coordinates | Direct Neighbors (distance ≤ 1.5) | # Neighbors | Core? (MinPts=3) |
|---|---|---|---|---|
| (1,1) | P1 | (1,2), (2,1), (2,2) | 3 | **Yes** |
| (1,2) | P2 | (1,1), (2,1), (2,2) | 3 | **Yes** |
| (2,1) | P3 | (1,1), (1,2), (2,2) | 3 | **Yes** |
| (2,2) | P4 | (1,1), (1,2), (2,1), (3,3) | 4 | **Yes** |
| (3,3) | P5 | (2,2) | 1 | No |
| (8,8) | P6 | (8,9), (9,8), (9,9) | 3 | **Yes** |
| (8,9) | P7 | (8,8), (9,8), (9,9) | 3 | **Yes** |
| (9,8) | P8 | (8,8), (8,9), (9,9) | 3 | **Yes** |
| (9,9) | P9 | (8,8), (8,9), (9,8), (10,10) | 4 | **Yes** |
| (10,10) | P10 | (9,9) | 1 | No |



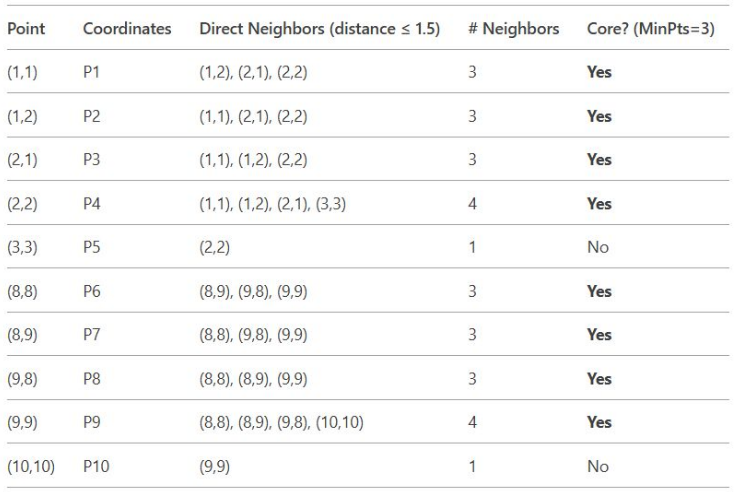
### Step 1: Start with P1 (1,1)

- **P1 (1,1) is a Core Point** because it has 3 neighbors: **(1,2), (2,1), (2,2)**
- **Form a new cluster C1** and mark all reachable neighbors.
- Cluster C1 = {P1 (1,1)}
- Expand to neighbors: P2 (1,2), P3 (2,1), P4 (2,2)

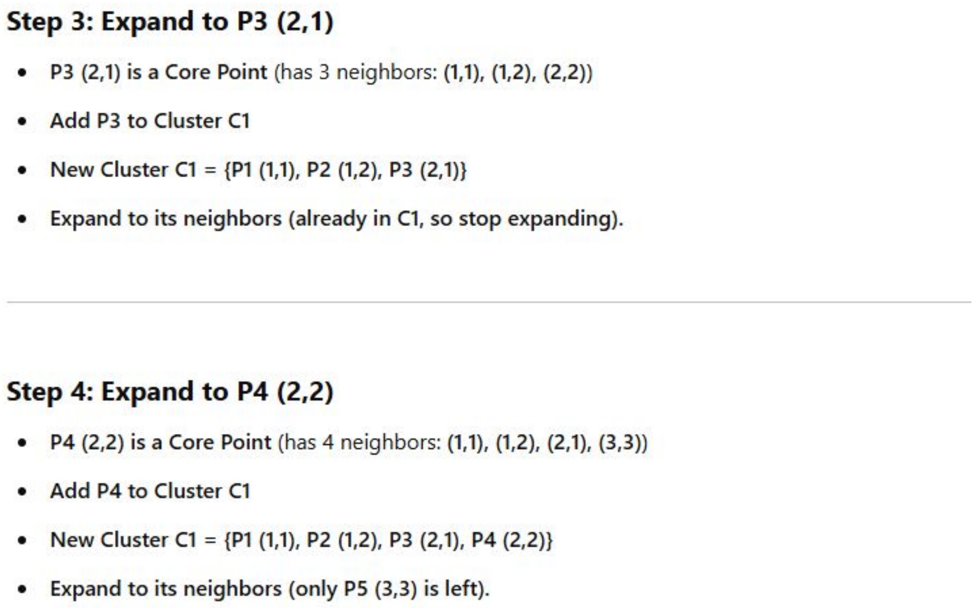
### Step 2: Expand to P2 (1,2)

- **P2 (1,2) is a Core Point** (has 3 neighbors: **(1,1), (2,1), (2,2)**)
- Add P2 to Cluster C1
- New Cluster C1 = {P1 (1,1), P2 (1,2)}
- Expand to its neighbors (already in C1, so stop expanding here).

# Example:DBSCAN



| Point | Coordinates | Direct Neighbors (distance ≤ 1.5) | # Neighbors | Core? (MinPts=3) |
|---|---|---|---|---|
| (1,1) | P1 | (1,2), (2,1), (2,2) | 3 | **Yes** |
| (1,2) | P2 | (1,1), (2,1), (2,2) | 3 | **Yes** |
| (2,1) | P3 | (1,1), (1,2), (2,2) | 3 | **Yes** |
| (2,2) | P4 | (1,1), (1,2), (2,1), (3,3) | 4 | **Yes** |
| (3,3) | P5 | (2,2) | 1 | No |
| (8,8) | P6 | (8,9), (9,8), (9,9) | 3 | **Yes** |
| (8,9) | P7 | (8,8), (9,8), (9,9) | 3 | **Yes** |
| (9,8) | P8 | (8,8), (8,9), (9,9) | 3 | **Yes** |
| (9,9) | P9 | (8,8), (8,9), (9,8), (10,10) | 4 | **Yes** |
| (10,10) | P10 | (9,9) | 1 | No |



### Step 3: Expand to P3 (2,1)

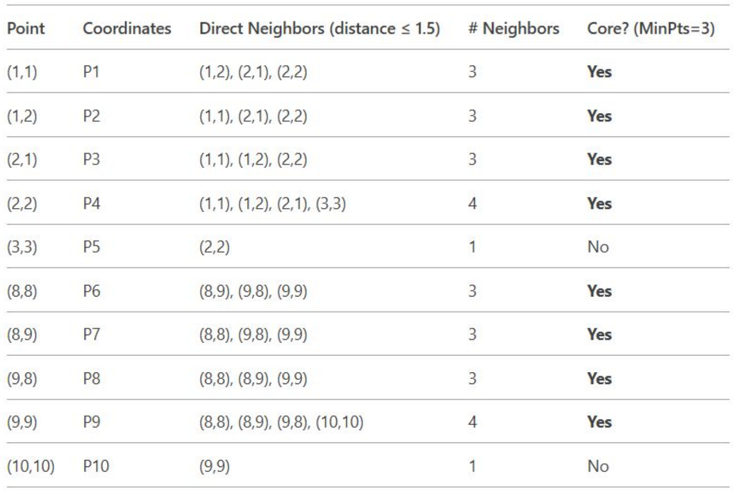
- **P3 (2,1) is a Core Point** (has 3 neighbors: **(1,1), (1,2), (2,2)**)
- **Add P3 to Cluster C1**
- **New Cluster C1 = {P1 (1,1), P2 (1,2), P3 (2,1)}**
- **Expand to its neighbors (already in C1, so stop expanding).**

---

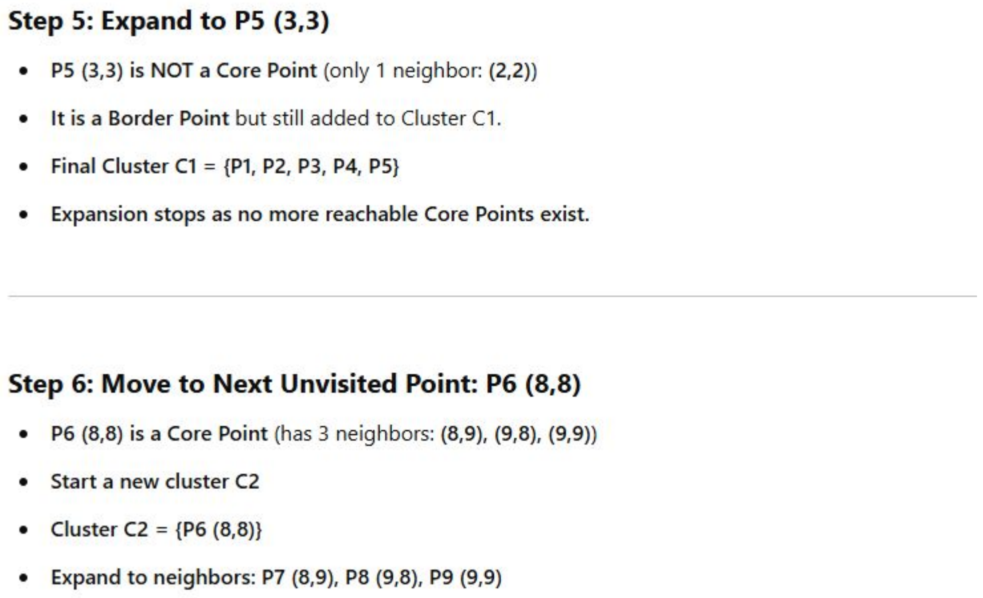
### Step 4: Expand to P4 (2,2)

- **P4 (2,2) is a Core Point** (has 4 neighbors: **(1,1), (1,2), (2,1), (3,3)**)
- **Add P4 to Cluster C1**
- **New Cluster C1 = {P1 (1,1), P2 (1,2), P3 (2,1), P4 (2,2)}**
- **Expand to its neighbors (only P5 (3,3) is left).**

# Example:DBSCAN



| Point | Coordinates | Direct Neighbors (distance ≤ 1.5) | # Neighbors | Core? (MinPts=3) |
|---|---|---|---|---|
| (1,1) | P1 | (1,2), (2,1), (2,2) | 3 | **Yes** |
| (1,2) | P2 | (1,1), (2,1), (2,2) | 3 | **Yes** |
| (2,1) | P3 | (1,1), (1,2), (2,2) | 3 | **Yes** |
| (2,2) | P4 | (1,1), (1,2), (2,1), (3,3) | 4 | **Yes** |
| (3,3) | P5 | (2,2) | 1 | No |
| (8,8) | P6 | (8,9), (9,8), (9,9) | 3 | **Yes** |
| (8,9) | P7 | (8,8), (9,8), (9,9) | 3 | **Yes** |
| (9,8) | P8 | (8,8), (8,9), (9,9) | 3 | **Yes** |
| (9,9) | P9 | (8,8), (8,9), (9,8), (10,10) | 4 | **Yes** |
| (10,10) | P10 | (9,9) | 1 | No |



### Step 5: Expand to P5 (3,3)

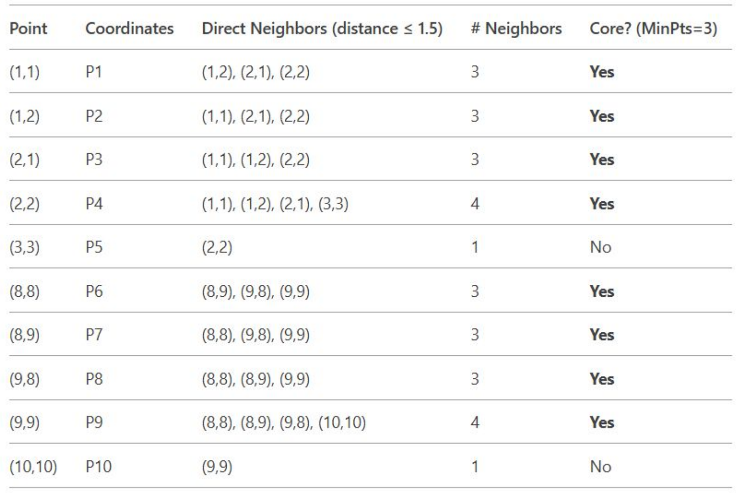
- **P5 (3,3) is NOT a Core Point** (only 1 neighbor: **(2,2)**)
- **It is a Border Point** but still added to Cluster C1.
- **Final Cluster C1 = {P1, P2, P3, P4, P5}**
- **Expansion stops as no more reachable Core Points exist.**

---

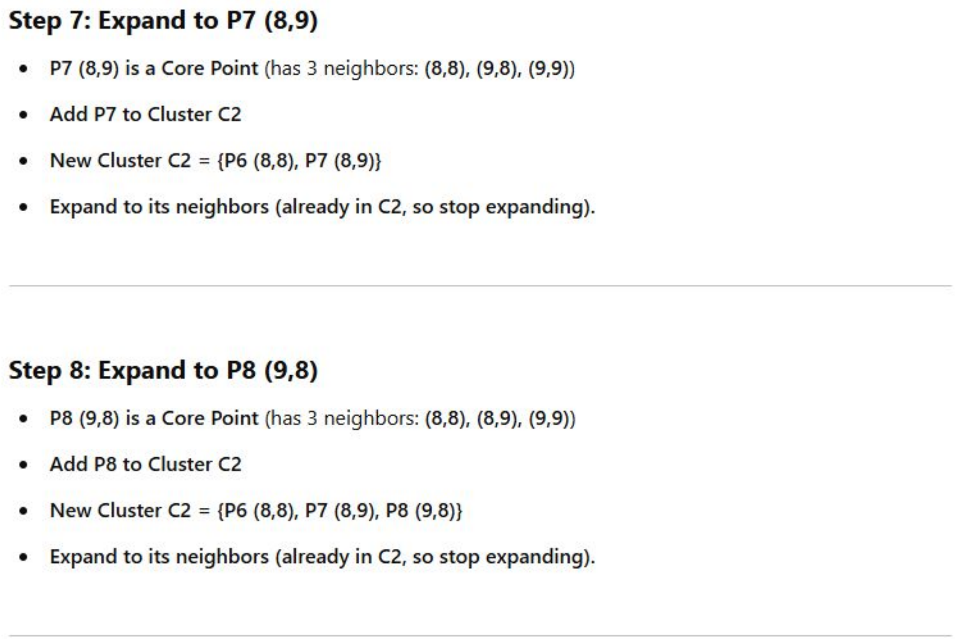
### Step 6: Move to Next Unvisited Point: P6 (8,8)

- **P6 (8,8) is a Core Point** (has 3 neighbors: **(8,9)**, **(9,8)**, **(9,9)**)
- **Start a new cluster C2**
- **Cluster C2 = {P6 (8,8)}**
- **Expand to neighbors: P7 (8,9), P8 (9,8), P9 (9,9)**

# Example:DBSCAN



| Point | Coordinates | Direct Neighbors (distance ≤ 1.5) | # Neighbors | Core? (MinPts=3) |
|---|---|---|---|---|
| (1,1) | P1 | (1,2), (2,1), (2,2) | 3 | **Yes** |
| (1,2) | P2 | (1,1), (2,1), (2,2) | 3 | **Yes** |
| (2,1) | P3 | (1,1), (1,2), (2,2) | 3 | **Yes** |
| (2,2) | P4 | (1,1), (1,2), (2,1), (3,3) | 4 | **Yes** |
| (3,3) | P5 | (2,2) | 1 | No |
| (8,8) | P6 | (8,9), (9,8), (9,9) | 3 | **Yes** |
| (8,9) | P7 | (8,8), (9,8), (9,9) | 3 | **Yes** |
| (9,8) | P8 | (8,8), (8,9), (9,9) | 3 | **Yes** |
| (9,9) | P9 | (8,8), (8,9), (9,8), (10,10) | 4 | **Yes** |
| (10,10) | P10 | (9,9) | 1 | No |



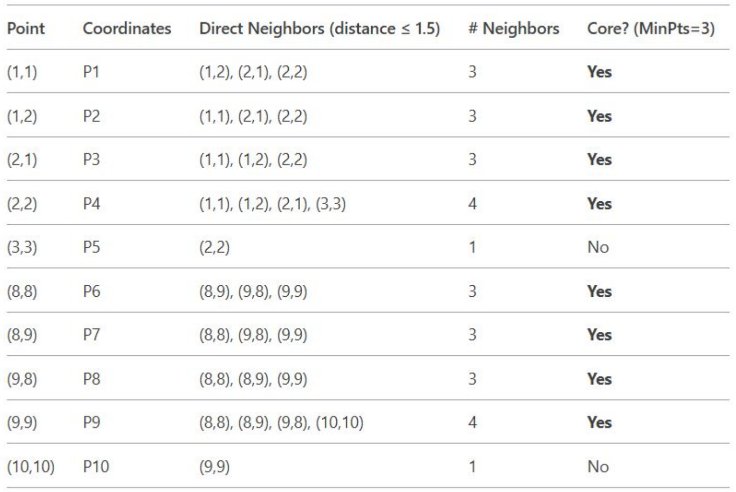
### Step 7: Expand to P7 (8,9)

- **P7 (8,9) is a Core Point** (has 3 neighbors: **(8,8), (9,8), (9,9)**)
- **Add P7 to Cluster C2**
- **New Cluster C2 = {P6 (8,8), P7 (8,9)}**
- **Expand to its neighbors (already in C2, so stop expanding).**

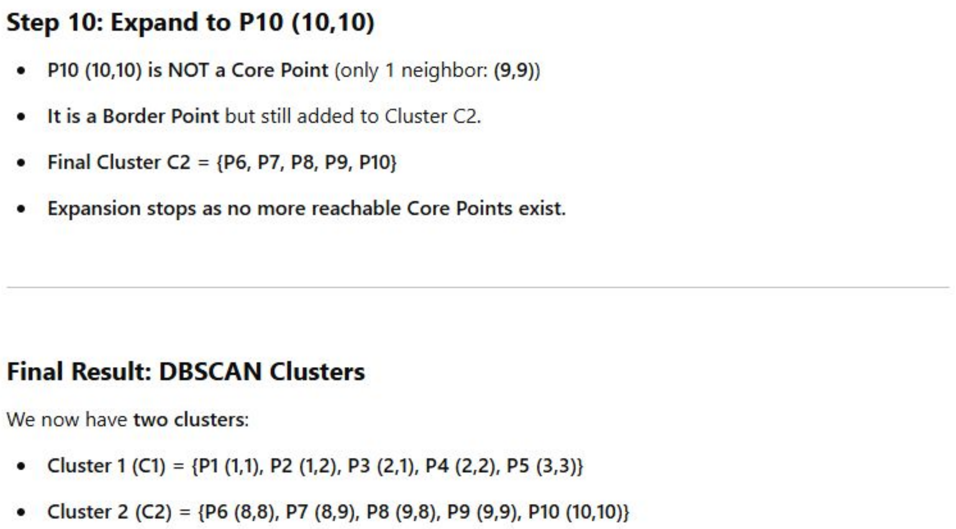
### Step 8: Expand to P8 (9,8)

- **P8 (9,8) is a Core Point** (has 3 neighbors: **(8,8), (8,9), (9,9)**)
- **Add P8 to Cluster C2**
- **New Cluster C2 = {P6 (8,8), P7 (8,9), P8 (9,8)}**
- **Expand to its neighbors (already in C2, so stop expanding).**

# Example:DBSCAN



| Point | Coordinates | Direct Neighbors (distance ≤ 1.5) | # Neighbors | Core? (MinPts=3) |
|---|---|---|---|---|
| (1,1) | P1 | (1,2), (2,1), (2,2) | 3 | **Yes** |
| (1,2) | P2 | (1,1), (2,1), (2,2) | 3 | **Yes** |
| (2,1) | P3 | (1,1), (1,2), (2,2) | 3 | **Yes** |
| (2,2) | P4 | (1,1), (1,2), (2,1), (3,3) | 4 | **Yes** |
| (3,3) | P5 | (2,2) | 1 | No |
| (8,8) | P6 | (8,9), (9,8), (9,9) | 3 | **Yes** |
| (8,9) | P7 | (8,8), (9,8), (9,9) | 3 | **Yes** |
| (9,8) | P8 | (8,8), (8,9), (9,9) | 3 | **Yes** |
| (9,9) | P9 | (8,8), (8,9), (9,8), (10,10) | 4 | **Yes** |
| (10,10) | P10 | (9,9) | 1 | No |



### Step 10: Expand to P10 (10,10)

- **P10 (10,10) is NOT a Core Point** (only 1 neighbor: **(9,9)**)
- **It is a Border Point** but still added to Cluster C2.
- **Final Cluster C2 = {P6, P7, P8, P9, P10}**
- **Expansion stops as no more reachable Core Points exist.**

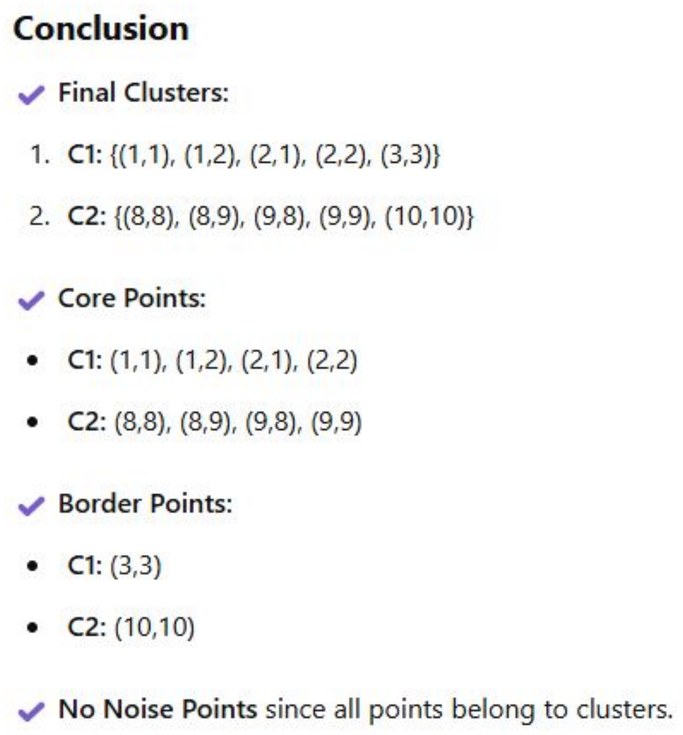
---

### Final Result: DBSCAN Clusters

We now have **two clusters:**

- **Cluster 1 (C1) = {P1 (1,1), P2 (1,2), P3 (2,1), P4 (2,2), P5 (3,3)}**
- **Cluster 2 (C2) = {P6 (8,8), P7 (8,9), P8 (9,8), P9 (9,9), P10 (10,10)}**

# Example:DBSCAN



### Conclusion

✓ Final Clusters:

1. **C1:** {(1,1), (1,2), (2,1), (2,2), (3,3)}
2. **C2:** {(8,8), (8,9), (9,8), (9,9), (10,10)}

✓ Core Points:

- **C1:** (1,1), (1,2), (2,1), (2,2)
- **C2:** (8,8), (8,9), (9,8), (9,9)

✓ Border Points:

- **C1:** (3,3)
- **C2:** (10,10)

✓ **No Noise Points** since all points belong to clusters.

# Assignment-10 (Cs-101- 2024) (Week-10)

Source

## Question-1

01:00

In a clustering evaluation, a cluster C contains 50 data points. Of these, 30 belong to class A, 15 to class B, and 5 to class C. What is the purity of this cluster?

a) 0.5

b) 0.6

c) 0.7

d) 0.8

## Question-1- Correct answer

In a clustering evaluation, a cluster C contains 50 data points. Of these, 30 belong to class A, 15 to class B, and 5 to class C. What is the purity of this cluster?

a) 0.5
b) 0.6
c) 0.7
d) 0.8

**Correct options: (b)-Purity = (Number of data points in the most frequent class) / (Total number of data points)**

## Question-2

01:00

Consider the following 2D dataset with 10 points:

(1, 1),(1, 2),(2, 1),(2, 2),(3, 3),(8, 8),(8, 9),(9, 8),(9, 9),(10, 10)

Using DBSCAN with $\epsilon = 1.5$ and MinPts = 3, how many core points are there in this dataset?

a) 4
b) 5
c) 8
d) 10

# Question-2-Explanation

| Point | (1,1) | (1,2) | (2,1) | (2,2) | (3,3) | (8,8) | (8,9) | (9,8) | (9,9) | (10,10) |
|---|---|---|---|---|---|---|---|---|---|---|
| **(1,1)** | 0 | 1✅ | 1✅ | 1.41✅ | 2.83❌ | 9.90❌ | 10.63❌ | 10.63❌ | 11.31❌ | 12.73❌ |
| **(1,2)** | 1✅ | 0 | 1.41✅ | 1✅ | 2.23❌ | 9.22❌ | 9.90❌ | 9.90❌ | 10.63❌ | 12.08❌ |
| **(2,1)** | 1✅ | 1.41✅ | 0 | 1✅ | 2.23❌ | 9.22❌ | 9.90❌ | 9.90❌ | 10.63❌ | 12.08❌ |
| **(2,2)** | 1.41✅ | 1✅ | 1✅ | 0 | 1.41✅ | 8.48❌ | 9.22❌ | 9.22❌ | 9.90❌ | 11.31❌ |
| **(3,3)** | 2.83❌ | 2.23❌ | 2.23❌ | 1.41✅ | 0 | 7.07❌ | 7.81❌ | 7.81❌ | 8.48❌ | 9.90❌ |
| **(8,8)** | 9.90❌ | 9.22❌ | 9.22❌ | 8.48❌ | 7.07❌ | 0 | 1✅ | 1✅ | 1.41✅ | 2.83❌ |
| **(8,9)** | 10.63❌ | 9.90❌ | 9.90❌ | 9.22❌ | 7.81❌ | 1✅ | 0 | 1.41✅ | 1✅ | 2.23❌ |
| **(9,8)** | 10.63❌ | 9.90❌ | 9.90❌ | 9.22❌ | 7.81❌ | 1✅ | 1.41✅ | 0 | 1✅ | 2.23❌ |
| **(9,9)** | 11.31❌ | 10.63❌ | 10.63❌ | 9.90❌ | 8.48❌ | 1.41✅ | 1✅ | 1✅ | 0 | 1.41✅ |
| **(10,10)** | 12.73❌ | 12.08❌ | 12.08❌ | 11.31❌ | 9.90❌ | 2.83❌ | 2.23❌ | 2.23❌ | 1.41✅ | 0 |

# Question-2- Correct answer

Consider the following 2D dataset with 10 points (1, 1),(1, 2),(2, 1),(2, 2),(3, 3),(8, 8),(8, 9),(9, 8),(9, 9),(10, 10)

Using DBSCAN with $\epsilon = 1.5$ and MinPts = 3, how many core points are there in this dataset?

a) 4

b) 5

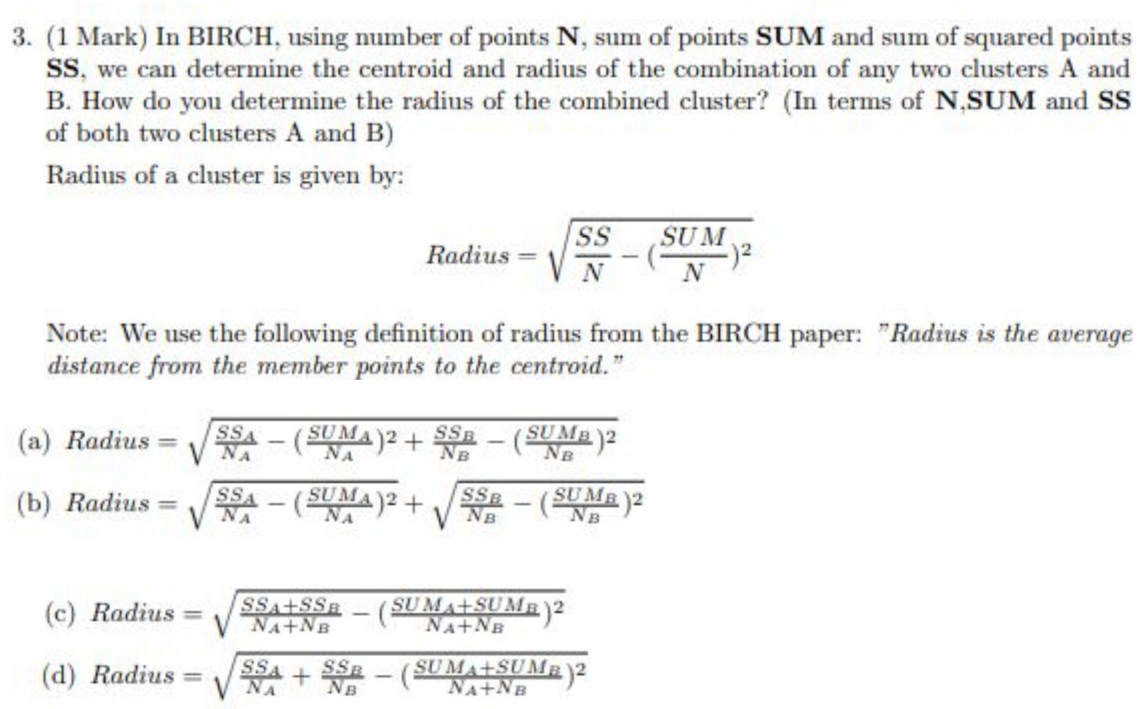
c) 8

d) 10

**Correct options: (c) To be a core point, it needs at least 3 points (including itself) within $\epsilon = 1.5$ distance. There are 8 core points: (1,1), (1,2), (2,1), (2,2) from first group and (8,8), (8,9), (9,8), (9,9) from second group.**

## Question-3

01:00



3. (1 Mark) In BIRCH, using number of points **N**, sum of points **SUM** and sum of squared points **SS**, we can determine the centroid and radius of the combination of any two clusters A and B. How do you determine the radius of the combined cluster? (In terms of **N**,**SUM** and **SS** of both two clusters A and B)

Radius of a cluster is given by:

$$Radius = \sqrt{\frac{SS}{N} - (\frac{SUM}{N})^2}$$

Note: We use the following definition of radius from the BIRCH paper: *"Radius is the average distance from the member points to the centroid."*

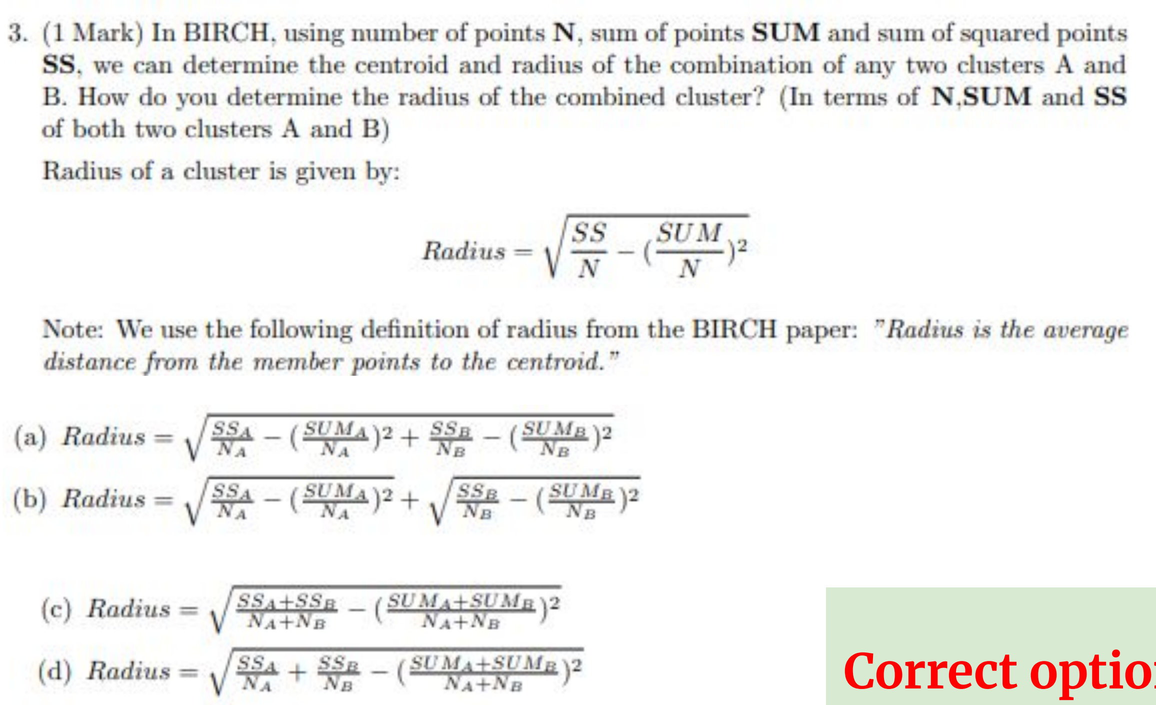
(a) $Radius = \sqrt{\frac{SS_A}{N_A} - (\frac{SUM_A}{N_A})^2 + \frac{SS_B}{N_B} - (\frac{SUM_B}{N_B})^2}$

(b) $Radius = \sqrt{\frac{SS_A}{N_A} - (\frac{SUM_A}{N_A})^2} + \sqrt{\frac{SS_B}{N_B} - (\frac{SUM_B}{N_B})^2}$

(c) $Radius = \sqrt{\frac{SS_A+SS_B}{N_A+N_B} - (\frac{SUM_A+SUM_B}{N_A+N_B})^2}$

(d) $Radius = \sqrt{\frac{SS_A}{N_A} + \frac{SS_B}{N_B} - (\frac{SUM_A+SUM_B}{N_A+N_B})^2}$

# Question-3 - Correct answer



3. (1 Mark) In BIRCH, using number of points **N**, sum of points **SUM** and sum of squared points **SS**, we can determine the centroid and radius of the combination of any two clusters A and B. How do you determine the radius of the combined cluster? (In terms of **N**,**SUM** and **SS** of both two clusters A and B)

Radius of a cluster is given by:

$$Radius = \sqrt{\frac{SS}{N} - (\frac{SUM}{N})^2}$$

Note: We use the following definition of radius from the BIRCH paper: *"Radius is the average distance from the member points to the centroid."*

(a) $Radius = \sqrt{\frac{SS_A}{N_A} - (\frac{SUM_A}{N_A})^2 + \frac{SS_B}{N_B} - (\frac{SUM_B}{N_B})^2}$

(b) $Radius = \sqrt{\frac{SS_A}{N_A} - (\frac{SUM_A}{N_A})^2} + \sqrt{\frac{SS_B}{N_B} - (\frac{SUM_B}{N_B})^2}$

(c) $Radius = \sqrt{\frac{SS_A+SS_B}{N_A+N_B} - (\frac{SUM_A+SUM_B}{N_A+N_B})^2}$

(d) $Radius = \sqrt{\frac{SS_A}{N_A} + \frac{SS_B}{N_B} - (\frac{SUM_A+SUM_B}{N_A+N_B})^2}$

Correct options: (c)

## Question-4

03:00

Which of the following properties are TRUE?

a) Using the CURE algorithm can lead to non-convex clusters.
b) K-means scales better than CURE for large datasets.
c) CURE is a simplification of K-means and hence scales better than k-means for large datasets.
d) K-means being more expensive to run on large datasets, can give non-convex clusters too.

## Question-4 - Correct answer

Which of the following properties are TRUE?

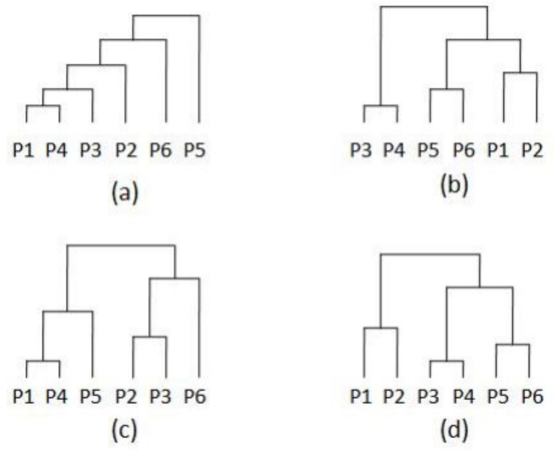
a) Using the CURE algorithm can lead to non-convex clusters.
b) K-means scales better than CURE for large datasets.
c) CURE is a simplification of K-means and hence scales better than k-means for large datasets.
d) K-means being more expensive to run on large datasets, can give non-convex clusters too.

Correct options: (a)

## Question-5

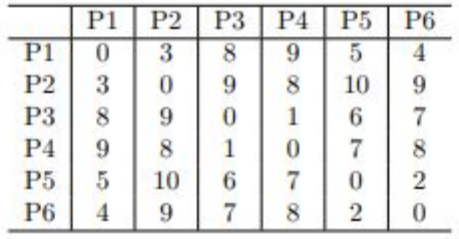
01:00

The pairwise distance between 6 points is given below. Which of the option shows the hierarchy of clusters created by single link clustering algorithm?



|  | P1 | P2 | P3 | P4 | P5 | P6 |
|---|---|---|---|---|---|---|
| P1 | 0 | 3 | 8 | 9 | 5 | 4 |
| P2 | 3 | 0 | 9 | 8 | 10 | 9 |
| P3 | 8 | 9 | 0 | 1 | 6 | 7 |
| P4 | 9 | 8 | 1 | 0 | 7 | 8 |
| P5 | 5 | 10 | 6 | 7 | 0 | 2 |
| P6 | 4 | 9 | 7 | 8 | 2 | 0 |

# Question-5-Explanation



| | P1 | P2 | P3 | P4 | P5 | P6 |
|---|---|---|---|---|---|---|
| P1 | 0 | 3 | 8 | 9 | 5 | 4 |
| P2 | 3 | 0 | 9 | 8 | 10 | 9 |
| P3 | 8 | 9 | 0 | 1 | 6 | 7 |
| P4 | 9 | 8 | 1 | 0 | 7 | 8 |
| P5 | 5 | 10 | 6 | 7 | 0 | 2 |
| P6 | 4 | 9 | 7 | 8 | 2 | 0 |

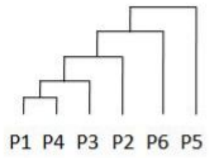
Step 1: Connect closest pair of points. Closest pairs are:
**[C1] d(P3, P4) = 1 ,      [C2] d(P5, P6) = 2,      [C3] d(P1, P2) = 3**

Step 2: Connect clusters with single link. The cluster pair to combine is bolded:
d(C3,C1) = 8      **[C4] d(C3, C2) = 4**      d(C2, C1) = 6

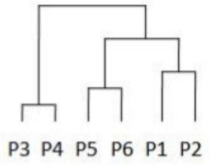
Step 3: Connect the final 2 clusters

## Question-5 - Correct answer

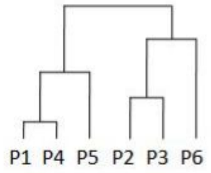
The pairwise distance between 6 points is given below. Which of the option shows the hierarchy of clusters created by single link clustering algorithm?



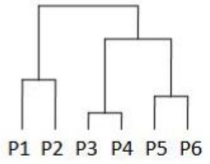
(a)



(b)



(c)



(d)

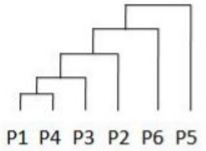
| | P1 | P2 | P3 | P4 | P5 | P6 |
|---|---|---|---|---|---|---|
| P1 | 0 | 3 | 8 | 9 | 5 | 4 |
| P2 | 3 | 0 | 9 | 8 | 10 | 9 |
| P3 | 8 | 9 | 0 | 1 | 6 | 7 |
| P4 | 9 | 8 | 1 | 0 | 7 | 8 |
| P5 | 5 | 10 | 6 | 7 | 0 | 2 |
| P6 | 4 | 9 | 7 | 8 | 2 | 0 |

**Correct options: (b)**

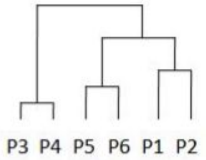
## Question-6

01:00

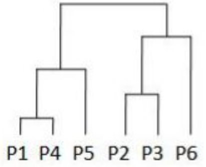
For the pairwise distance matrix given in the previous question, which of the following shows the hierarchy of clusters created by the complete link clustering algorithm.



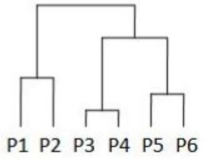
(a)



(b)



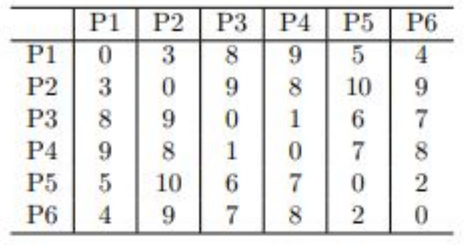
(c)



(d)

| | P1 | P2 | P3 | P4 | P5 | P6 |
|---|---|---|---|---|---|---|
| P1 | 0 | 3 | 8 | 9 | 5 | 4 |
| P2 | 3 | 0 | 9 | 8 | 10 | 9 |
| P3 | 8 | 9 | 0 | 1 | 6 | 7 |
| P4 | 9 | 8 | 1 | 0 | 7 | 8 |
| P5 | 5 | 10 | 6 | 7 | 0 | 2 |
| P6 | 4 | 9 | 7 | 8 | 2 | 0 |

# Question-6-Explanation



| | P1 | P2 | P3 | P4 | P5 | P6 |
|---|---|---|---|---|---|---|
| P1 | 0 | 3 | 8 | 9 | 5 | 4 |
| P2 | 3 | 0 | 9 | 8 | 10 | 9 |
| P3 | 8 | 9 | 0 | 1 | 6 | 7 |
| P4 | 9 | 8 | 1 | 0 | 7 | 8 |
| P5 | 5 | 10 | 6 | 7 | 0 | 2 |
| P6 | 4 | 9 | 7 | 8 | 2 | 0 |

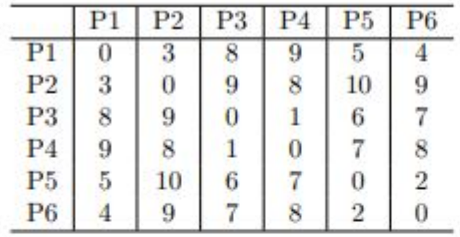
Step 1: Connect closest pair of points. Closest pairs are:
**[C1] d(P3, P4) = 1 ,      [C2] d(P5, P6) = 2,      [C3] d(P1, P2) = 3**

Step 2: Connect clusters with complete link. The cluster pair to combine is bolded:
d(C3,C1) = 9      [C4] d(C3, C2) = 10      **d(C2, C1) = 8**

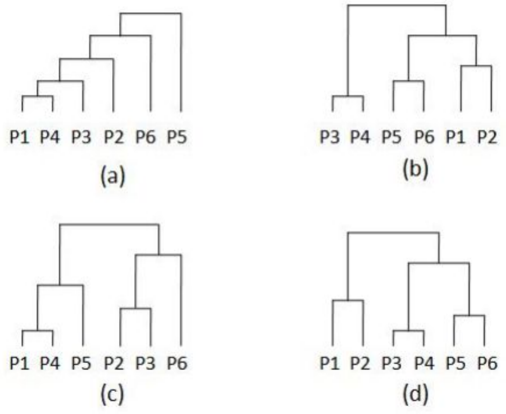
Step 3: Connect the final 2 clusters

# Question-6 - Correct answer

For the pairwise distance matrix given in the previous question, which of the following shows the hierarchy of clusters created by the complete link clustering algorithm.



| | P1 | P2 | P3 | P4 | P5 | P6 |
|---|---|---|---|---|---|---|
| P1 | 0 | 3 | 8 | 9 | 5 | 4 |
| P2 | 3 | 0 | 9 | 8 | 10 | 9 |
| P3 | 8 | 9 | 0 | 1 | 6 | 7 |
| P4 | 9 | 8 | 1 | 0 | 7 | 8 |
| P5 | 5 | 10 | 6 | 7 | 0 | 2 |
| P6 | 4 | 9 | 7 | 8 | 2 | 0 |



**Correct options: (d)**

## THANK YOU

## Suggestions and Feedback

**Next Session:**

**Friday: 03-Oct-2025**
**2:00 - 4:00 PM**