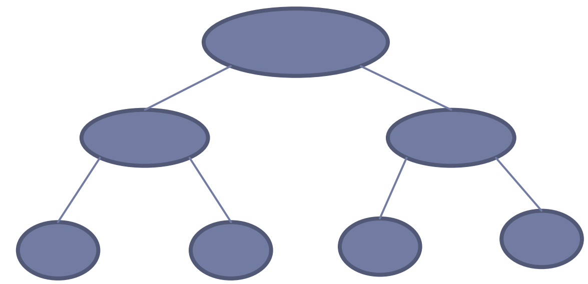
Chapter 3: Cluster Analysis

- **3.1 Basic Concepts of Clustering**
- **3.2 Partitioning Methods**
- **3.3 Hierarchical Methods**
  - 3.3.1 The Principle
  - 3.3.2 Agglomerative and Divisive Clustering
  - 3.3.3 BIRCH
  - 3.3.4 Rock
- **3.4 Density-based Methods**
  - 3.4.1 The Principle
  - 3.4.2 DBSCAN
  - 3.4.3 OPTICS
- **3.5 Clustering High-Dimensional Data**
- **3.6 Outlier Analysis**

3.3.1 The Principle

- Group data objects into a tree of clusters

- Hierarchical methods can be
  - → **Agglomerative**: bottom-up approach
  - → **Divisive**: top-down approach



- Hierarchical clustering has **no backtracking**
  - → If a particular merge or split turns out to be poor choice, it **cannot be corrected**

3.3.2 Agglomerative and Divisive

Agglomerative Hierarchical Clustering

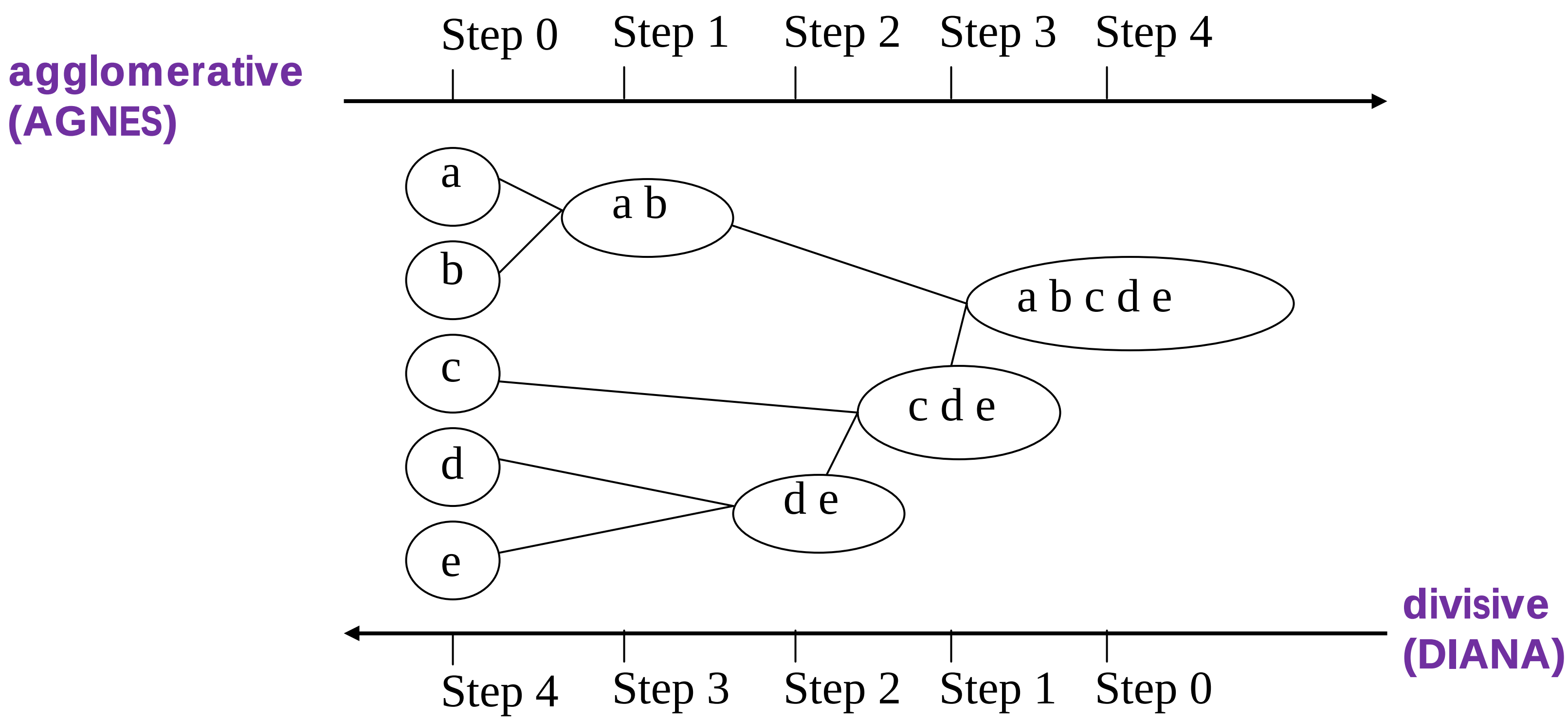
- Bottom-up strategy
- Each cluster starts with only one object
- Clusters are merged into larger and larger clusters until:
  - → All the objects are in a single cluster
  - → Certain termination conditions are satisfied

Divisive Hierarchical Clustering

- Top-down strategy
- Start with all objects in one cluster
- Clusters are subdivided into smaller and smaller clusters until:
  - → Each object forms a cluster on its own
  - → Certain termination conditions are satisfied

Example

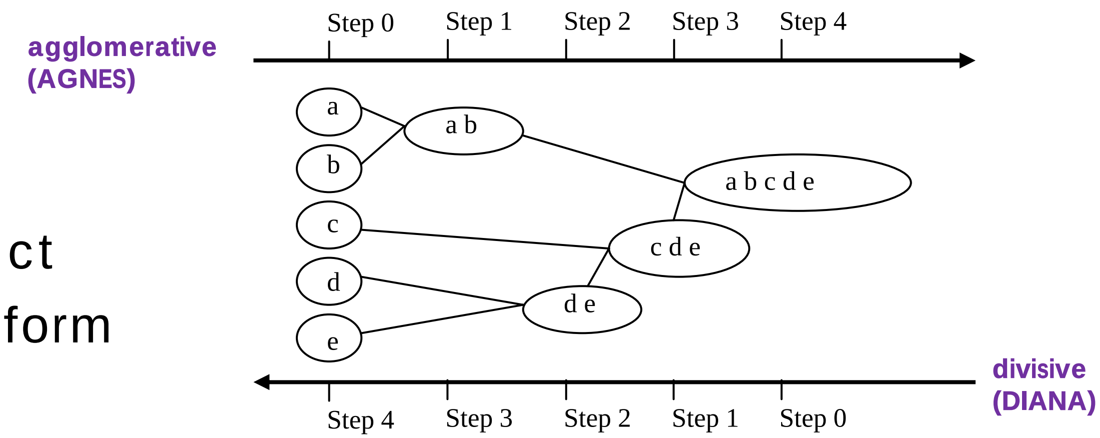
- Agglomerative and divisive algorithms on a data set of five objects **{a, b, c, d, e}**



Example

- **AGNES**

  → Clusters C1 and C2 may be merged if an object in C1 and an object in C2 form the minimum Euclidean distance between any two objects from different clusters



- **DIANA**

  → A cluster is split according to some principle, e.g., the maximum Euclidian distance between the closest neighboring objects in the cluster

Distance Between Clusters

- First measure: **Minimum distance**

$$d_{\min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} |p - p'|$$

  - → **|p-p’|** is the distance between two objects p and p’
- **Use cases**
  - → An algorithm that uses the minimum distance to measure the distance between clusters is called sometimes **nearest-neighbor clustering algorithm**
  - → If the clustering process terminates when the minimum distance between nearest clusters exceeds an arbitrary threshold, it is called **single-linkage algorithm**
  - → An agglomerative algorithm that uses the minimum distance measure is also called **minimal spanning tree algorithm**

Distance Between Clusters

- Second measure: **Maximum distance**

$$d_{\max}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} |p - p'|$$

  - → **|p-p'|** is the distance between two objects p and p'
- **Use cases**
  - → An algorithm that uses the maximum distance to measure the distance between clusters is called sometimes **farthest-neighbor clustering algorithm**
  - → If the clustering process terminates when the maximum distance between nearest clusters exceeds an arbitrary threshold, it is called **complete-linkage algorithm**

Distance Between Clusters

- Minimum and maximum distances are extreme implying that they are overly sensitive to outliers or noisy data
- Third measure: **Mean distance**

$$d_{mean}(C_i, C_j) = |m_i - m_j|$$

  - → $\mathbf{m_i}$ and $\mathbf{m_j}$ are the means for cluster C_i and C_j respectively
- Fourth measure: **Average distance**

$$d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i} \sum_{p' \in C_j} |p - p'|$$

  - → **|p-p'|** is the distance between two objects p and p'
  - → $\mathbf{n_i}$ and $\mathbf{n_j}$ are the number of objects in cluster C_i and C_j respectively
- → Mean is difficult to compute for categorical data

Challenges & Solutions

- It is **difficult** to select merge or split points
- No **backtracking**
- Hierarchical clustering **does not scale** well: examines a good number of objects before any decision of split or merge
- One promising directions to solve these problems is to combine hierarchical clustering with other clustering techniques: **multiple-phase clustering**

3.3.3 BIRCH

- **BIRCH**: Balanced Iterative Reducing and Clustering Using Hierarchies
- **Agglomerative** Clustering designed for clustering a **large** amount of numerical **data**

- What Birch algorithm tries to solve?

  - Most of the existing algorithms DO NOT consider the case that **datasets** can be **too large** to **fit** in main **memory**

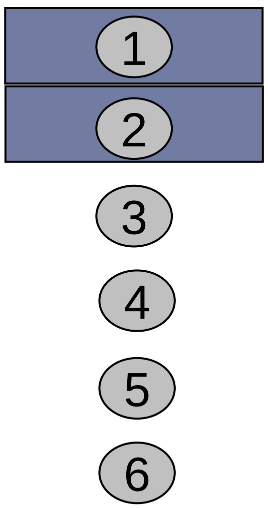
  - They DO NOT concentrate on **minimizing** the number of **scans** of the dataset

  - **I/O costs** are very high

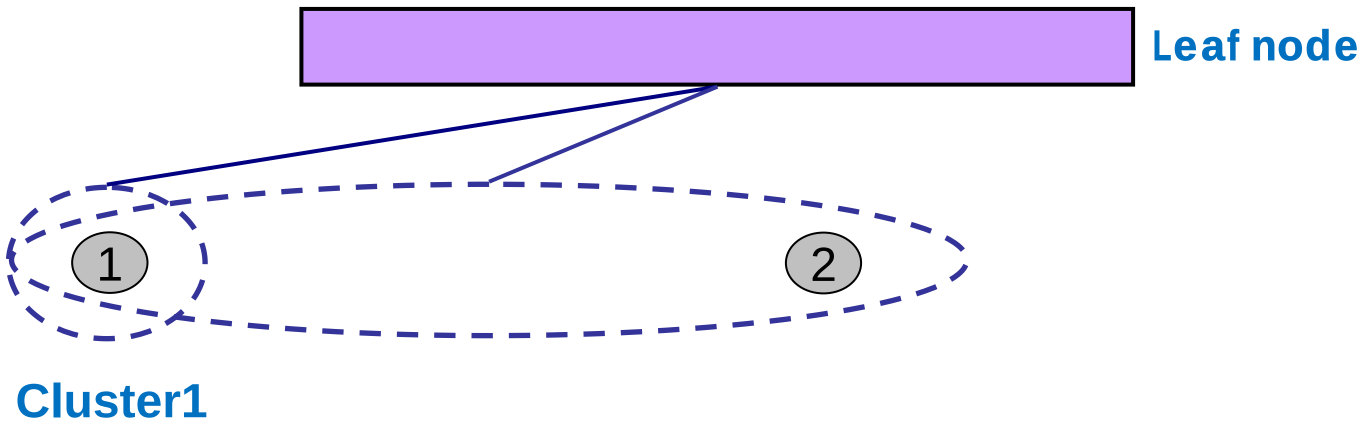
- The complexity of BIRCH is O(n) where n is the number of objects to be clustered.

BIRCH: The Idea by example

Data Objects



Clustering Process (build a tree)

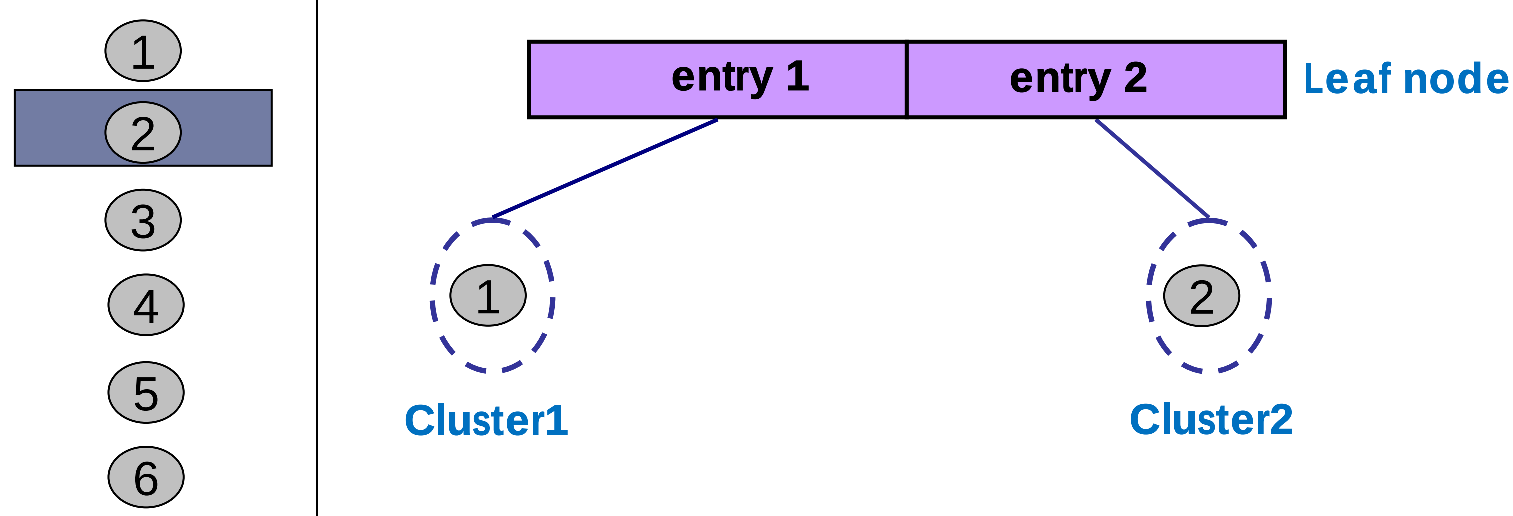


If cluster 1 becomes too large (not compact) by adding object 2, then split the cluster

BIRCH: The Idea by example

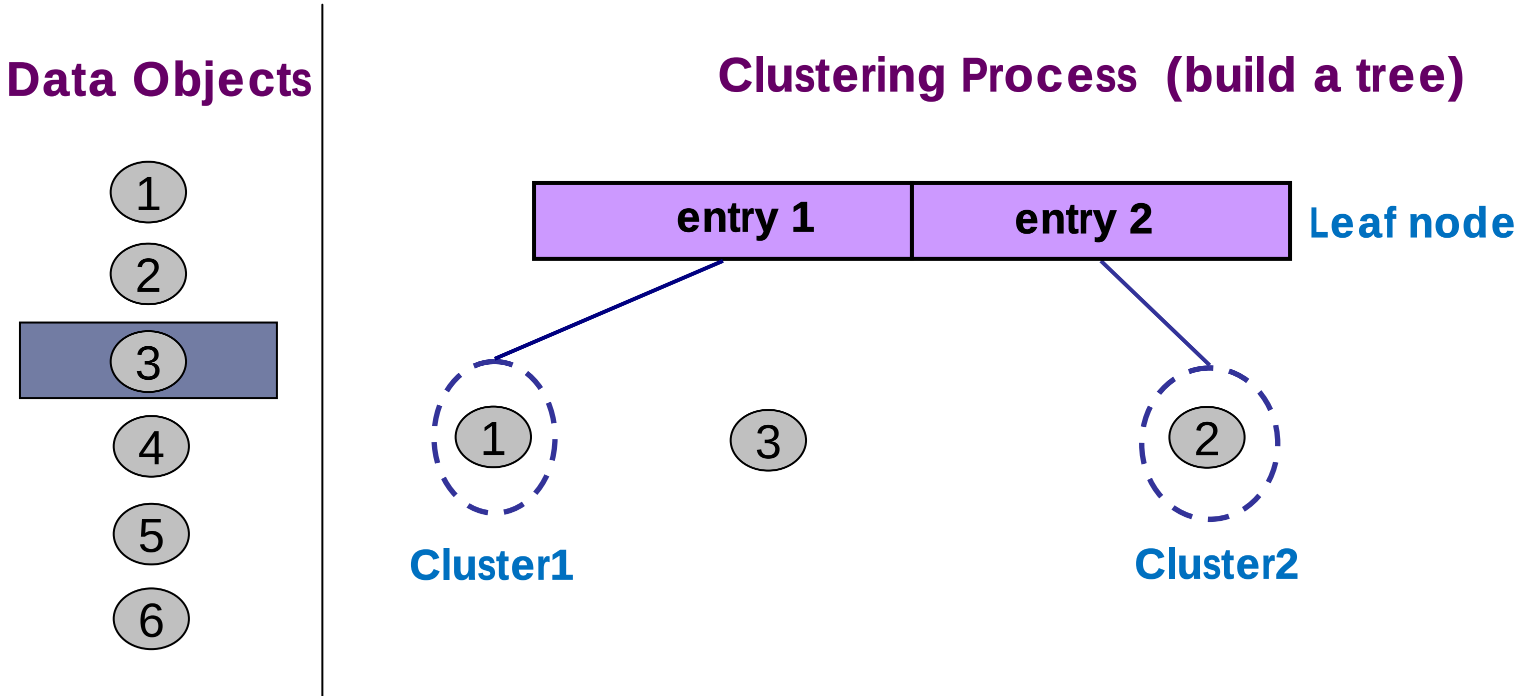
Data Objects

Clustering Process (build a tree)



Leaf node with two entries

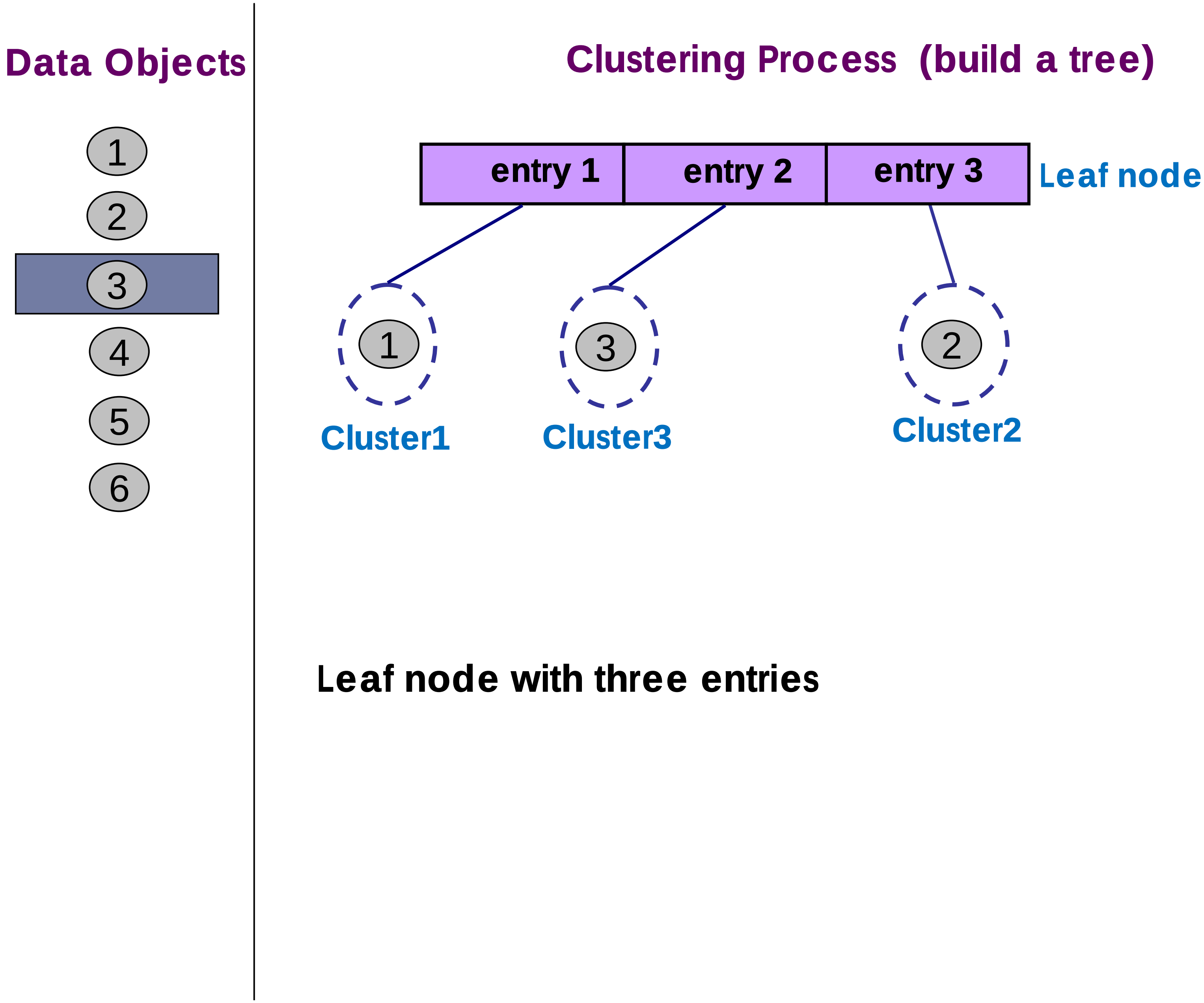
BIRCH: The Idea by example



entry1 is the closest to object 3

If cluster 1 becomes too large by adding object 3, then split the cluster

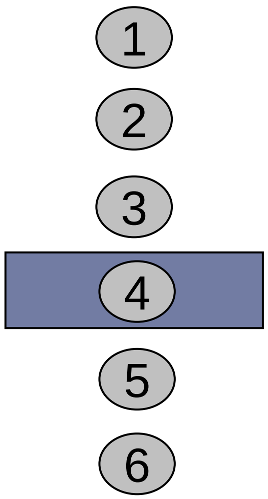
BIRCH: The Idea by example



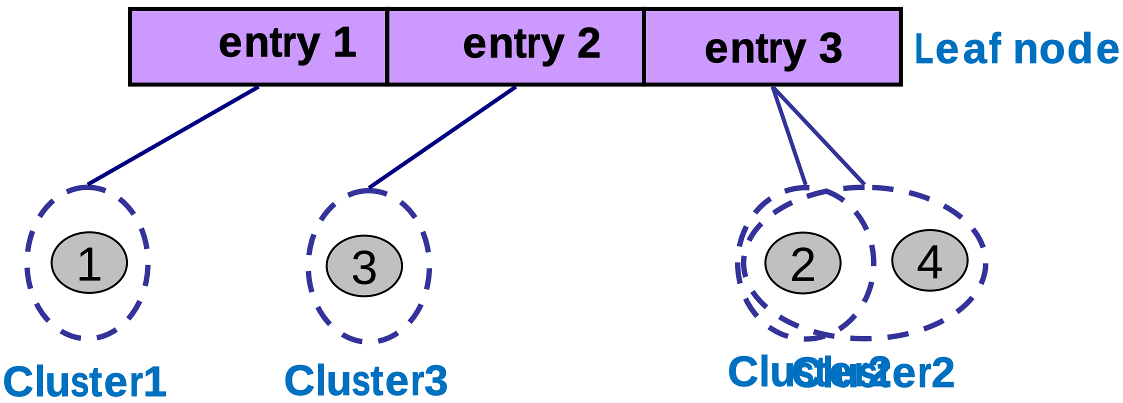
Leaf node with three entries

BIRCH: The Idea by example

Data Objects



Clustering Process (build a tree)

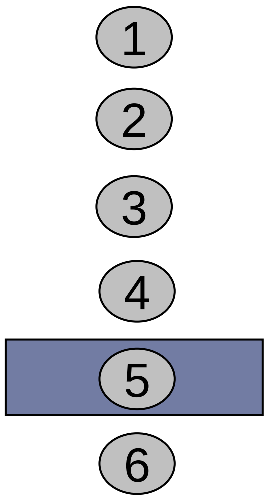


entry3 is the closest to object 4

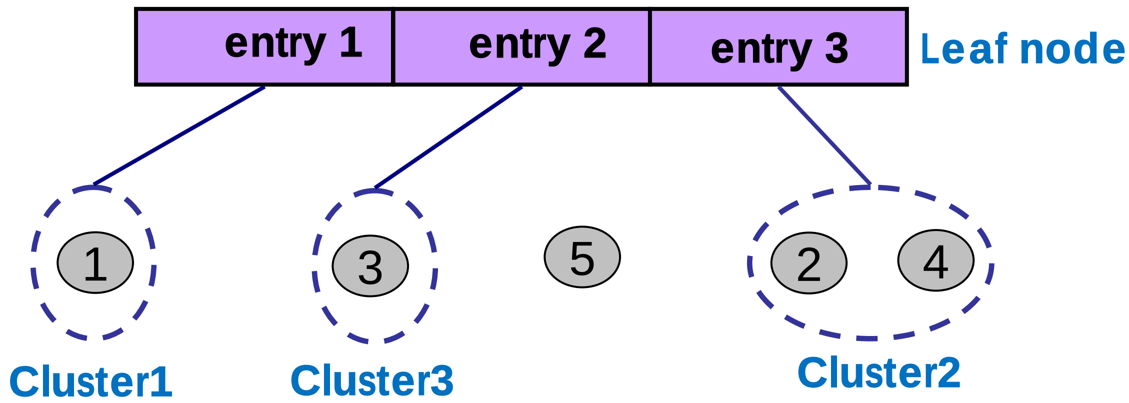
Cluster 2 remains compact when adding object 4
then add object 4 to cluster 2

BIRCH: The Idea by example

Data Objects



Clustering Process (build a tree)

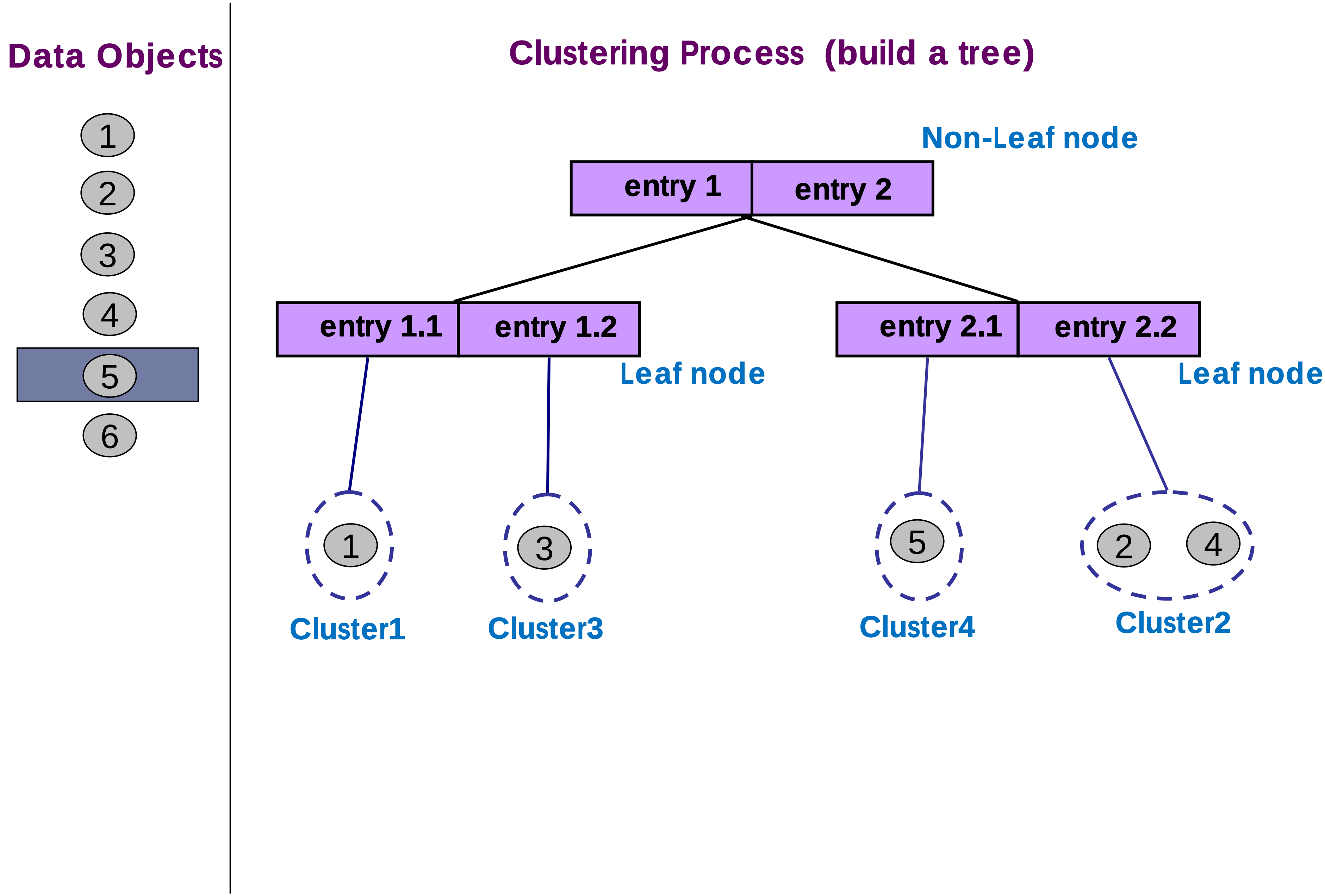


entry2 is the closest to object 5

Cluster 3 becomes too large by adding object 5
then split cluster 3?

BUT there is a limit to the number of entries a node can have
Thus, split the node

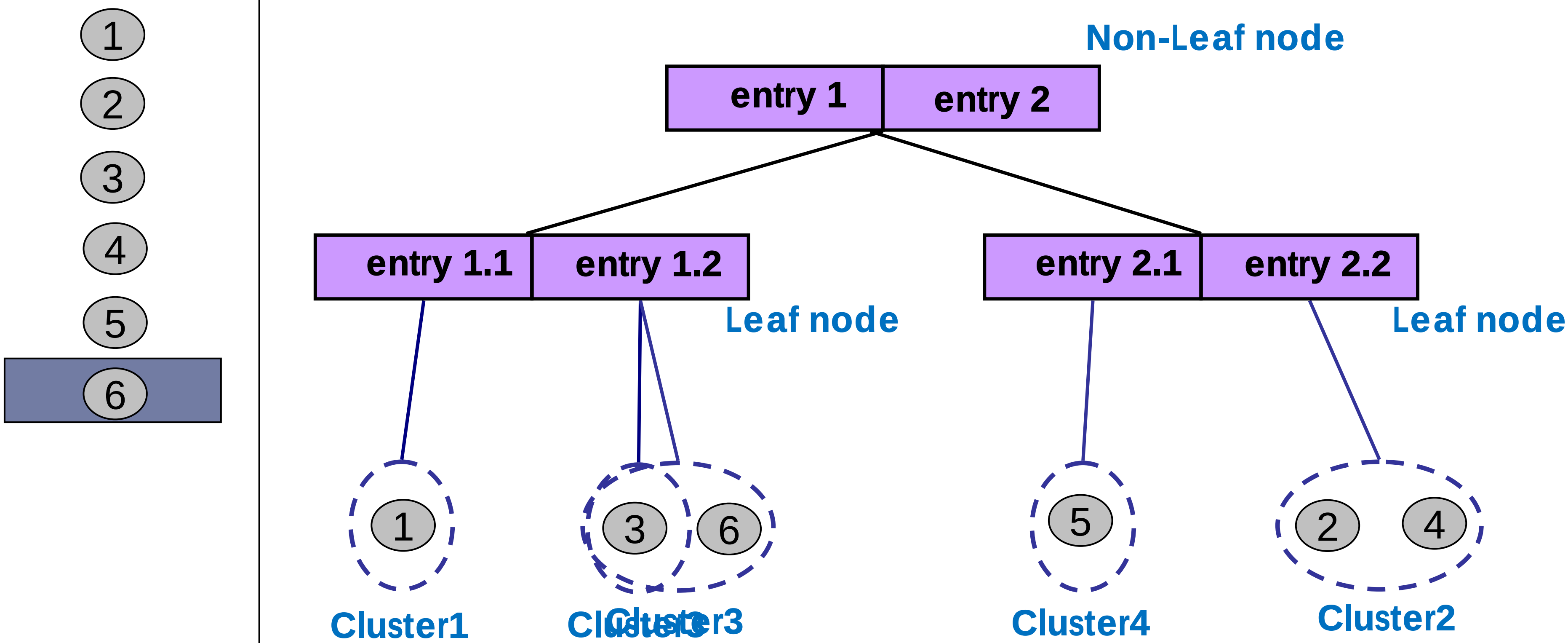
BIRCH: The Idea by example



BIRCH: The Idea by example

Data Objects

Clustering Process (build a tree)



entry1.2 is the closest to object 6

Cluster 3 remains compact when adding object 6
then add object 6 to cluster 3

BIRCH: Key Components

- **Clustering Feature (CF)**
  - → Summary of the statistics for a given cluster: the 0-th, 1st and 2nd moments of the cluster from the statistical point of view
  - → Used to compute centroids, and measure the compactness and distance of clusters
- **CF-Tree**
  - → height-balanced tree
  - → two parameters:
    - number of entries in each node
    - The *diameter* of all entries in a leaf node
  - → Leaf nodes are connected via *prev* and *next* pointers

Clustering Feature

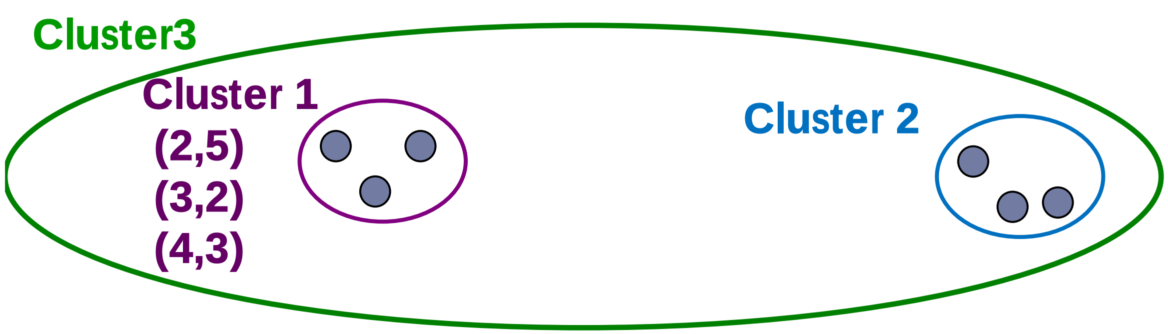
Clustering Feature (CF): CF = (N, LS, SS)

N: Number of data points

LS: linear sum of N points: $\sum_{i=1}^{N} X_i$

SS: square sum of N points: $\sum_{i=1}^{N} X_i^{\,2}$

CF3=CF1+CF2= $\langle$3+3, (9+35, 10+36), (29+417 , 38+440)$\rangle$ = $\langle$6, (44,46), (446 ,478)$\rangle$



CF_2= $\langle$3, (35,36), (417 ,440)$\rangle$

CF_1= $\langle$3, (2+3+4 , 5+2+3), ($2^2+3^2+4^2$, $5^2+2^2+3^2$)$\rangle$ = $\langle$3, (9,10), (29 ,38)$\rangle$

Properties of Clustering Feature

- **CF** entry is a **summary** of statistics of the cluster

- A **representation** of the cluster

- A CF entry has **sufficient information** to calculate the centroid, radius, diameter and many other distance measures

- **Additively** theorem allows us to **merge sub-clusters incrementally**

Distance Measures

- Given a cluster with data points ,
  - **Centroid:**

$$x_0 = \frac{\sum_{i=1}^{n} X_i}{n}$$

  - **Radius:** average distance from any point of the cluster to its centroid

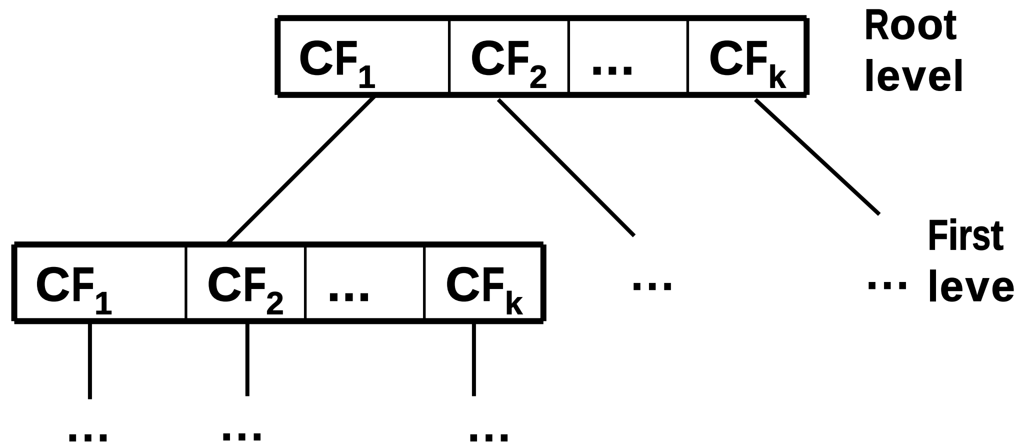
$$R = \sqrt{\frac{\sum_{i=1}^{n} (x_i - x_0)^2}{n}}$$

  - **Diameter:** square root of average mean squared distance between all pairs of points in the cluster

$$D = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{n} (x_i - x_j)^2}{n}}$$

CF Tree

- **B** = Branching Factor,
  maximum children
  in a non-leaf node
- **T** = Threshold for
  diameter or radius
  of the cluster in a leaf
- **L** = number of entries in
  a leaf
- CF entry in parent = sum of CF entries of a child of that entry



- In-memory, height-balanced tree

CF Tree Insertion

- Start with the root
- Find the CF entry in the root closest to the data point, move to that child and repeat the process until a closest leaf entry is found.
- At the leaf
  - If the point can be accommodated in the cluster, update the entry
  - If this addition violates the threshold T, split the entry, if this violates the limit imposed by L, split the leaf. If its parent node is full, split that and so on
- Update the CF entries from the leaf to the root to accommodate this point

Birch Algorithm

Data

Initial CF tree

Phase 1: Load into memory by building a CF tree

Smaller CF tree

Phase 2 (optional): Condense tree into desirable range by building a smaller CF tree

Good Clusters

Phase 3: Global Clustering

Better Clusters

Phase 4: (optional and offline): Cluster Refining

Birch Algorithm: Phase 1

- Choose an initial value for threshold, start inserting the data points one by one into the tree as per the insertion algorithm
- If, in the middle of the above step, the size of the CF tree exceeds the size of the available memory, increase the value of threshold
- Convert the partially built tree into a new tree
- Repeat the above steps until the entire dataset is scanned and a full tree is built
- Outlier Handling

Birch Algorithm: Phase 2,3, and 4

- **Phase 2**
  - → A bridge between phase 1 and phase 3
  - → Builds a smaller CF tree by increasing the threshold
- **Phase 3**
  - → Apply global clustering algorithm to the sub-clusters given by leaf entries of the CF tree
  - → Improves clustering quality
- **Phase 4**
  - → Scan the entire dataset to label the data points
  - → Outlier handling

3.3.4 ROCK: for Categorical Data

- Experiments show that distance functions do not lead to high quality clusters when clustering categorical data
- Most clustering techniques assess the similarity between points to create clusters
- At each step, points that are similar are merged into a single cluster
- Localized approach prone to errors
- ROCK: uses links instead of distances

Example: Compute Jaccard Coefficient

Transaction items: **a,b,c,d,e,f,g**

Compute Jaccard coefficient between transactions

$$sim\,(T_i, T_j)\,\frac{|\,T_i \cap T_j\,|}{|\,T_i \cup T_j\,|}$$

Sim({a,b,c},{b,d,e})=1/5=0.2

Jaccard coefficient between transactions of Cluster1 ranges from 0.2 to 0.5

Jaccard coefficient between transactions belonging to different clusters can also reach 0.5

Sim({a,b,c},{a,b,f})=2/4=0.5

Two clusters of transactions

Cluster1. <a, b, c, d, e>

{a, b, c}
{a, b, d}
{a, b, e}
{a, c, d}
{a, c, e}
{a, d, e}
{b, c, d}
{b, c, e}
{b, d, e}
{c, d, e}

Cluster2. <a, b, f, g>

{a, b, f}
{a, b, g}
{a, f, g}
{b, f, g}

Example: Using Links

Transaction items: **a,b,c,d,e,f,g**

The number of links between T_i and T_j is the number of common neighbors

T_i and T_j are neighbors if $Sim(Ti,Tj)>\theta$

Consider $\theta=0.5$

Link({a,b,f}, {a,b,g}) = 5 (common neighbors)

Link({a,b,f},{a,b,c})=3 (common neighbors)

Link is a better measure than Jaccard coefficient

Two clusters of transactions

Cluster1. <a, b, c, d, e>

{a, b, c}
{a, b, d}
{a, b, e}
{a, c, d}
{a, c, e}
{a, d, e}
{b, c, d}
{b, c, e}
{b, d, e}
{c, d, e}

Cluster2. <a, b, f, g>

{a, b, f}
{a, b, g}
{a, f, g}
{b, f, g}

ROCK

- **ROCK:** Robust Clustering using linKs
- **Major Ideas**
  - Use links to measure similarity/proximity
  - Not distance-based
  - Computational complexity $O(n^2 + n m_m m_a + n^2 \log n)$
    - **m_a**: average number of neighbors
    - **m_m**: maximum number of neighbors
    - **n**: number of objects
- **Algorithm**
  - Sampling-based clustering
  - Draw random sample
  - Cluster with links
  - Label data in disk

Chapter 3: Cluster Analysis

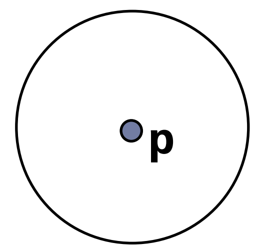
- **3.1 Basic Concepts of Clustering**
- **3.2 Partitioning Methods**
- **3.3 Hierarchical Methods**
  - 3.3.1 The Principle
  - 3.3.2 Agglomerative and Divisive Clustering
  - 3.3.3 BIRCH
  - 3.3.4 Rock
- **3.4 Density-based Methods**
  - 3.4.1 The Principle
  - 3.4.2 DBSCAN
  - 3.4.3 OPTICS
- **3.5 Clustering High-Dimensional Data**
- **3.6 Outlier Analysis**

3.4.1 The Principle

- Regard clusters as dense regions in the data space separated by regions of low density
- **Major features**
  - → Discover clusters of arbitrary shape
  - → Handle noise
  - → One scan
  - → Need density parameters as termination condition
- **Several interesting studies**
  - → DBSCAN: Ester, et al. (KDD'96)
  - → OPTICS: Ankerst, et al (SIGMOD'99).
  - → DENCLUE: Hinneburg & D. Keim (KDD'98)
  - → CLIQUE: Agrawal, et al. (SIGMOD'98) (more grid-based)

Basic Concepts: ε-neighborhood & core objects

- The neighborhood within a radius ε of a given object is called the **ε-neighborhood** of the object



ε = 1 cm

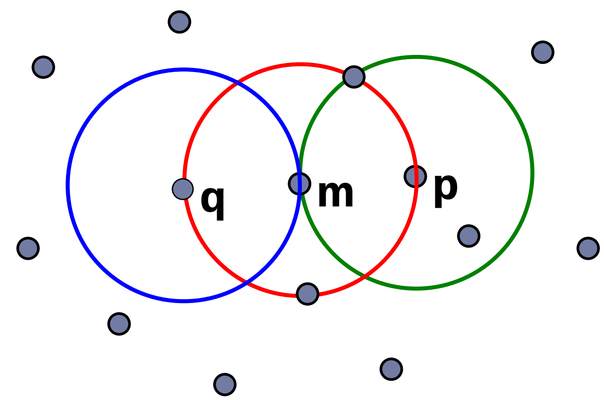
- If the ε-neighborhood of an object contains at least a minimum number, **MinPts**, of objects then the object is called a **core object**

→ **Example:** ε = 1 cm, MinPts=3

m and **p** are core objcets because

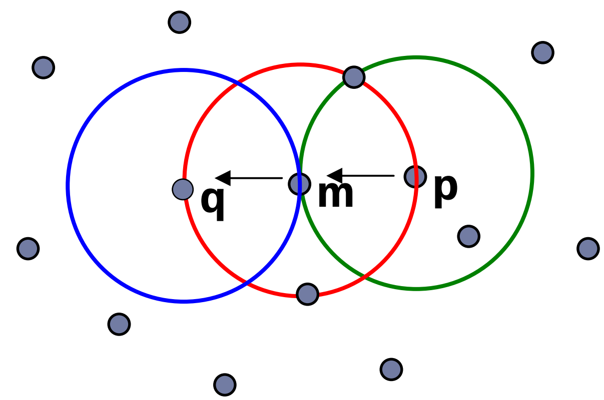
their ε-neighborhoods

contain at least 3 points



Directly density-Reachable Objects

- An object **p** is **directly density-reachable** from object **q** if **p** is within the ε-neighborhood of **q** and ***q*** is a core object

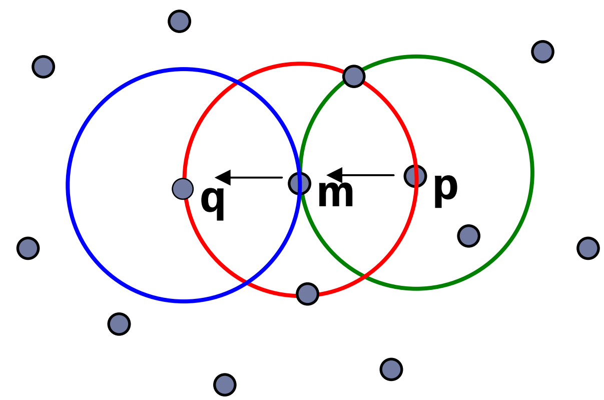


→ **Example:**

q is directly density-reachable from **m**
m is directly density-reachable from **p**
and vice versa

Density-Reachable Objects

- An object **p** is **density-reachable** from object **q** with respect to ε and **MinPts** if there is a chain of objects $\mathbf{p_1}, \ldots \mathbf{p_n}$ where $\mathbf{p_1 = q}$ and $\mathbf{p_n = p}$ such that $\mathbf{p_{i+1}}$ is directly reachable from $\mathbf{p_i}$ with respect to ε and MinPts



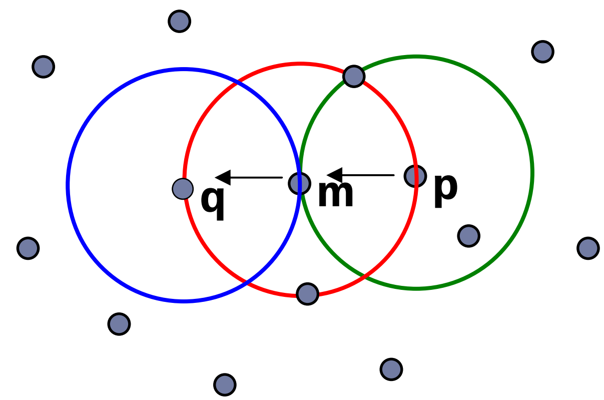
→ **Example:**

q is density-reachable from **p** because **q** is directly density-reachable from **m** and **m** is directly density-reachable from **p**

p is not density-reachable from **q** because **q** is not a core object

Density-Connectivity

- An object **p** is **density-connected** to object **q** with respect to ε and **MinPts** if there is an object **O** such as both **p** and **q** are density reachable from **O** with respect to ε and MinPts



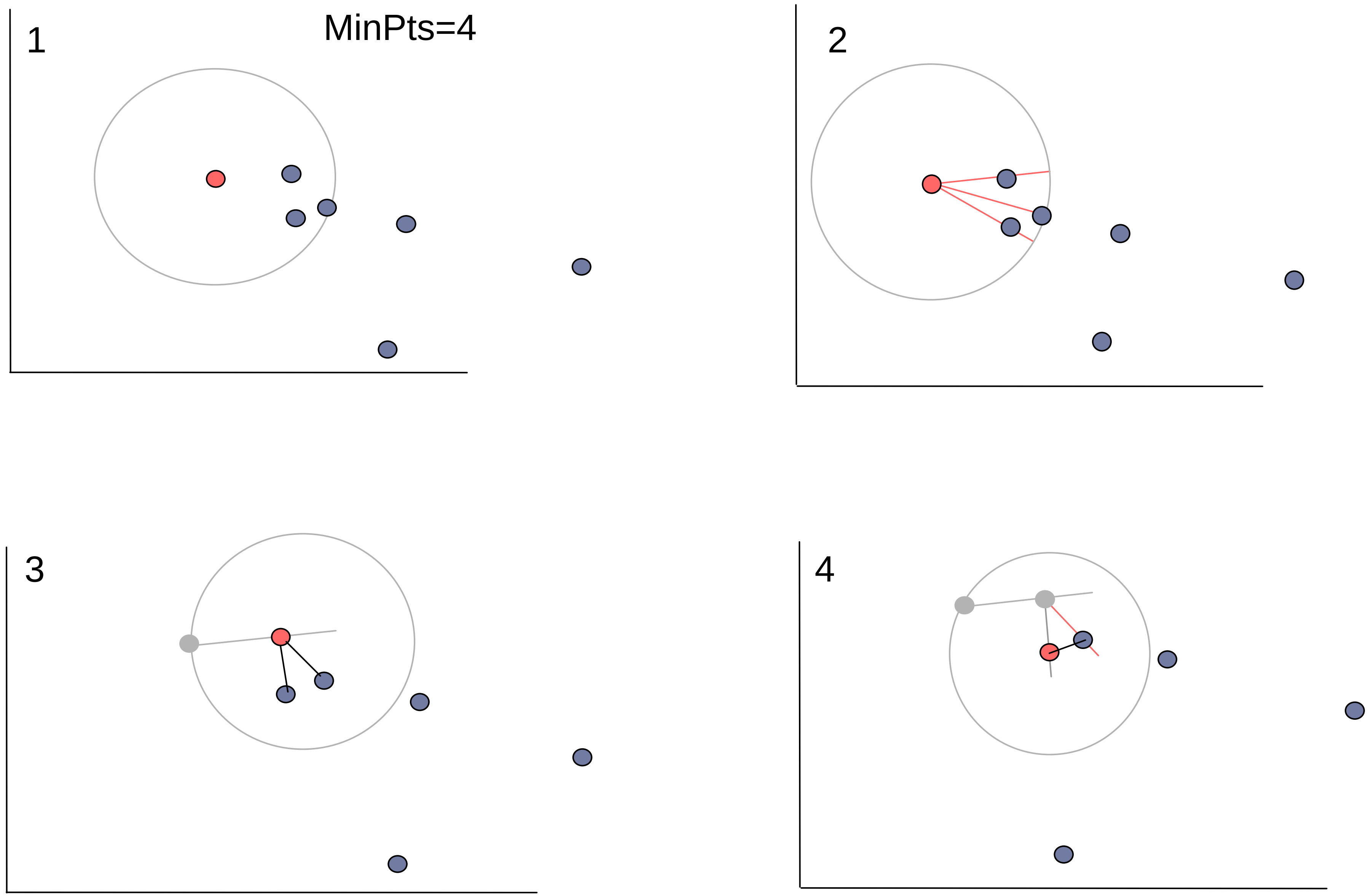
→ **Example:**

p,q and **m** are all density connected

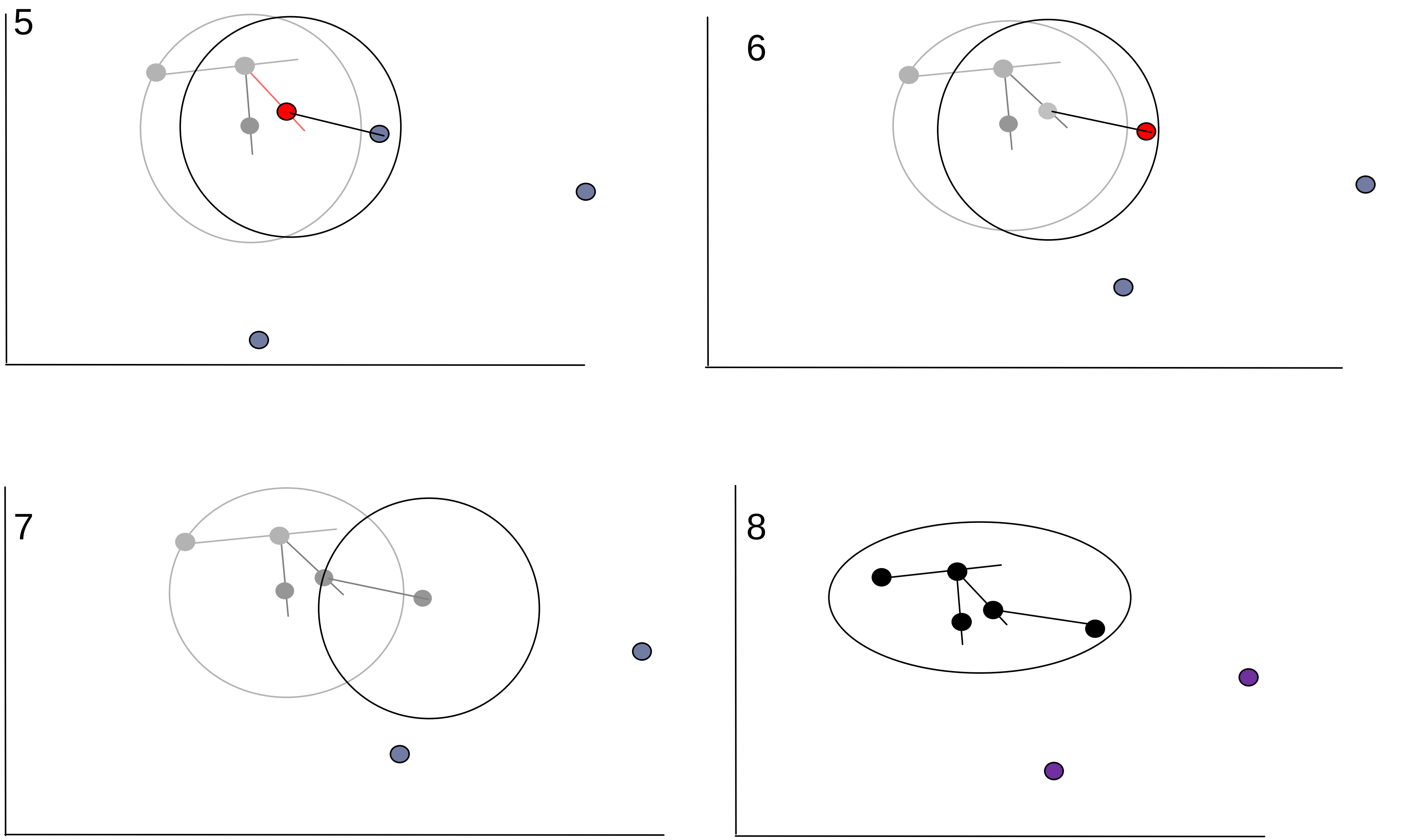
3.4.2 DBSCAN

- Searches for clusters by checking the ε-neighborhood of each point in the database
- If the ε-neighborhood of a point p contains more than MinPts, a new cluster with a core object is created
- DBSCAN iteratively collects directly density reachable objects from these core objects. Which may involve the merge of a few density-reachable clusters
- The process terminates when no new point can be added to any cluster

Density-based Clustering

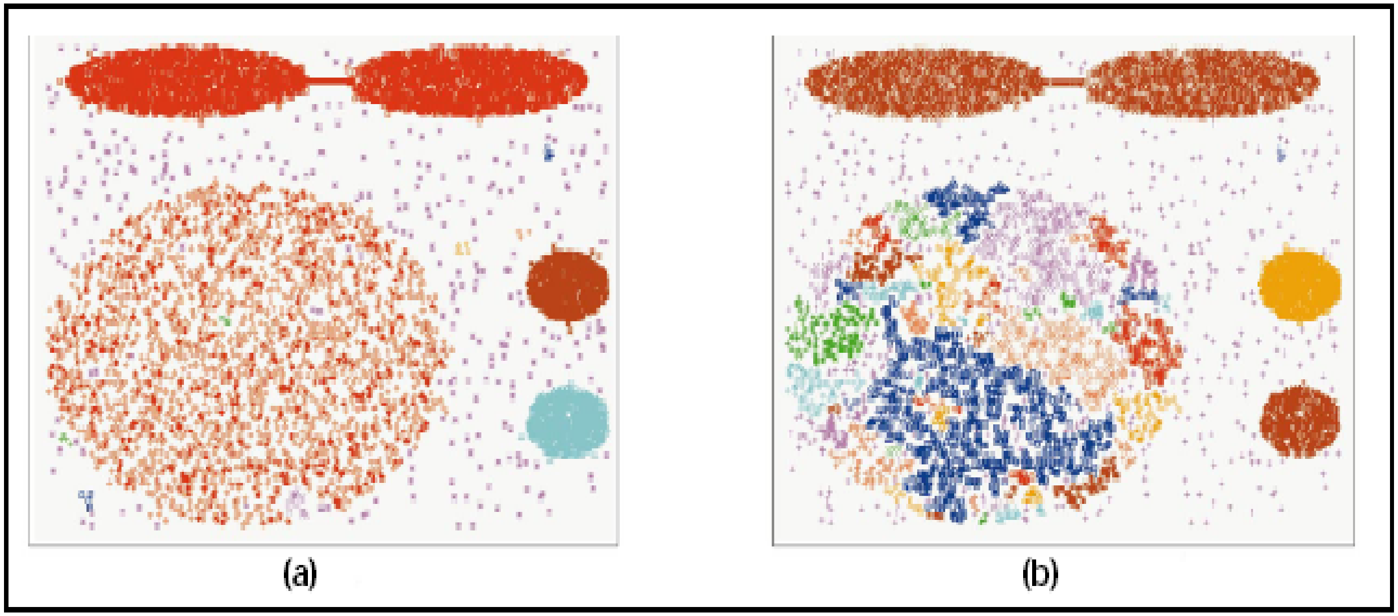


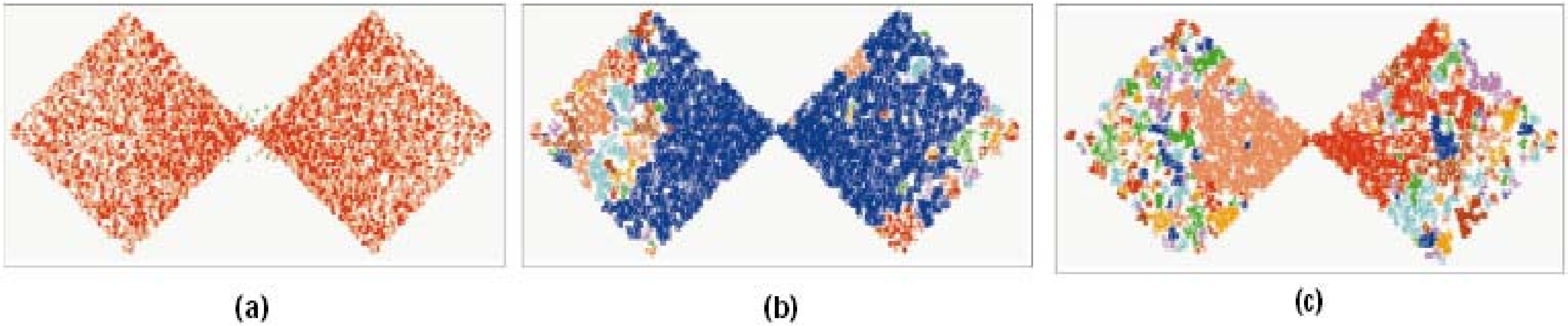
Density-based Clustering



DBSCAN: Sensitive to Parameters

Figure 8. DBScan results for DS1 with MinPts at 4 and Eps at (a) 0.5 and (b) 0.4.

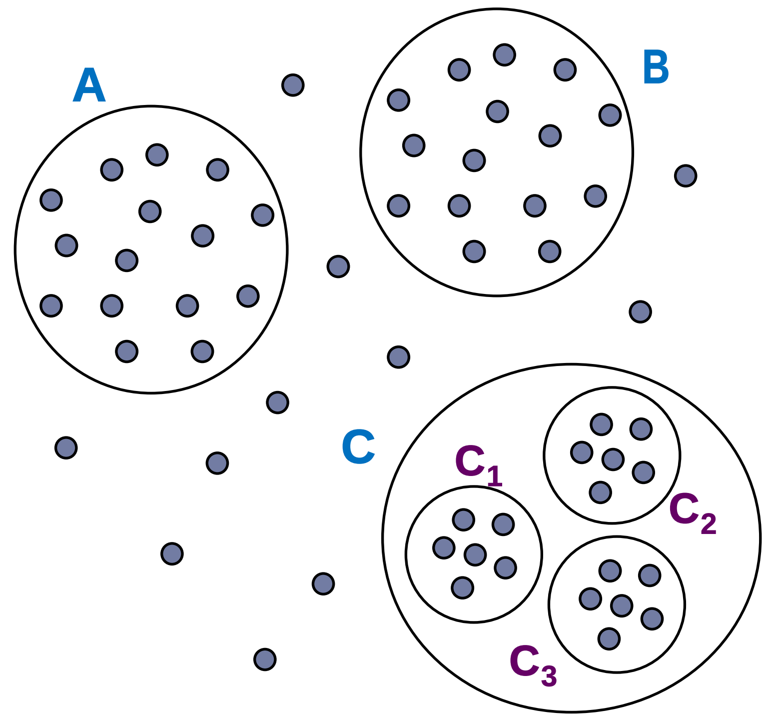




3.4.3 OPTICS

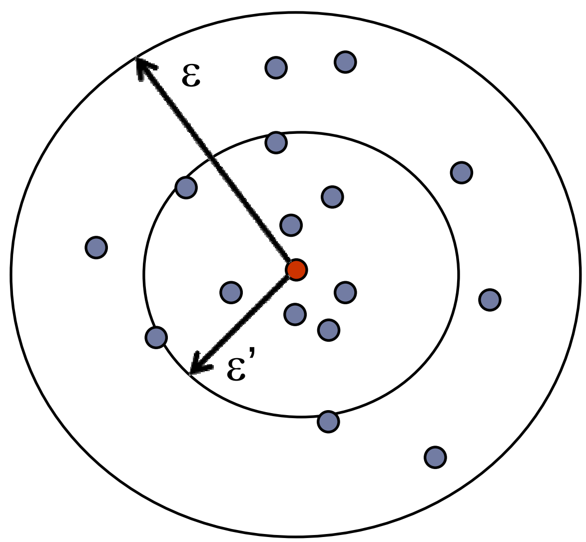
Motivation

- Very different local densities may be needed to reveal clusters in different regions
- Clusters **A**,**B**,$\mathbf{C_1}$,$\mathbf{C_2}$, and $\mathbf{C_3}$ cannot be detected using one global density parameter
- A global density parameter can detect either **A**,**B**,**C** or $\mathbf{C_1}$,$\mathbf{C_2}$,$\mathbf{C_3}$
- **Solutions**
  - → Use OPTICS



OPTICS Principle

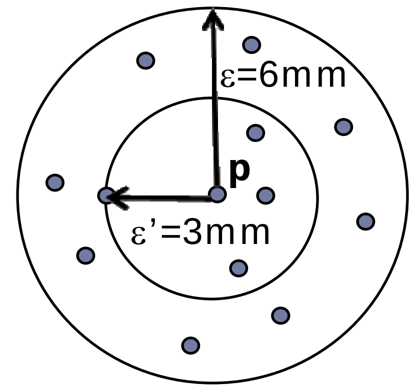
- Produce a special order of the database
  - → with respect to its density-based clustering structure
  - → contain information about every clustering level of the data set (up to a generating distance ε)



→ Which information to use?

Core-distance and Reachability-distance

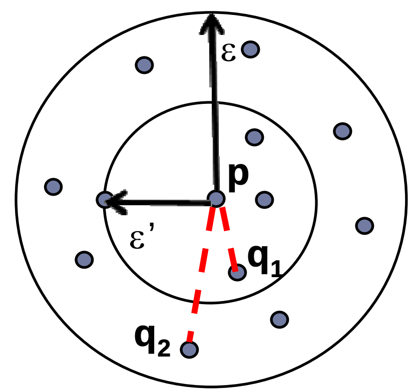
- The **core-distance** of an object is the smallest ε'that makes **{p}** a core object
  - → If **p** is not a core object, the core distance of **p** is **undefined**
  - → Example (ε, MinPts=5)
    - ε' is the core distance of p
    - It is the distance between p and the fourth closest object



- The **reachability-distance** of an object **q** with respect to object to object **p** is:

Max(core-distance(p), Euclidian(p,q))

  - → **Example**
    - Reachability-distance(q_1,p)=core-distance(p)=ε
    - Reachability-distance(q_2,p)=Euclidian(q_2,p)



OPTICS Algorithm

- Creates an ordering of the objects in the database and stores for each object its:
  - Core-distance
  - Distance reachability from the closest core object from which an object have been directly density-reachable
- This information is sufficient for the extraction of all density-based clustering with respect to any distance ε' that is smaller than ε used in generating the order

Illustration of Cluster Ordering