Chapter 7. Cluster Analysis

Dong-Kyu Chae

PI of the Data Intelligence Lab @HYU
Department of Computer Science & Data Science
Hanyang University

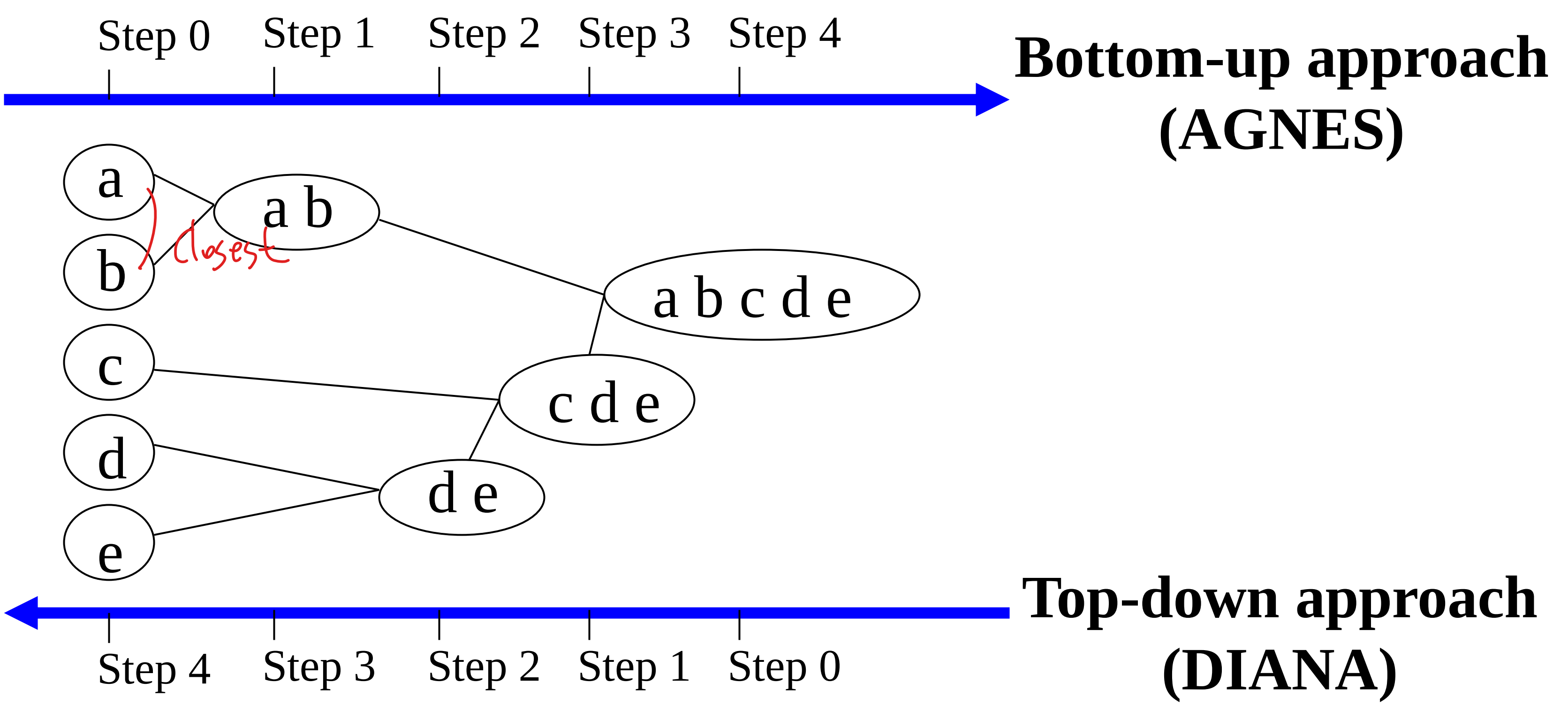
Data
Intelligence
LAB

Contents

Data Intelligence LAB

Hierarchical Clustering

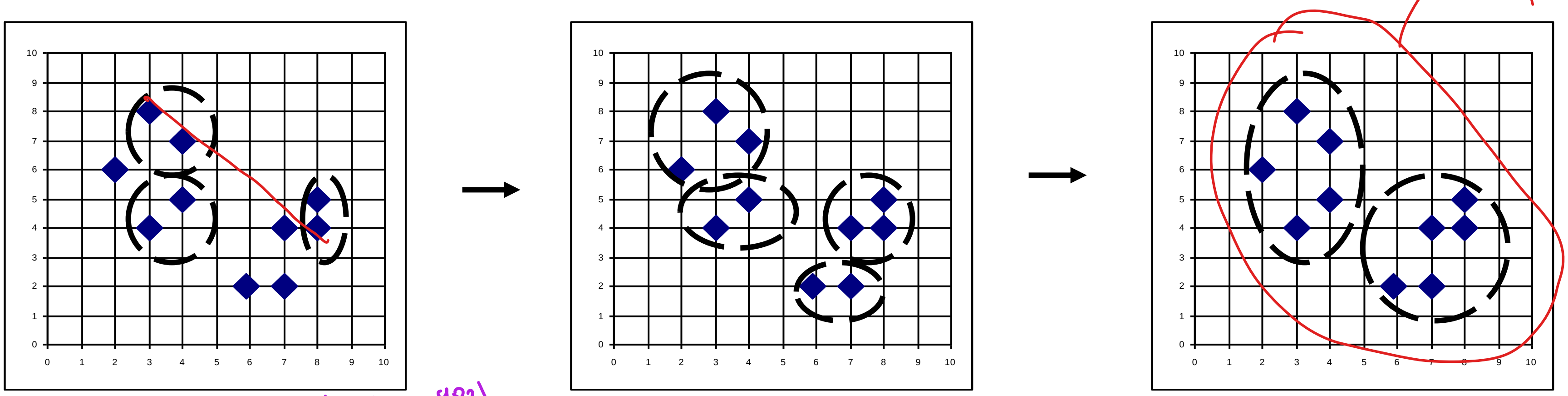
- It also uses distance as clustering criteria
- Does not require the number of clusters *k* as an input, but needs a termination condition



AGNES (AGglomerative NESting)

- AGNES uses the **single-link method** to compute the distance between a cluster and another cluster (or a data point)
- **Merge nodes that have the least dissimilarity**
  - Nodes = a data point or a cluster of data points
- **Eventually all nodes belong to the same cluster**

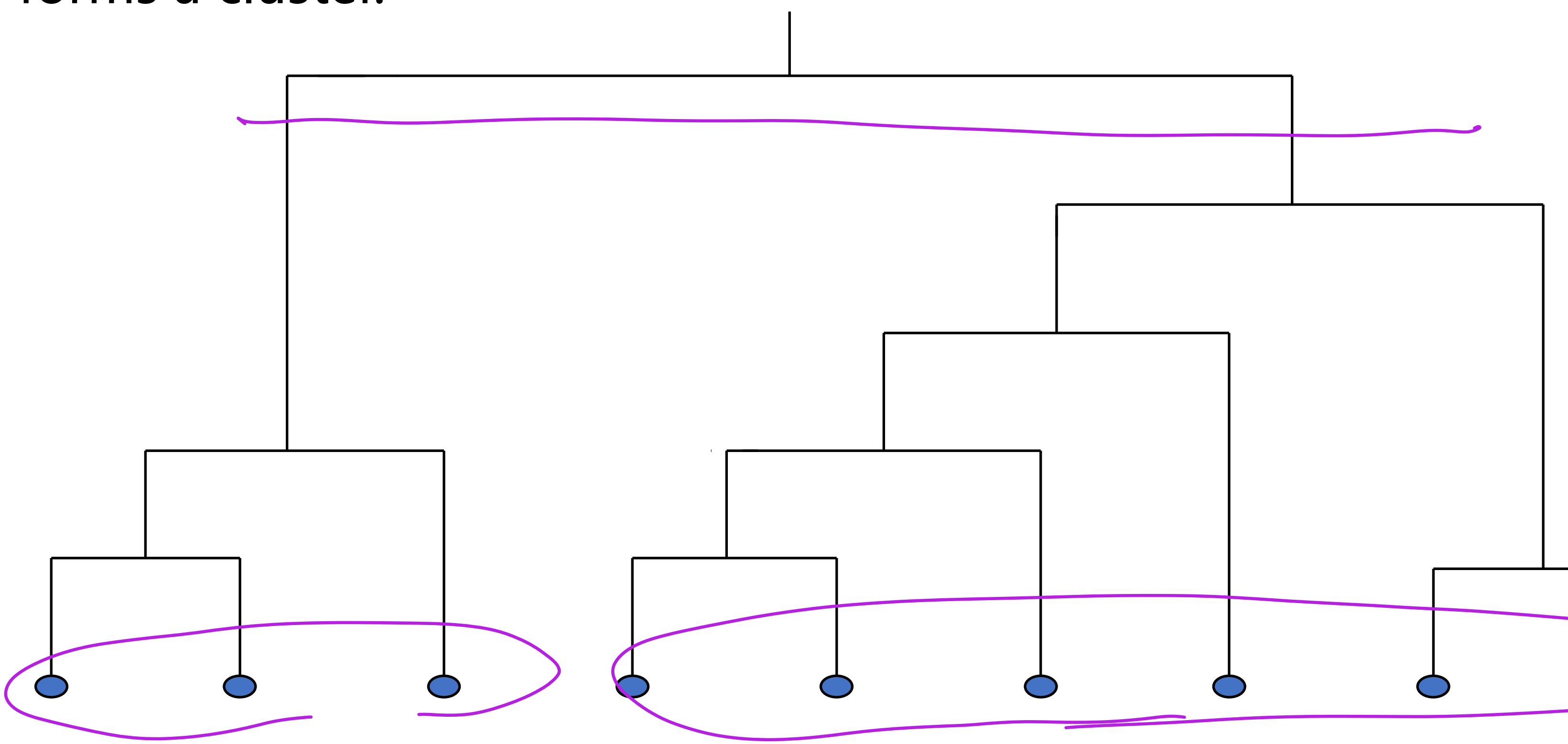
final



example is not very good

Dendrogram: How the Clusters are Merged

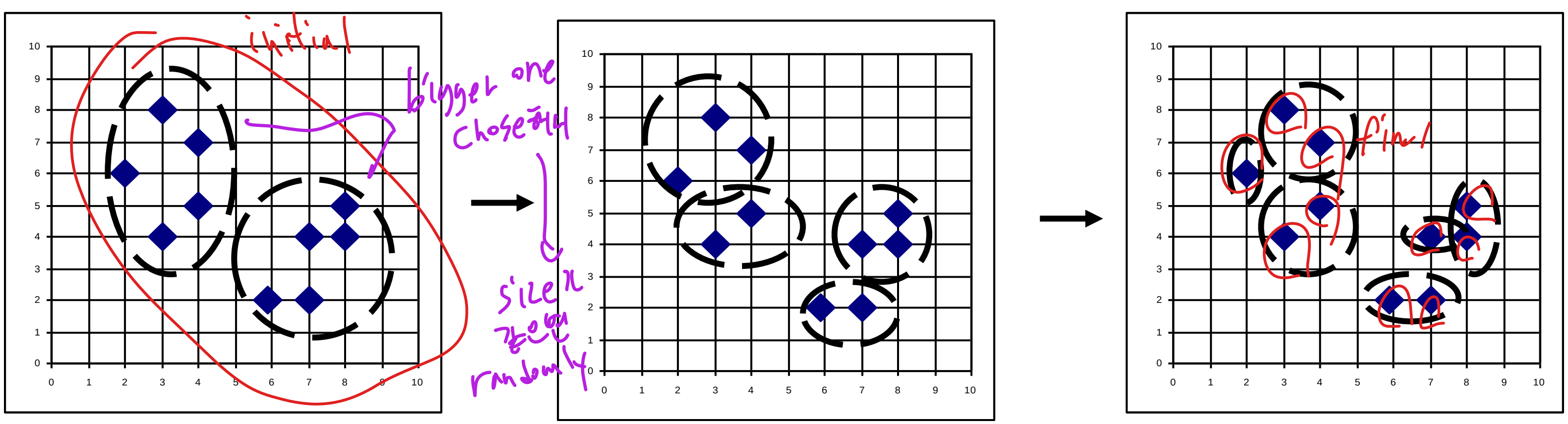
- Decompose data objects into a several levels of nested partitioning (tree of clusters), called a dendrogram.
- A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster.



level 별로
cut
level
choose
해야 함

DIANA (DIvisive ANAlysis)

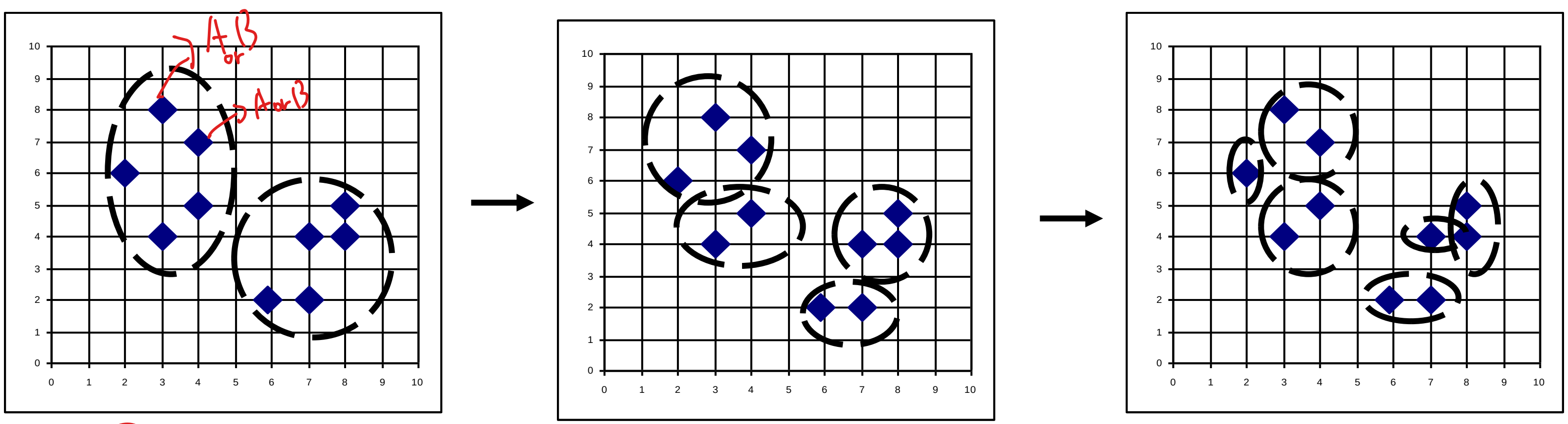
- **DIANA inverses the order of AGNES**
- **Eventually each node forms a cluster on its own**
- **Outline**
  - Initially, there is **one** large cluster consisting of **all** *n* objects
  - At each subsequent step, the largest available cluster is split into two clusters
    - Until finally all clusters comprise of a single object.
    - Thus, the hierarchy is built in *n*-1 steps.



DIANA

❑ Complexity in the first step

- ❑ AGNES: $\frac{n(n-1)}{2}$ possible combinations should be tested nC_2
- ❑ DIANA: $2^{n-1}-1$ possible combinations should be tested 모두 포함된 cluster
  A에 넣어이면 2개씩됨
  - Considerably larger than the bottom-up method



모두 A 모두 B

는 나빠야 함 모두 A인 경우 마지막거면
무조건 B

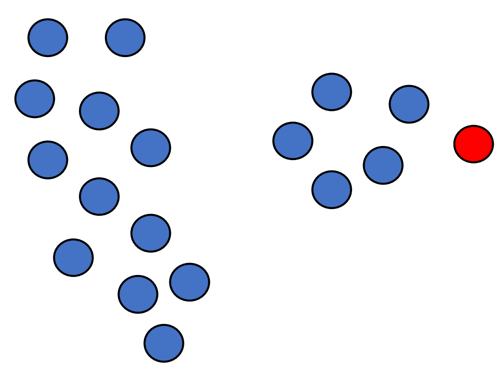
DIANA

- **To avoid considering all possibilities, an approximation algorithm proceeds as follows.**

1. Find the object, which has the highest average dissimilarity to all other objects. This object initiates a new cluster– a sort of a *splinter group*.
2. For each object *i* outside the *splinter group*, compute:

$$D_i = \left[average\ d(i,j)\ j \notin R_{splinter\ group}\right] - \left[average\ d(i,j)\ j \in R_{splinter\ group}\right]$$

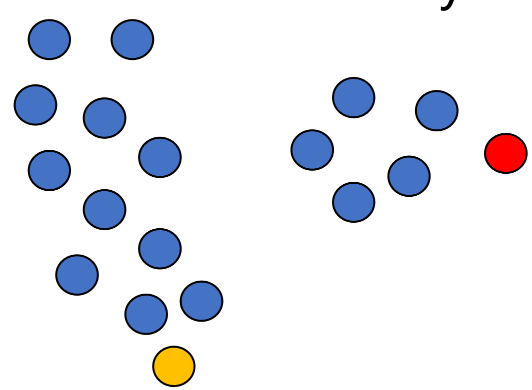
3. Find an object *h* for which the difference D_h is the largest. If D_h is positive, then put *h* into the splinter group.



DIANA

- **To avoid considering all possibilities, an approximation algorithm proceeds as follows.**

4. Repeat *Steps* 2 and 3 until all differences D_h are negative. The data set is then split into two clusters: the **splinter group** and **the rest.**
5. Select the cluster with the largest diameter.
   - The diameter of a cluster is the largest dissimilarity between any two of its objects.
6. Then divide this cluster, following steps 1-4.
7. Repeat *Step* 5 until all clusters contain only a single object.



Contents

Advanced Hierarchical Clustering Methods

- **Major weakness of hierarchical clustering methods**
  - Do not scale well: time complexity of at least $O(n^2)$, where n is the number of data points

- **Integration of hierarchical with distance-based clustering**
  - BIRCH (1996): uses CF-tree and incrementally adjusts the quality of sub-clusters
  - ROCK (1999): clustering categorical data by neighbor and link analysis
  - CHAMELEON (1999): hierarchical clustering using dynamic modeling

BIRCH: Overview

- **Birch:** Balanced Iterative Reducing and Clustering using Hierarchies
- **Incrementally construct a CF (Clustering Feature) tree, a hierarchical data structure for multiphase clustering**
  - **Phase 1**: scans DB to build a CF tree (a multi-level compression of the data that tries to preserve its inherent clustering structure)
  - **Phase 2**: uses an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree, and (optional) refine the clustering results
- **Scales linearly :** finds a good clustering with a single scan and improves the quality with a few additional scans
- **Weakness***:* handles only numeric data, and sensitive to the order of the data records

order에 따라 result가 달라짐

Background: Clustering Feature

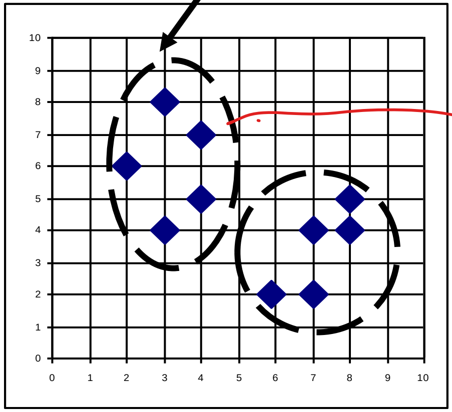
- Clustering feature (CF)
  - **Summary** of the statistics for a given cluster
  - Registers very small but important measurements for computing a cluster and utilizes storage efficiently

- **Clustering Feature:** $CF = (n, \overrightarrow{LS}, \overrightarrow{SS})$
  - n: **Number of data points**
  - LS: $\sum_{i=1}^{n} \overrightarrow{X_i}$
  - SS: $\sum_{i=1}^{n} \overrightarrow{X_i}^2$

linear sum

Square sum

CF = (5, (16,30),(54,190))



(3,4)
(2,6)
(4,5)
(4,7)
(3,8)

=16,30

3^2+2^2+..
...=54

Data
Intelligence
LAB

Usage of CF

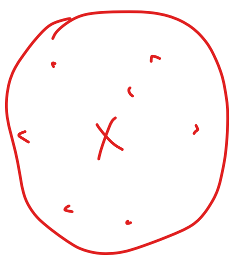
- **Centroid** can be obtained from **n** and **LS** $\frac{LS}{n}$ $\overrightarrow{x_0} = \frac{\sum_{i=1}^{n} \overrightarrow{x_i}}{n}$

- **Radius** and **Diameter** can be obtained from **n, LS** and **SS**

↓수정됨

$SS = (a, b)$
$= a+b$

$LS = (a, b)$
$LS^2 = a^2+b^2$



$$R = \sqrt{\frac{\sum_{i=1}^{n} (\vec{x}_i - \vec{x}_0)^2}{n}} = \sqrt{\frac{n\underbrace{SS}_{a+b} - 2\underbrace{LS^2}_{a^2+b^2} + n\underbrace{LS}_{a+b}}{n^2}},$$

vector sign이 없으면

$$D = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} (\vec{x}_i - \vec{x}_j)^2}{n(n-1)}} = \sqrt{\frac{2nSS - 2LS^2}{n(n-1)}}.$$

CF-Tree in BIRCH

each
root에서 leaf node까지 같은 length

h

$CF_0 = CF_1 + CF_2 + CF_3$

- **A CF tree is a height-balanced tree that stores the CFs for a hierarchical clustering**
  - **A non-leaf node (including root)**: has descendants or "children" and stores the sum of the CFs of its children
    - A CF of a **combined cluster** can be easily computed by the sum of CFs of its children
  - **Leaf node**: includes several sub-clusters and their CFs

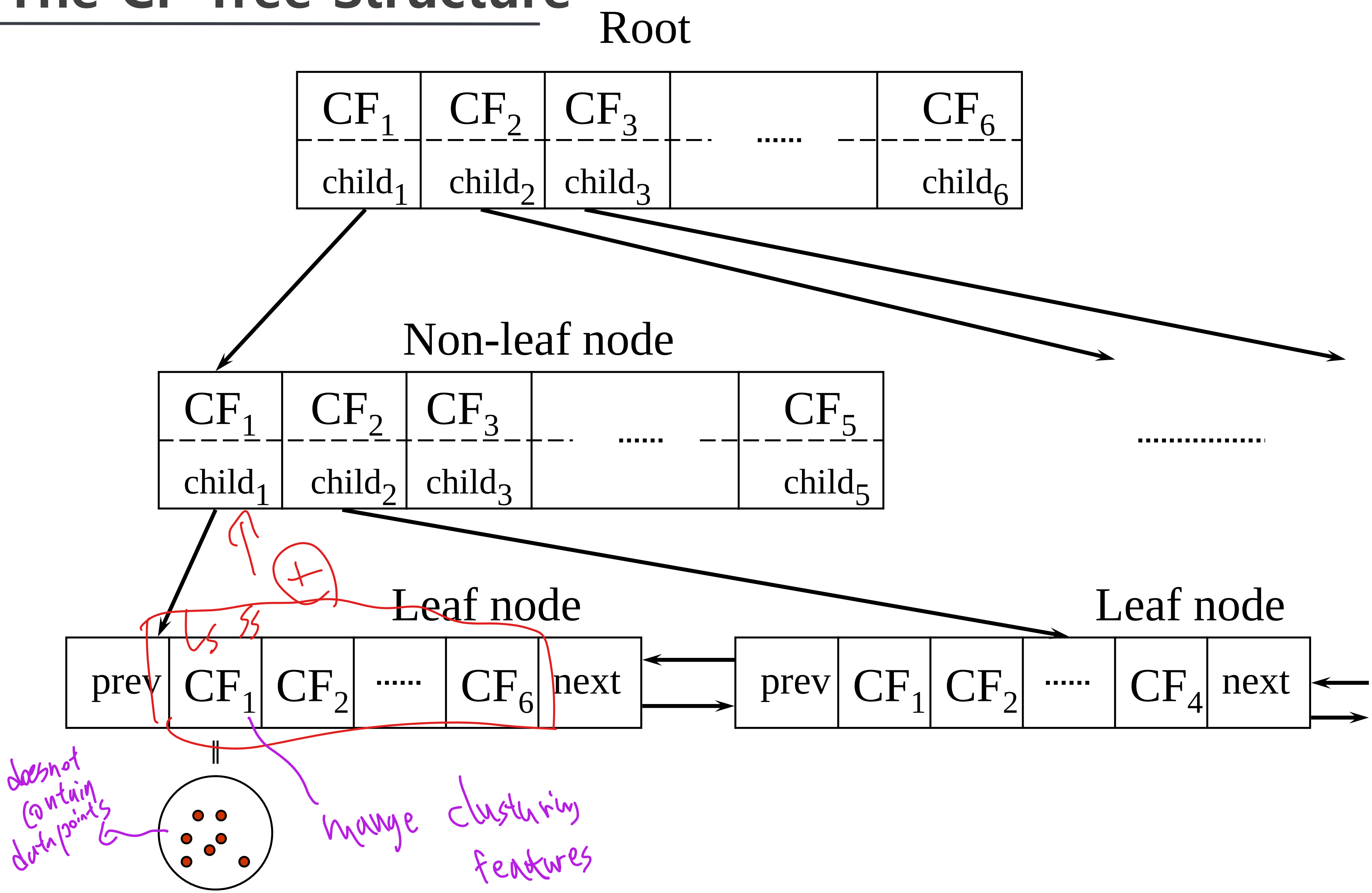
B=3 L=4

exceed

- **Hyper-parameters of the CF-tree**
  - **Branching factor (B, L)** : specify the maximum number of children
    leaf node
    non leaf node 초과되면 split해야함
  - **Threshold**: max radius (or, diameter) of a cluster stored at the leaf node

Threshold 초과하면 split

The CF Tree Structure



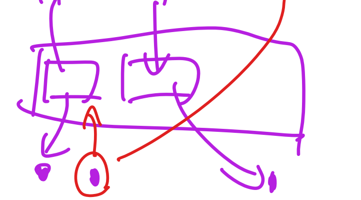
BIRCH Phase 1

- **Initially** the tree has only an empty root.
- **For each** data point *d*, do:
  1. Start from root, traverse down tree to choose the **closest leaf** node for *d*
  2. Search for the closest entry L_i in the chosen leaf node
  3. If: *d* can be inserted in L_i, then **update** CF vector of L_i
     - It is judged based on the **threshold** value
  4. Else if: node has space to insert new entry, insert
     - It is judged based on the **branching factor**
  5. Else: split node
     - It may result in further splitting in its parent nodes
  6. Update CFs of nodes accordingly along path to the root

$CF_1=(1,45,55)$

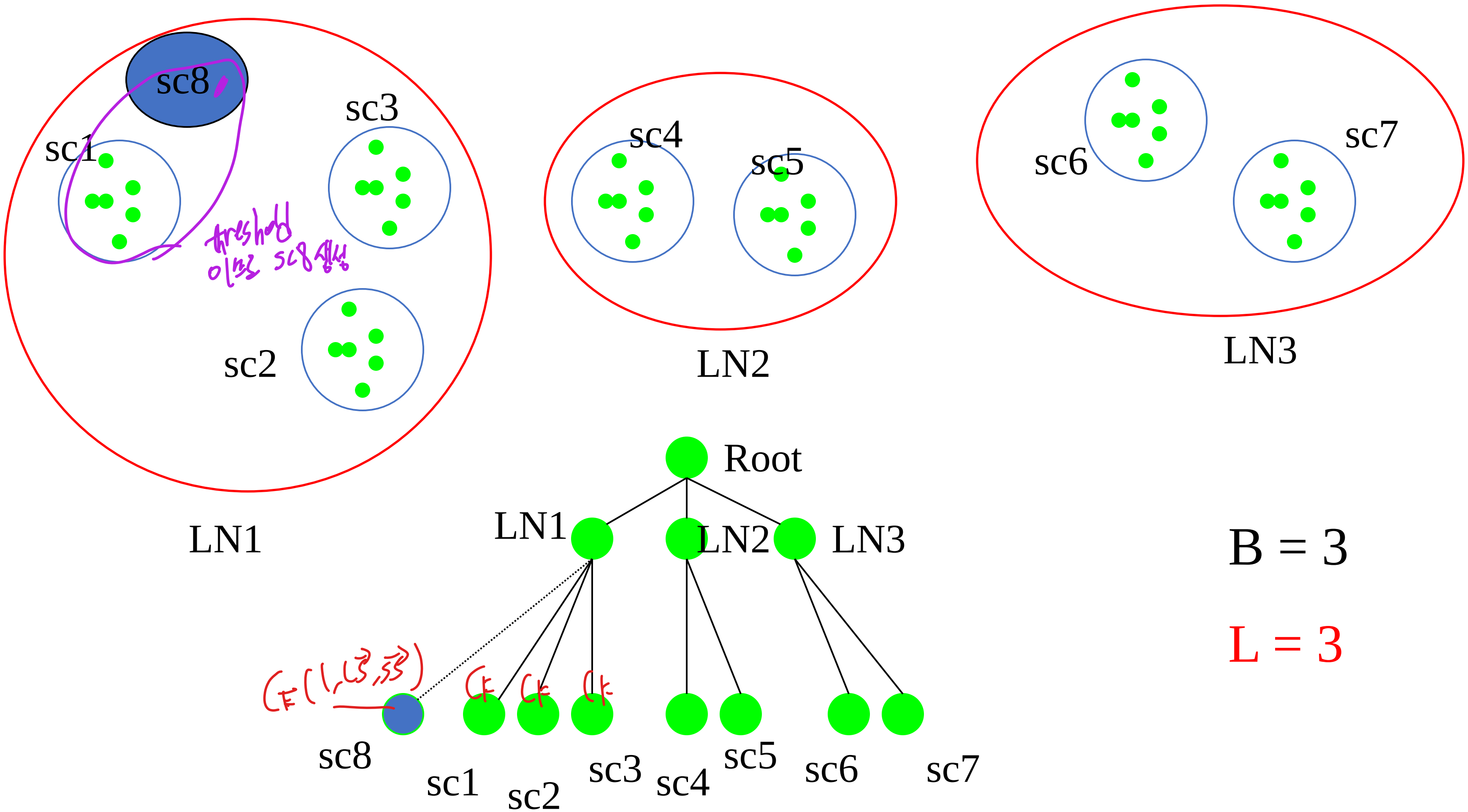
단지 이걸 업데이트 → (2,45,55)

추가



Example of the BIRCH Algorithm

- When a new data point comes and creates a new sub-cluster

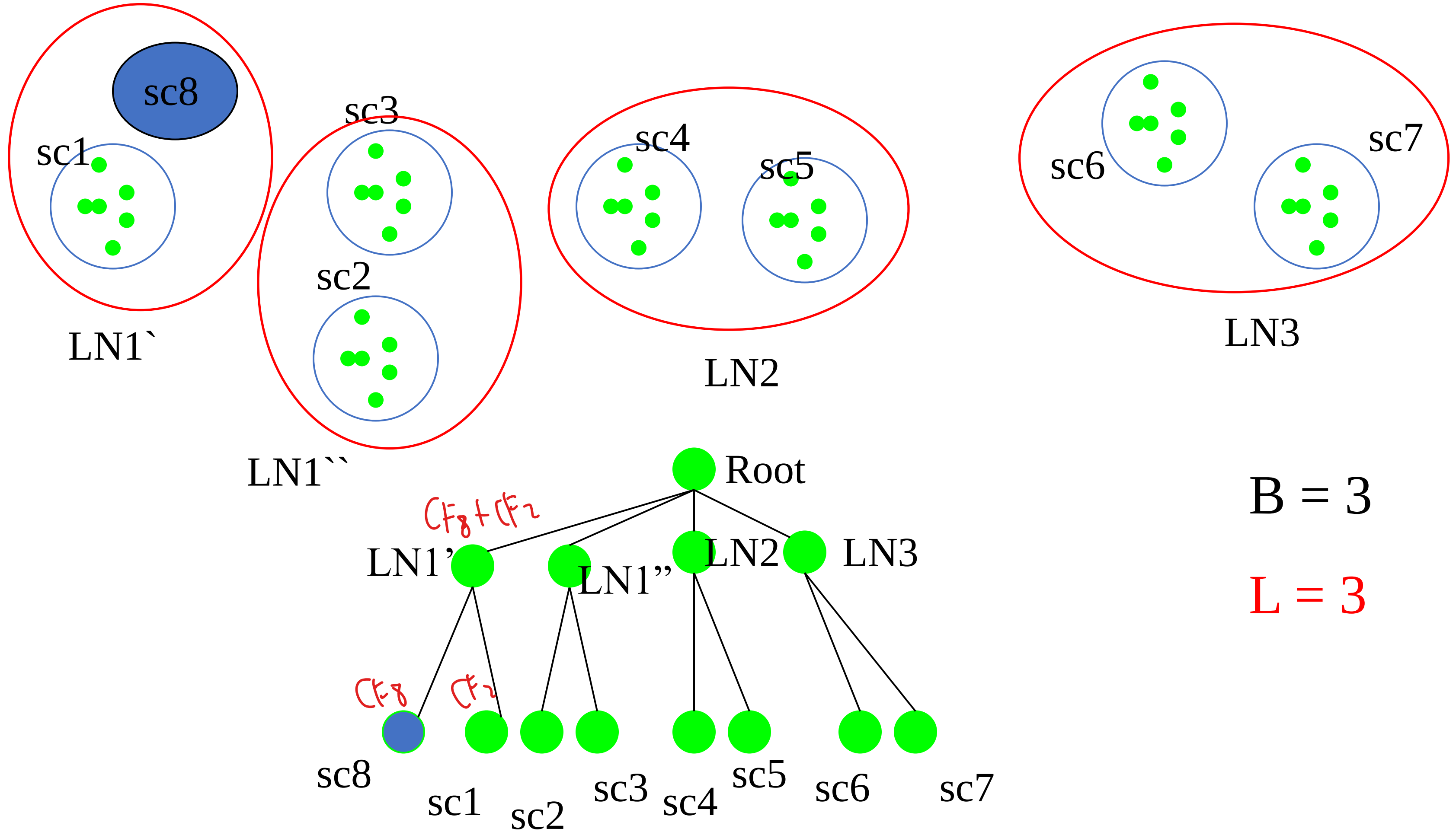


B = 3

L = 3

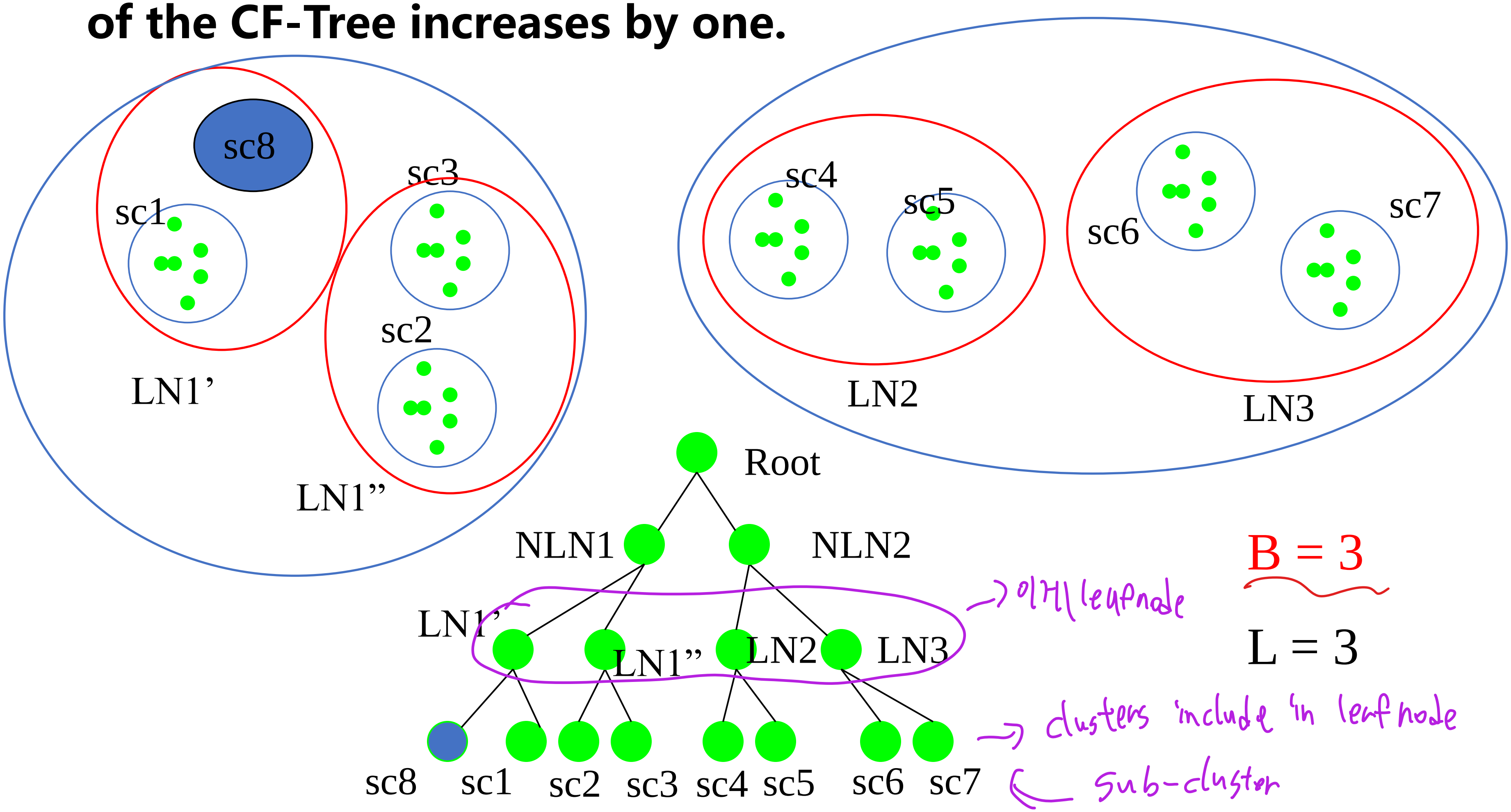
Example of the BIRCH Algorithm

- Because of the branching factor L=3, a leaf node cannot include more than 3 members, so LN1 is split.



Example of the BIRCH Algorithm

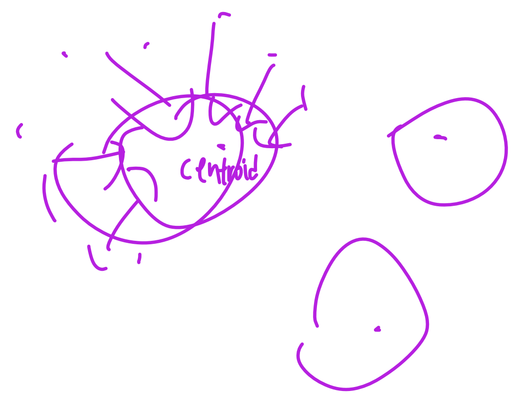
- Because of the branching factor B=3, a non-leaf node also cannot include more than 3 members, so the root is split and the height of the CF-Tree increases by one.



BIRCH Phases 2

Global clustering

- **Treat each sub-cluster as a point (its centroid) and perform clustering on these points**
  - We have reasonable performance, since we have much smaller number of objects to be handled
- Use existing clustering algorithm on sub-clusters at leaf nodes
- Finally, scan DB again to assign all the data points to the identified clusters



Thank You

Data
Intelligence
Lab