CLUSTERING

BIRCH Algorithm

Clustering Large Datasets

- Clustering algorithms we've seen are not good for large datasets because they
  - Assume dataset and data structures used fit in main memory
  - Scan the dataset many times
  - Assume the whole dataset is present at once
    - An incremental algorithm can handle dynamic and incoming data
    - An online algorithm is capable of providing the best answer so far

BIRCH

- BIRCH stands for Balanced Iterative Reducing and Clustering using Hierarchies
- It can handle large datasets by using data structures for summarizing information about the clusters
- Scalable and incremental

Clustering Feature (CF)

- Definition: A clustering feature (CF) is a triple (N, **LS**, **SS**) summarizing information about clusters or sub-clusters of objects where:
  - N is the number of objects
  - **LS** is the linear sum ($\sum_{i=1}^{N} \boldsymbol{t}_{\mathrm{i}}$) sum the individual coordinates
  - **SS** is square sum ($\sum_{i=1}^{N} \boldsymbol{t}_{\mathrm{i}}^{2}$) sum the squared coordinates

Clustering Feature Example

- Find the CF for a cluster that contains the following data points:

$$t1 = (3, 4), t2 = (2, 6), t3 = (4, 5), t4 = (4, 7), and\ t5 = (3, 8)$$

- CF = (N, **LS**, **SS**)
- N = 5
- LS = $\sum_{i=1}^{N} \boldsymbol{t}_{\mathrm{i}} = t1 + t2 + t3 + t4 + t5$
- LS = $(3, 4) + (2, 6) + (4, 5) + (4, 7) + (3, 8)$

$$= (3 + 2 + 4 + 4 + 3, 4 + 6 + 5 + 7 + 8) = (16, 30)$$

- SS = $\sum_{i=1}^{N} \boldsymbol{t}_{\mathrm{i}}{}^{2} = (3^2 + 2^2 + 4^2 + 4^2 + 3^2, 4^2 + 6^2 + 5^2 + 7^2 + 8^2)$
- SS $= (9 + 4 + 16 + 16 + 9, 16 + 36 + 25 + 49 + 64) = (54, 190)$
- CF = (N, **LS**, **SS**) = $(5, (16, 30), (54, 190))$

CF Additivity Property

- Let $CF_1 = (N_1, \mathbf{LS_1}, \mathbf{SS_1})$ and $CF_2 = (N_2, \mathbf{LS_2}, \mathbf{SS_2})$ be the CF vectors of two **disjoint** clusters K_1 and K_2.

  The CF vector of the cluster that is formed by merging K_1 and K_2 is:

  $CF_1 + CF_2 = (N_1 + N_2, \mathbf{LS_1} + \mathbf{LS_2}, \mathbf{SS_1} + \mathbf{SS_2})$

- Proof: straight forward algebra

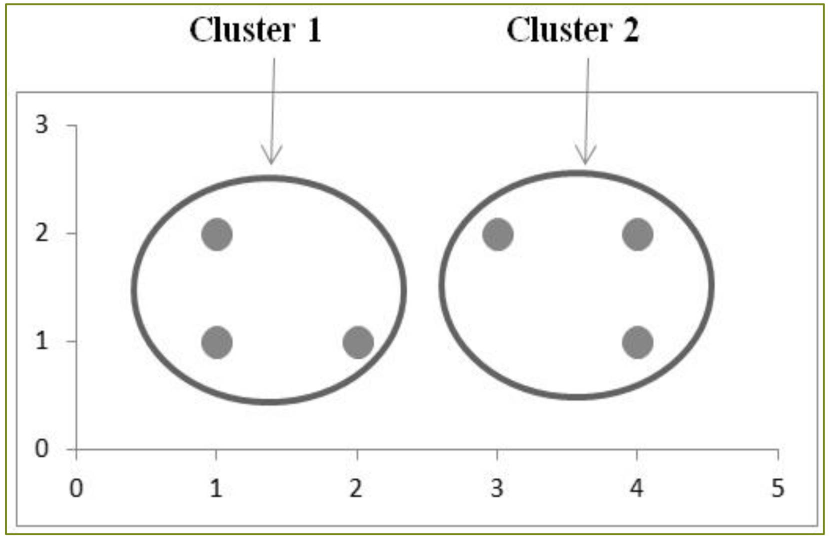
CF Additivity Property - Example

- Let CF1 = $(3, (4, 4), (6,6))$ and CF2 = $(3, (11, 5), (41, 9))$ be the CF vectors of two disjoint clusters K1 and K2. Find the CF vector for the cluster that results from merging the two clusters.

CF = CF1 + CF2 = (N1 + N2, **LS1** + **LS2**, **SS1** + **SS2**)

= $(3 + 3, (4 + 11, 4 + 5), (6 + 41, 6 + 9)$

= $(6, (15, 9), (47, 15))$



Cluster 1 = $\{(1, 1), (2, 1), (1, 2)\}$
Cluster 2 = $\{ (3, 2), (4, 1), (4, 2)\}$

Using the CF to Compute Cluster Parameters

- Knowing the CF vector of a cluster, it is easy to compute the cluster’s centroid, radius, and diameter
- Remember, for a cluster $K_m = \{ t_{m1}, t_{m2}, \ldots, t_{mN}\}$ of N points:

$$centroid = C_m = \frac{\sum_{i=1}^{N}(t_{mi})}{N}$$

$$radius = R_m = \sqrt{\frac{\sum_{i=1}^{N}(t_{mi} - C_m)^2}{N}}$$

$$diameter = D_m = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}(t_{mi} - t_{mj})^2}{(N)(N-1)}}$$

Using the CF to Compute Cluster Parameters – Centroid

- Knowing the CF vector of a cluster, it is easy to compute the cluster's centroid, radius, and diameter
- For a cluster $K_m = \{ t_{m1}, t_{m2}, \ldots, t_{mN}\}$ of N points:

$$centroid = C_m = \frac{\sum_{i=1}^{N}(t_{mi})}{N} = \frac{LS}{N}$$

Using the CF to Compute Cluster Parameters – Radius

- Knowing the CF vector of a cluster, it is easy to compute the cluster's centroid, radius, and diameter
- For a cluster $K_m = \{ t_{m1}, t_{m2}, \ldots, t_{mN}\}$ of N points:

$$radius = R_m = \sqrt{\frac{\sum_{i=1}^{N}(t_{mi} - C_m)^2}{N}}$$

$$Rm = \sqrt{\frac{\sum_{i=1}^{N}(t_{mi}{}^2 - 2t_{mi}C_m + C_m{}^2)}{N}} = \sqrt{\frac{\sum_{i=1}^{N} t_{mi}{}^2 - 2C_m \sum_{i=1}^{N} t_{mi} + \sum_{i=1}^{N} C_m{}^2}{N}}$$

$$Rm = \sqrt{\frac{\boldsymbol{SS} - 2C_m\boldsymbol{LS} + NC_m{}^2}{N}}$$

Using the CF to Compute Cluster Parameters – Diameter

- Knowing the CF vector of a cluster, it is easy to compute the cluster's centroid, radius, and diameter
- For a cluster $K_m = \{ t_{m1}, t_{m2}, \ldots, t_{mN}\}$ of N points:

$$diameter = D_m = \sqrt{\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (t_{mi} - t_{mj})^2}{(N)(N-1)}}$$

We can show, $D_m = \sqrt{\frac{2N\boldsymbol{SS} - 2\boldsymbol{LS}^2}{N(N-1)}}$

Using the CF to Compute Distance Between Two Cluster

- The centroid Euclidean distance D2 between two clusters K1 and K2 is

$$D2 = \sqrt{(C_1 - C_2)^2}$$

$$= \sqrt{(\frac{\boldsymbol{LS1}}{N1} - \frac{\boldsymbol{LS2}}{N2})^2}$$

- The centroid Manhattan distance D1 between two clusters K1 and K2 is

$D1 = |C_1 - C_2| = \sum_{i=1}^{d} |C_{1i} - C_{2i}|$, where d is number of dimensions

$$= |\frac{\boldsymbol{LS1}}{N1} - \frac{\boldsymbol{LS2}}{N2}|$$

Example

- Let CF1 = $(3, (4, 4), (6,6))$ and CF2 = $(3, (11, 5), (41, 9))$ be the CF vectors of two clusters K1 and K2.

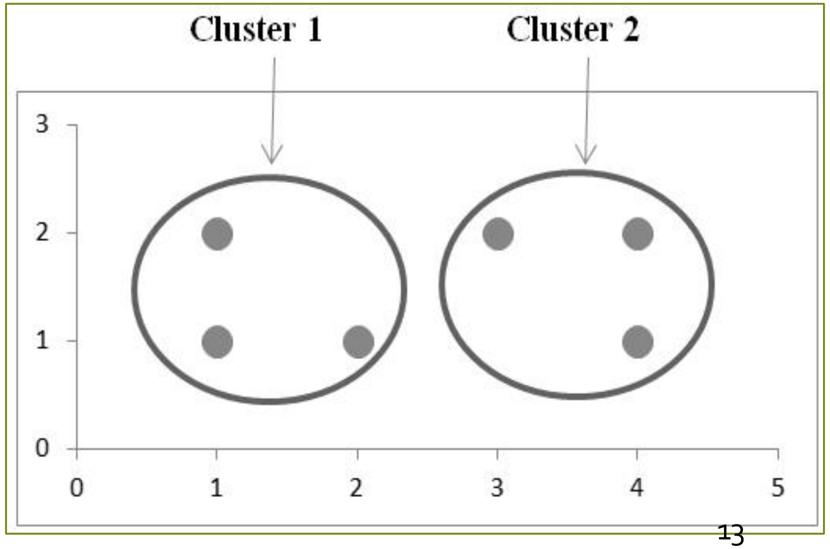
- D2= $\sqrt{(C_1 - C_2)^2} = \sqrt{(\frac{\boldsymbol{LS1}}{N1} - \frac{\boldsymbol{LS2}}{N2})^2}$

$$= \sqrt{(\frac{(4,4)}{3} - \frac{(11,5)}{3})^2} = \sqrt{\big((1.33, 1.33) - (3.67, 1.67)\big)^2}$$

$$= \sqrt{(1.33 - 3.67)^2 + (1.33 - 1.67)^2} = 5.48$$

- D1 = $|C_1 - C_2| = \sum_{i=1}^{d} |C_{1i} - C_{2i}|$

$$= |1.33 - 3.67| + |1.33 - 1.67| = 2.68$$

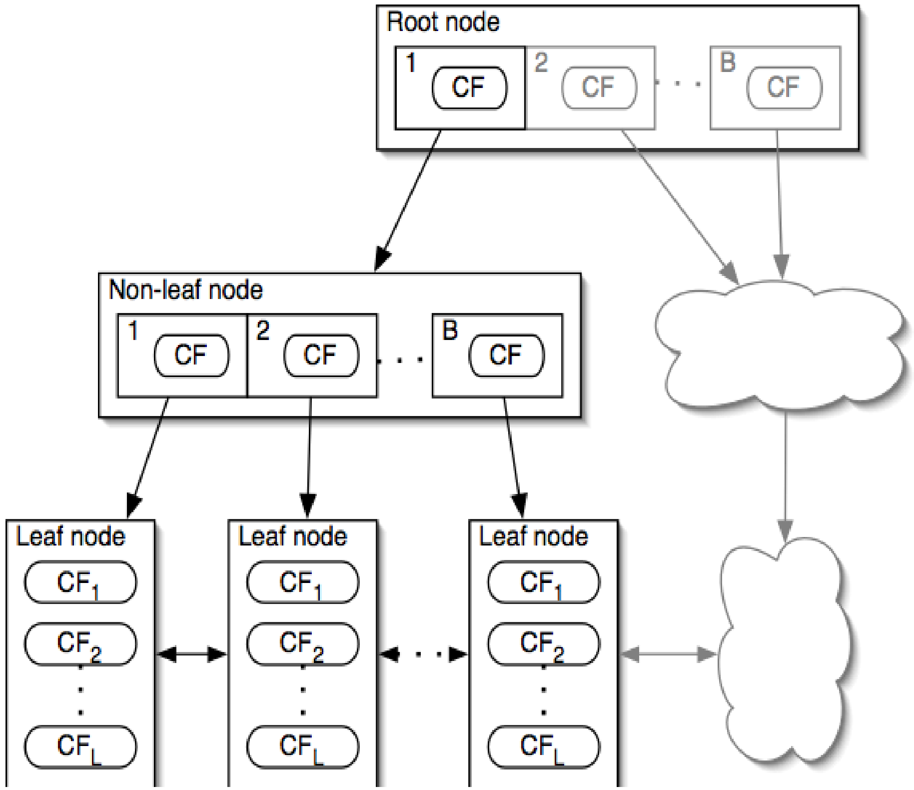


Advantage of CF

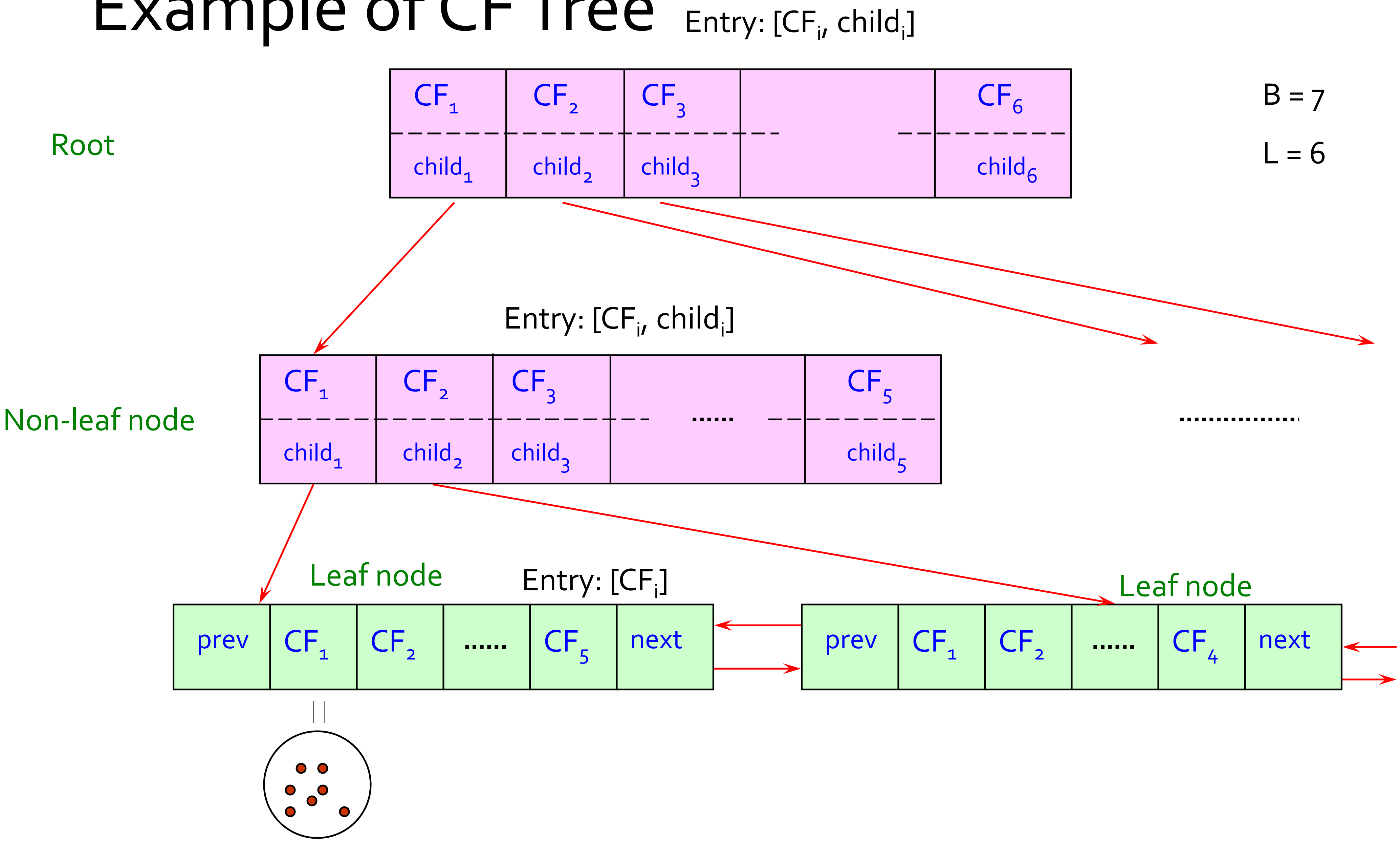
- A CF vector provides summary information about the objects a in cluster
- This summary information is sufficient for calculating all the measurements that are needed in BIRCH
- This makes the algorithm efficient because it stores much less data

Clustering Feature Tree (CF-Tree)

- Height balanced with parameters B (branching factor), L (leaf entries), T (threshold)
- Each **non-leaf node** has at most B entries
  - each entry is a CF tuple and a child node link
  - a non-leaf node represents a cluster made of the clusters represented by its entries
- Each **leaf node** has at most L CF entries
  - each leaf entry satisfies threshold T
  - T is maximum diameter (or radius) of any CF in a leaf node
  - CF represents a subcluster
  - each leaf node has links to the next and previous leaf nodes
- Each node must fit a memory page



Example of CF Tree

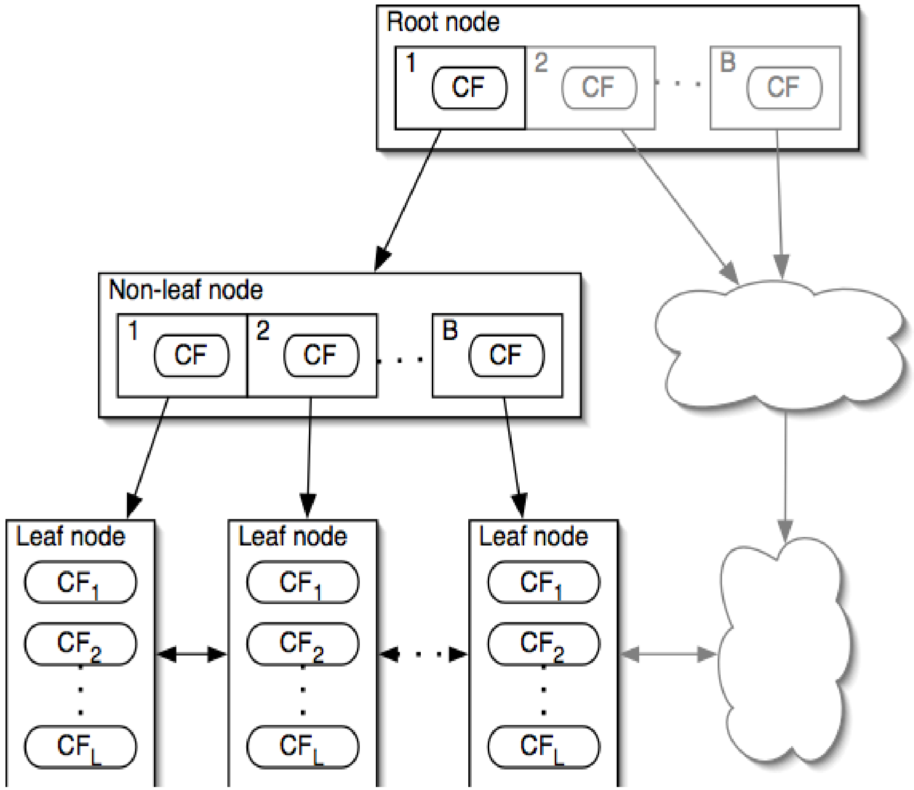


CF Tree

- The CF is a very compact representation of the dataset as each entry in a leaf node is not a single data point but a subcluster/cluster
- Tree size is a function of T
  - larger T, more points in each cluster, smaller tree
  - If not enough memory for a given tree, adjust T
- CF tree built dynamically as data is scanned from the dataset and inserted

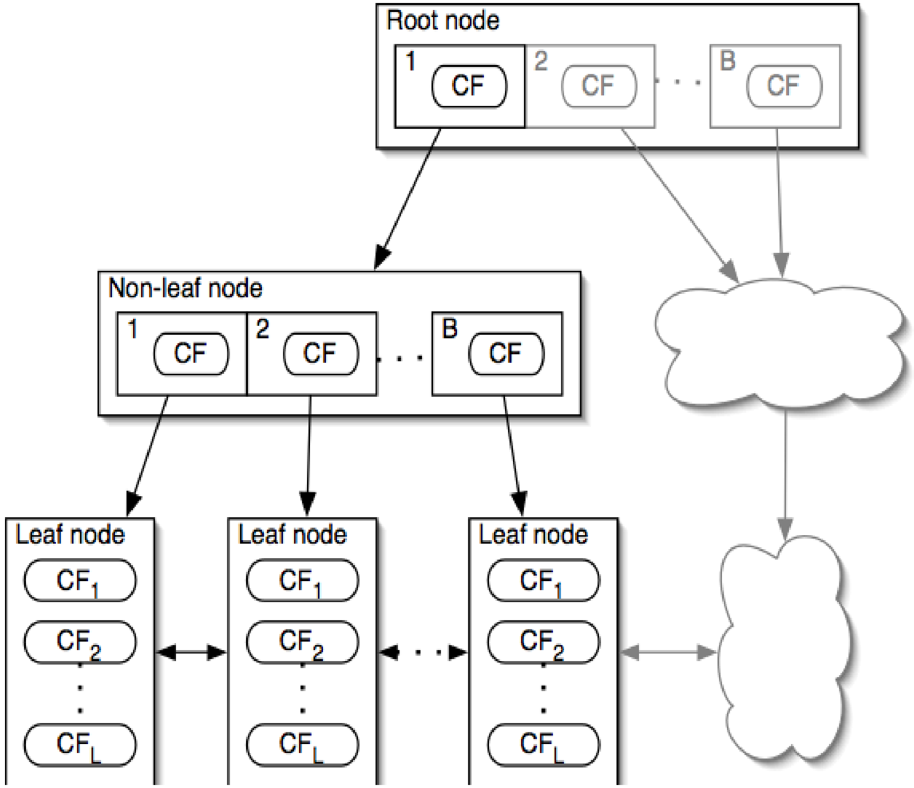
Insertion into a CF Tree

- Identify the appropriate leaf
  - start at root, use appropriate distance measure to compare with CF features
  - Recursively follow closest child until you reach a leaf node
- Modify the leaf
  - find closest leaf entry, L_i, and test if it can absorb new object without violating threshold condition T
  - if yes, update CF_i for L_i
  - if no and there is space on this leaf for a new entry to hold new object, do so
  - leaves have a max size; may need to be split

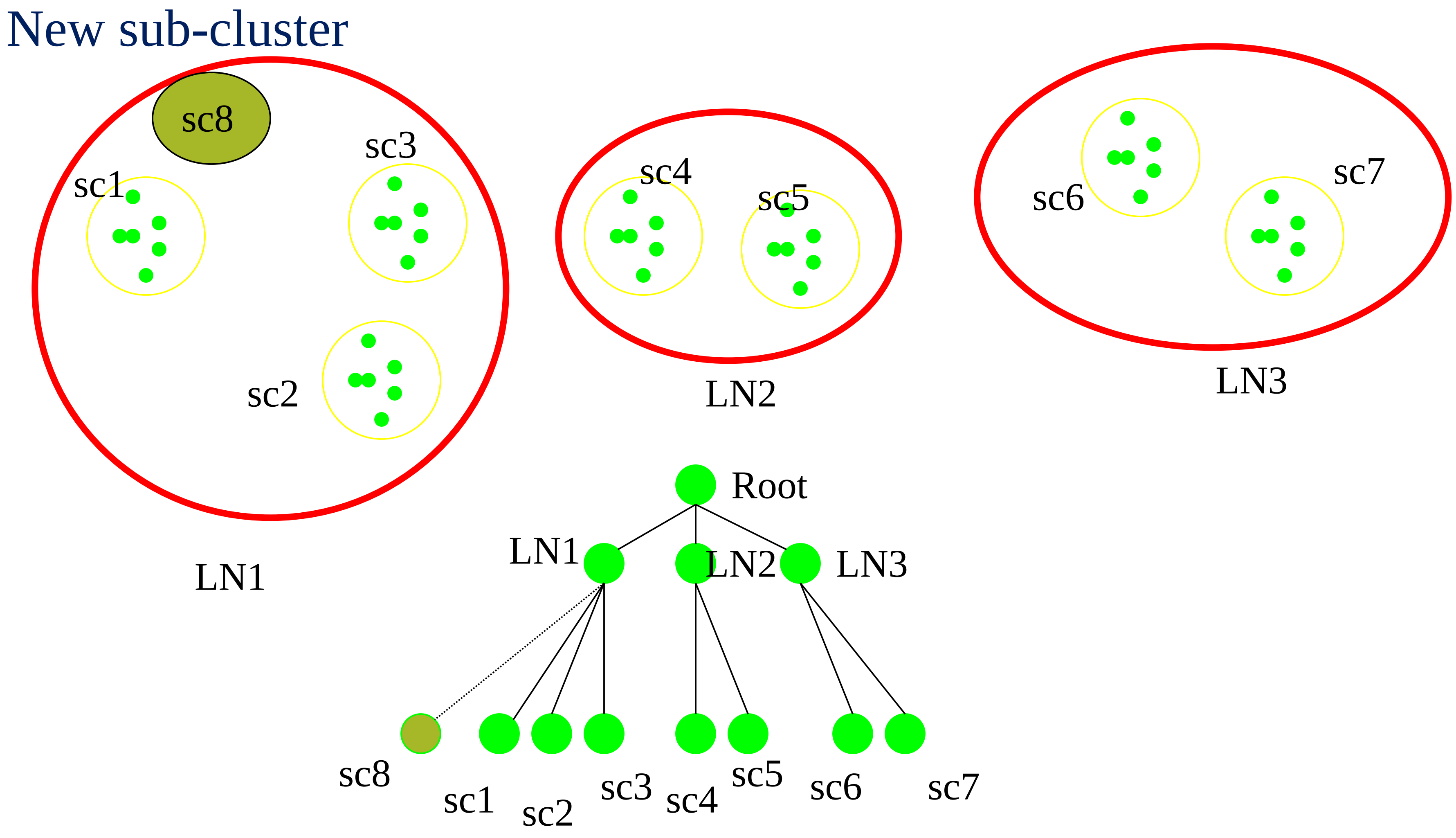


Insertion into a CF Tree (cont.)

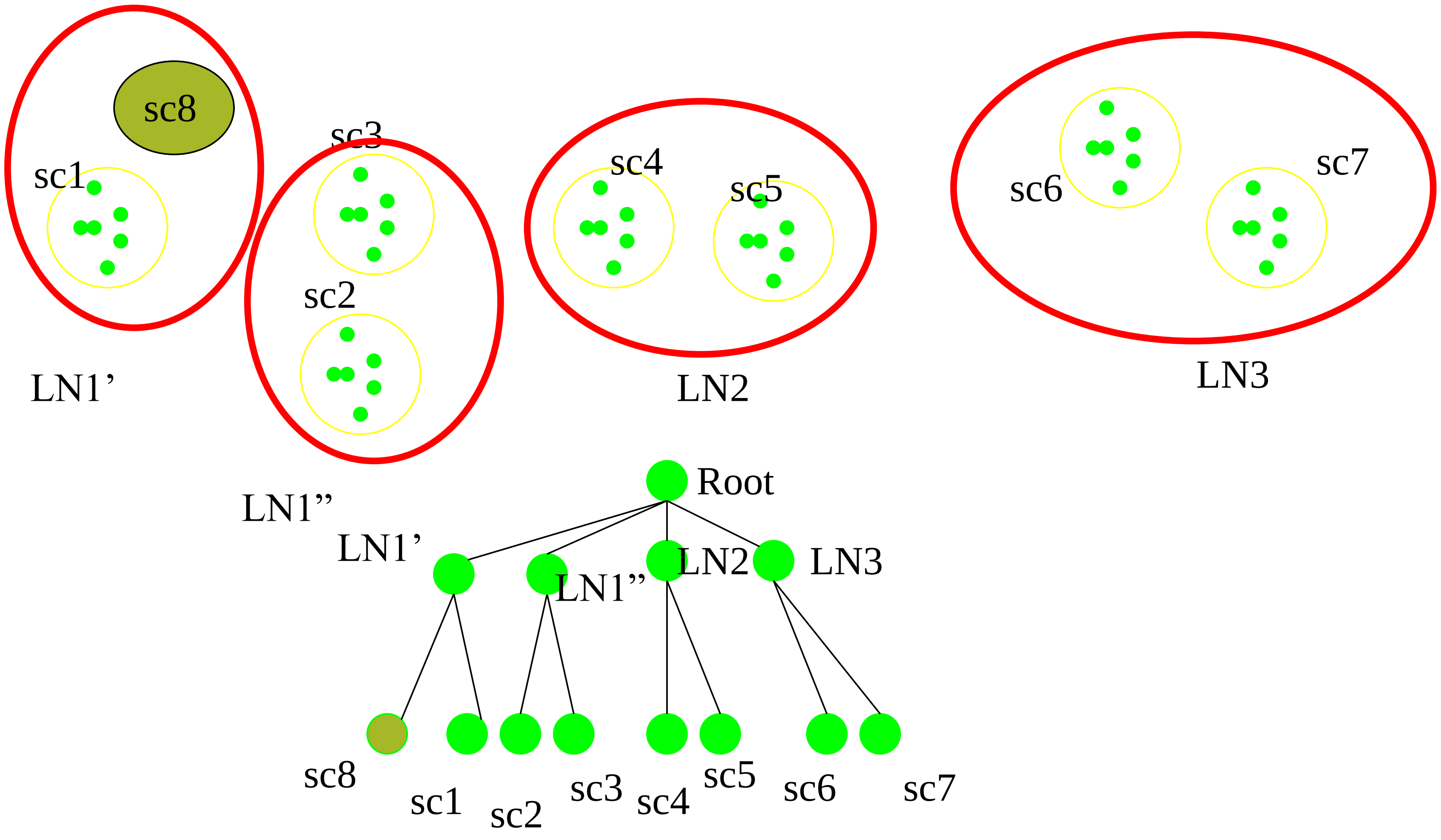
- Modifying the path to the leaf:
  - once object has been inserted, we must update the CF of all ancestors
  - if no node splitting, we only need to update CF vectors
  - a node split requires inserting a new entry into the parent node
  - if parent has space for this entry, we only need to update CF vectors
  - in general, we may need to split the parent (so we do not violate branching factor B) and so on up to the root
  - if root is split, the tree height increases by one



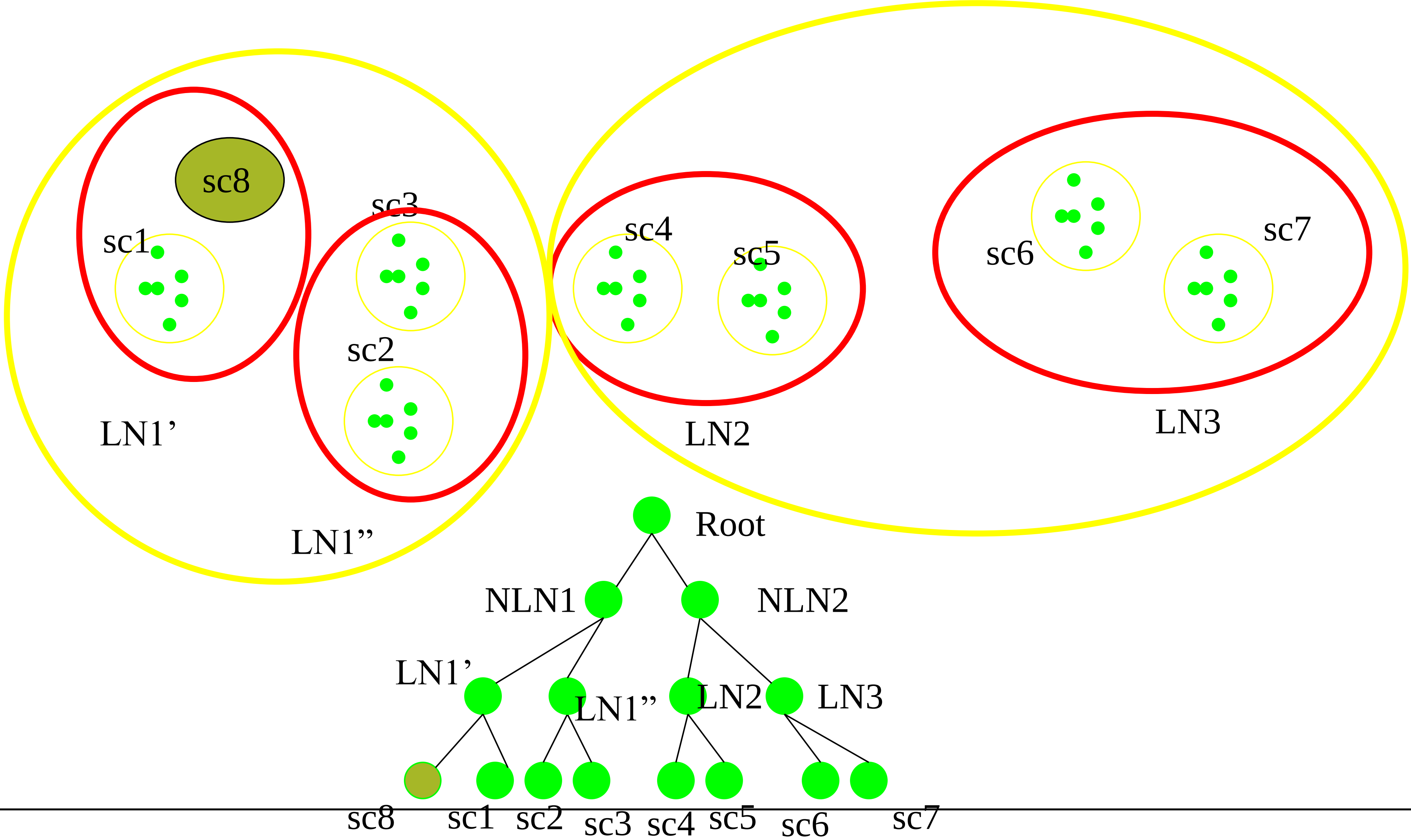
Example of Insertion - Assume that the sub-clusters are numbered according to the order of formation (B=L=3)



If the number of entries of a leaf node (L) can not exceed 3, then LN1 is split.



If the branching factor of a non-leaf node (B) can not exceed 3, then the root is split and the height of the CF Tree increases by one.



Improving the Clusters

- A node in a CF tree can hold a limited number of entries
- Entries can hold limited number of objects
  - If a data point is inserted twice at different times, the two copies may end up in separate clusters
  - Two sub-clusters that should be in one cluster can be split in different nodes
  - Two sub-clusters that should not be in one cluster can end up in same node
- So, a node may not always correspond to a natural cluster
- These problems can be solved by an additional scan of the dataset, which results in better clustering

BIRCH Algorithm

- Phase 1 – Build the CF tree
  - ➢ If there is not enough memory, increase T and build a smaller CF tree
- Phase 2 – Apply any clustering algorithm to cluster the leaf nodes
- Optional phase to improve the clustering accuracy
  - ➢ Scan the whole dataset to re-cluster all data points
  - ➢ A data point is placed in the cluster with the closest centroid

Remarks

- Optional scan solves previously mentioned problems
- BIRCH uses the following ideas to detect outliers:
  - points that are far from any centroid
  - sparse clusters
- BIRCH can handle only numeric data
- BIRCH is a scalable algorithm
- BIRCH does not perform well if clusters are not spherical in shape