BIRCH Clustering Algorithm :

- Existing data clustering methods do not adequately address the problem of processing large datasets with a limited amount of resources (i.e. memory and cpu cycles).
- At a high level, **Balanced Iterative Reducing and Clustering using Hierarchies**, or **BIRCH** for short, deals with large datasets by first generating a more compact summary that retains as much distribution information as possible, and then clustering the data summary instead of the original dataset.
- BIRCH actually complements other clustering algorithms by virtue if the fact that different clustering algorithms can be applied to the summary produced by BIRCH
- BIRCH can only deal with metric attributes (similar to the kind of features KMEANS can handle). A metric attribute is one whose values can be represented by explicit coordinates in an Euclidean space (no categorical variables).

BIRCH Algorithm:

→ Want to cluster really large datasets.

→ Do very rough clustering at the beggining producing very large no of small and tight clusters.

→ No we can take centroid (representatives) of these clusters and once again perform clustering iteration.

→ So we need to have some particular no. of iterations to run through and finally have clusters.

→ But if this could be done in one pass through the dataset then it would save a lot of computational power and time.

→ <u>CFT: Clustering Feature Tree:</u>

If there is a cluster

$$\boxed{C_j = \{x_{1j}, x_{2j}, \ldots, x_{nj}\}}$$

then,

$$\boxed{CF(C_j) = \langle \underbrace{N_j}_{\text{No. of data points}}, \underbrace{\sum_{i=1}^{N} x_i}_{\text{linear sum of N-data points}}, \underbrace{\sum_{i=1}^{N} x_i^2}_{\text{square sum of N-data points}} \rangle}$$

→ Given the clustering feature (CF) of 2 different clusters we can compute the distance b/w 2 clusters, by first obtaining the centroids through $\left(\frac{\sum_{i=1}^{n} x_i}{n}\right)$ for each cluster.

For a particular cluster: $\boxed{CF = (N, LS, SS)}$

cluster statistics:

Given n-dimensional data objects or points in a cluster, we can define centroid (x_0), radius (R) and diameter (D) of the clusters:-

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

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

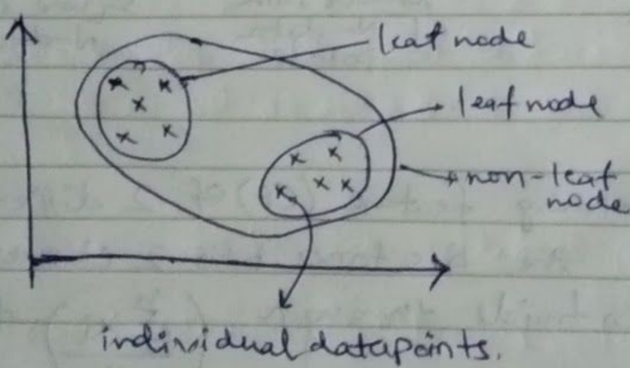
$$\boxed{D = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{n} (x_i - x_j)^2}{n(n-1)}}}$$

R: Average distance from member object to centroid.

D: Average pair wise distance within the cluster.

Both (R) and (D) reflect the tightness of a cluster around a centroid.

Tree:



Clustering feature (CF) and Cluster Feature Tree (CF Tree):

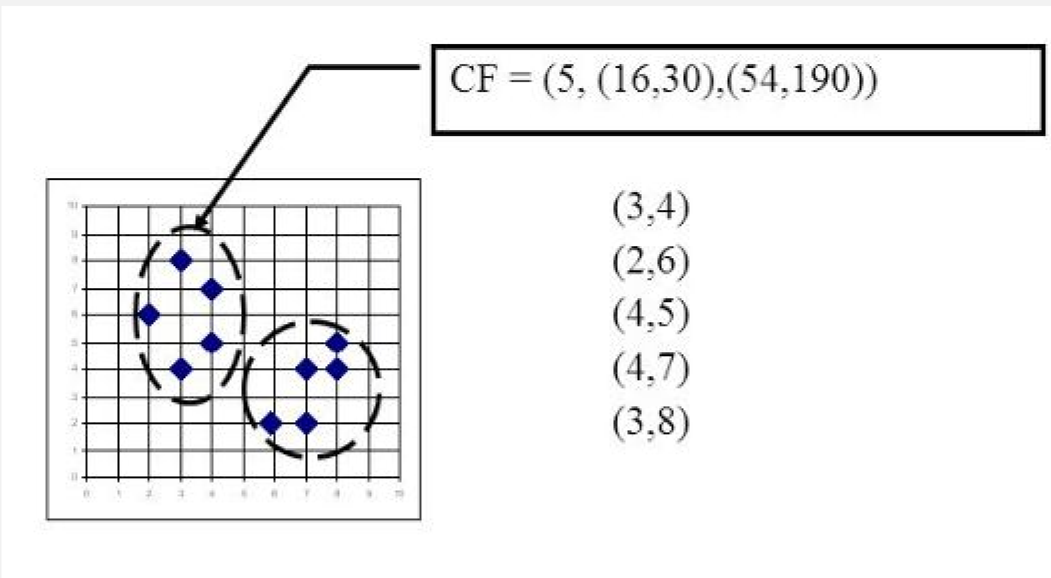
In the clustering feature tree, a clustering feature (CF) is defined as follows:Each CF is a triplet, which can be represented by (N, LS, SS).

- Where N represents the number of sample points in the CF, which is easy to understand
- LS represents the vector sum of the feature dimensions of the sample points in the CF
- SS represents the square of the feature dimensions of the sample points in the CF.

For example, as shown in the following figure, in a CF of a node in the CF Tree, there are the following 5 samples (3,4), (2,6), (4,5), (4,7), (3,8). Then it corresponds to

$$N = 5,\ LS = (3+2+4+4+3,\ 4+6+5+7+8)$$

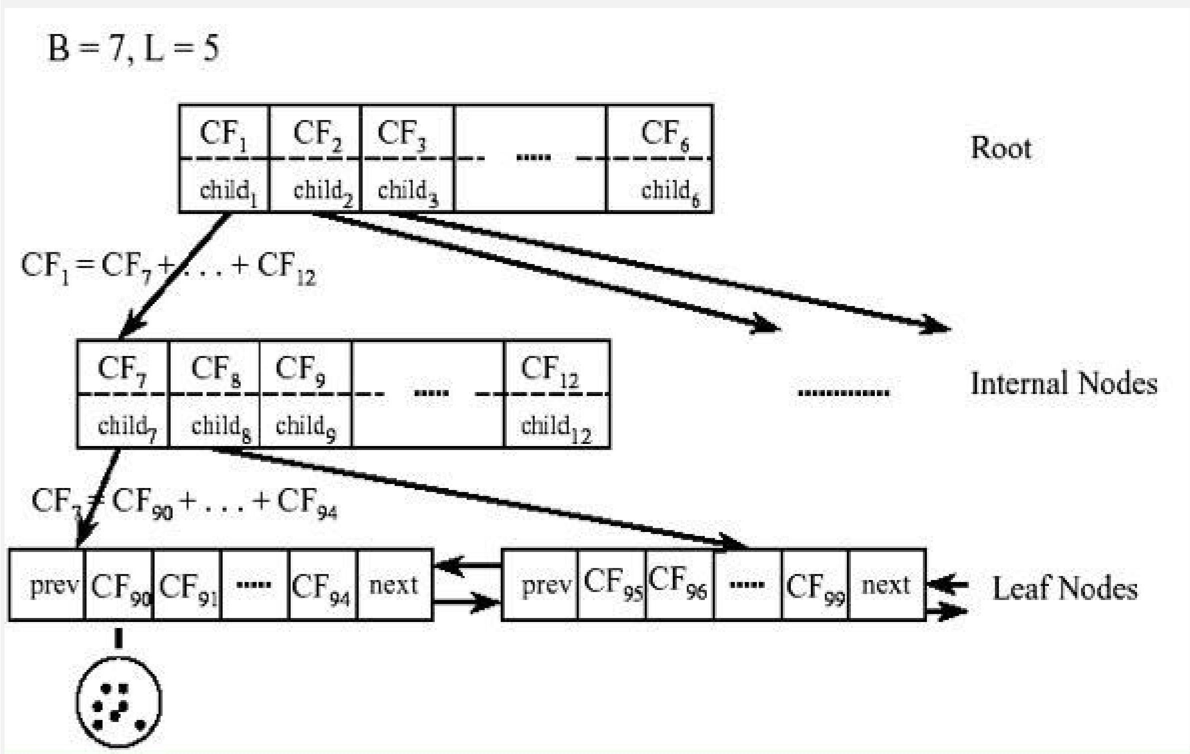
$$SS = 3^2+2^2+4^2+4^2+3^2+4^2+6^2+5^2+7^2+8^2$$



CF has a very good property. It satisfies the linear relationship, that is:

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

This property is also well understood by definition. If you put this property on the CF Tree, that is to say, in the CF Tree, for each CF node in the parent node, its (N, LS, SS) triplet value is equal to the CF node pointed to The sum of the triples of all child nodes.



For CF Tree, we generally have several important parameters,

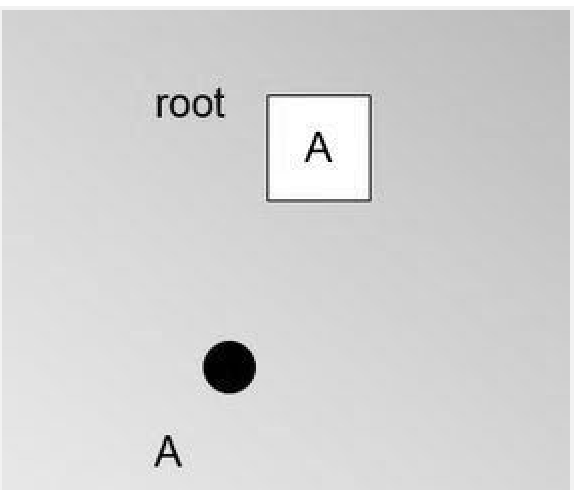
- The first parameter is the maximum CF number B of each internal node,
- The second parameter is the maximum CF number L of each leaf node,
- The third parameter is for the sample points in a CF in the leaf node. It is the maximum sample radius threshold T of each CF in the leaf node. That is to say, all sample points in this CF must be in the radius In a hyper-sphere less than T.

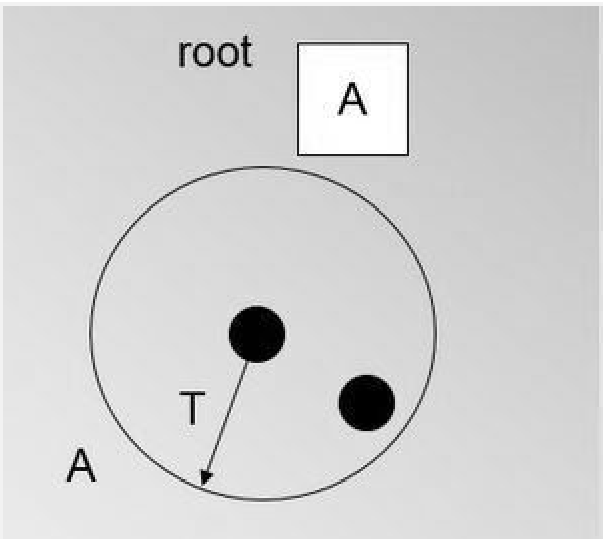
For the CF Tree in the above figure, B = 7 and L = 5 are defined, which means that the internal node has a maximum of 7 CFs, and the leaf node has a maximum of 5 CFs.

Generation of Clustering Feature Tree (CF Tree):

Let us see how to generate CF Tree. We already define the parameters of the CF Tree above

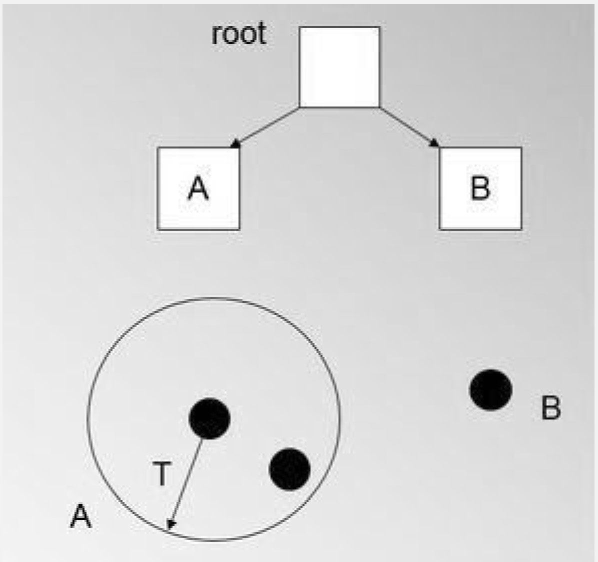
- In the beginning, the CF Tree is empty and there are no samples. We read the first sample point from the training set and put it into a new CF triplet A. That's why N1 initially.



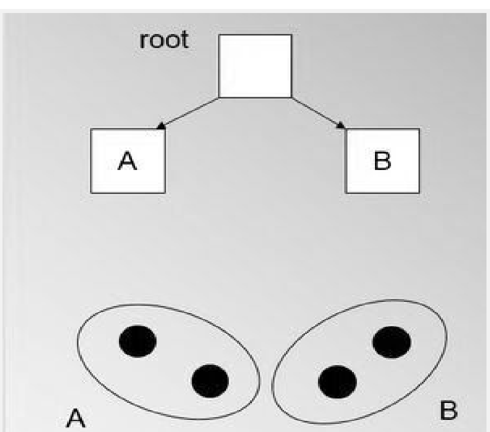
- Now we continue to read the second sample point, we find that this sample point and the first sample point A are within the range of a hyper-sphere with a radius T. Now, N=2 in the triplet of A.
- 

- At this point, the third node came, and we found that this node could not be integrated into the hyper-sphere formed by the previous node, that is, we needed a new CF triple B to accommodate this new value

- At this time, the root node has two CF triples A and B. The CF Tree is as follows:

- 

When we came to the fourth sample point, we found that the radius of B and B is smaller than T.

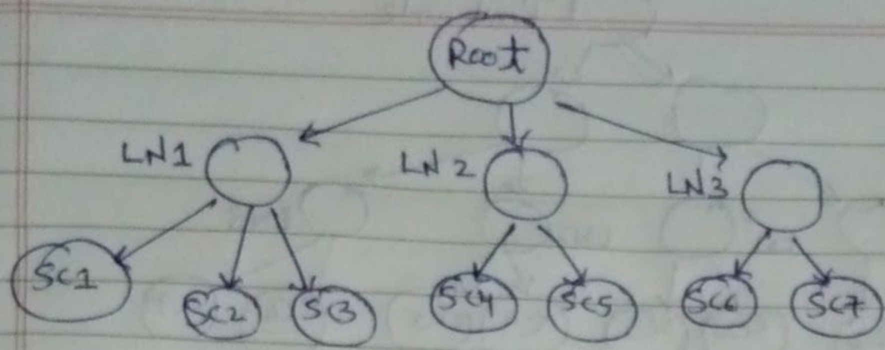
- 

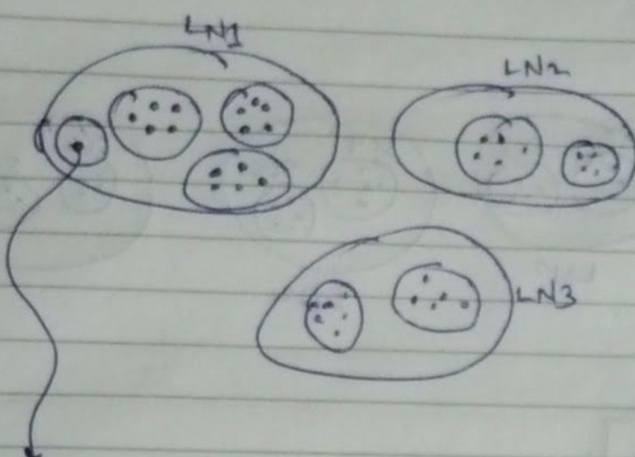
Now when does the node of the CF Tree need to be split?

Suppose our current CF Tree is shown in the following figure. The leaf node LN1 has three CFs, and LN2 and LN3 each have two CFs.

The maximum CF number of our leaf nodes is L = 3. At this time a new sample point is coming, we find that it is closest to the LN1 node, so we start to judge whether it is within the super sphere corresponding to the three CFs of sc1, sc2, sc3

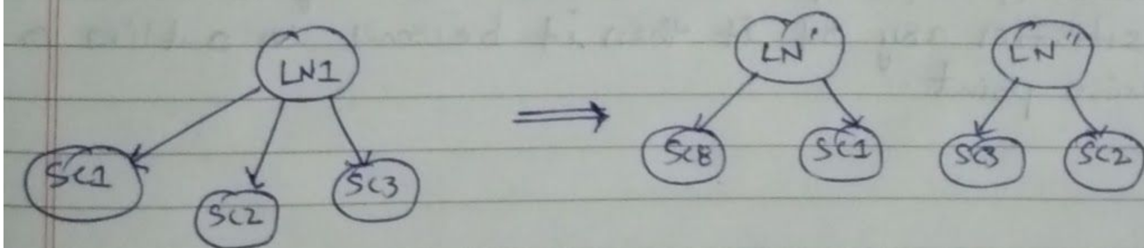
But, unfortunately, it is not, so it needs to create a new CF, sc8, to accommodate it. The problem is that our L = 3, which means that the number of CFs of LN1 has reached the maximum value, and no new CF can be created. What should I do? At this point, it is necessary to split the LN1 leaf node into two.



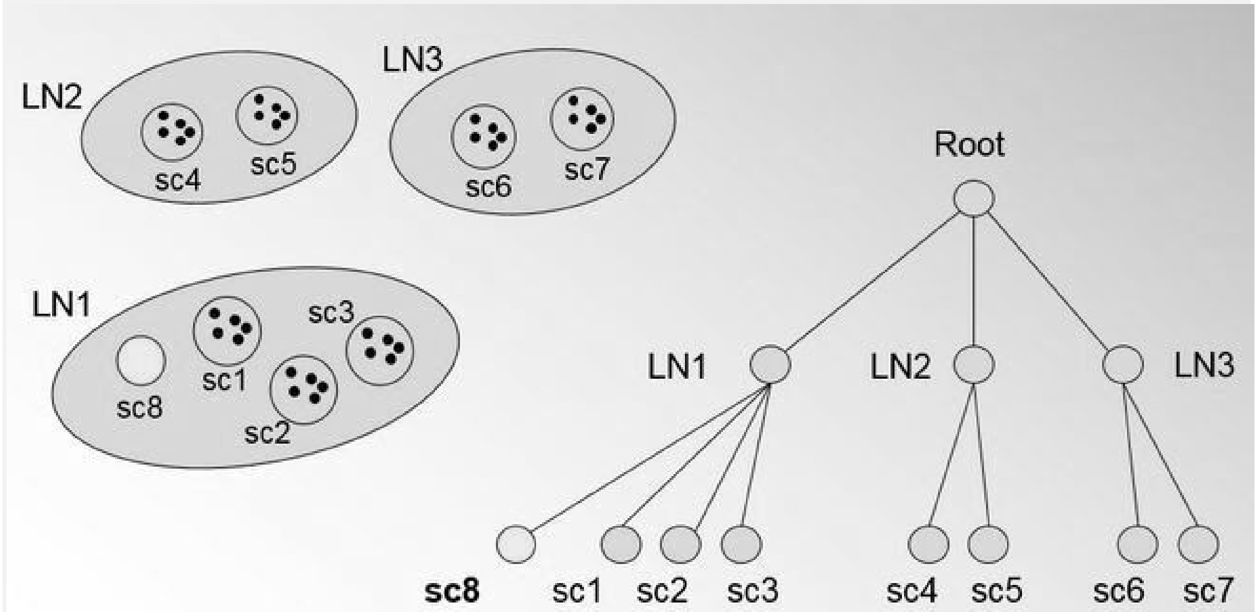


Let say a new data entry made.
But since all clusters already have 5 points in each we need to assign a different cluster.

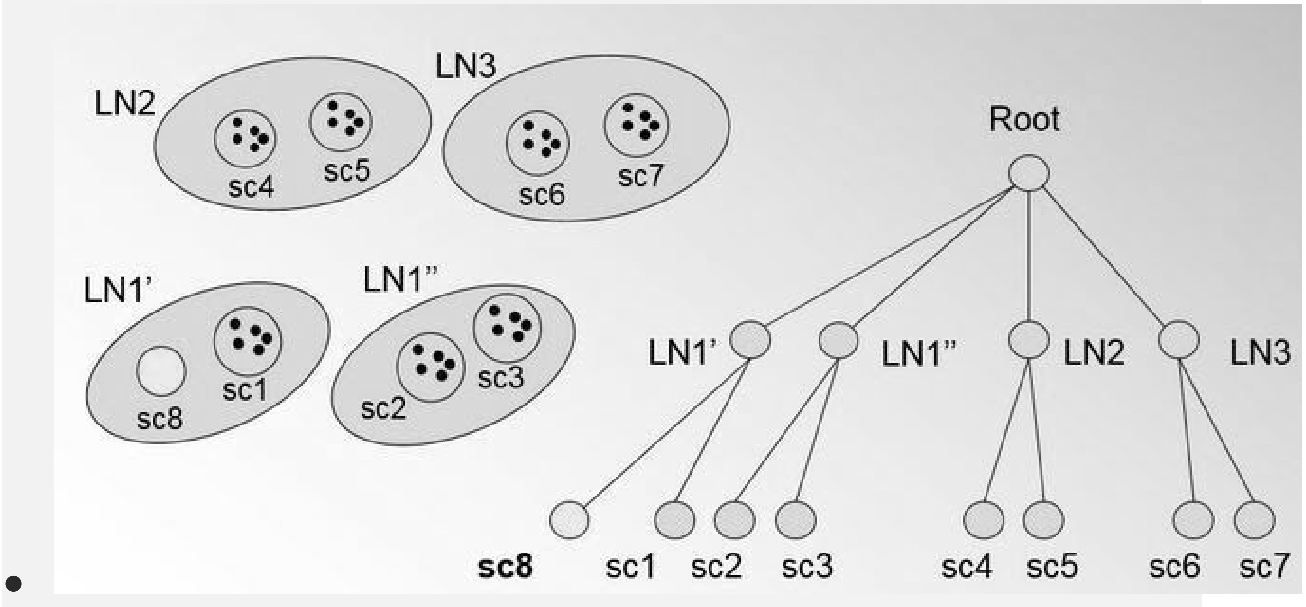
Let say we had $B=3, L=3$
But now in LN1 we have 4 clusters.
So splitting:



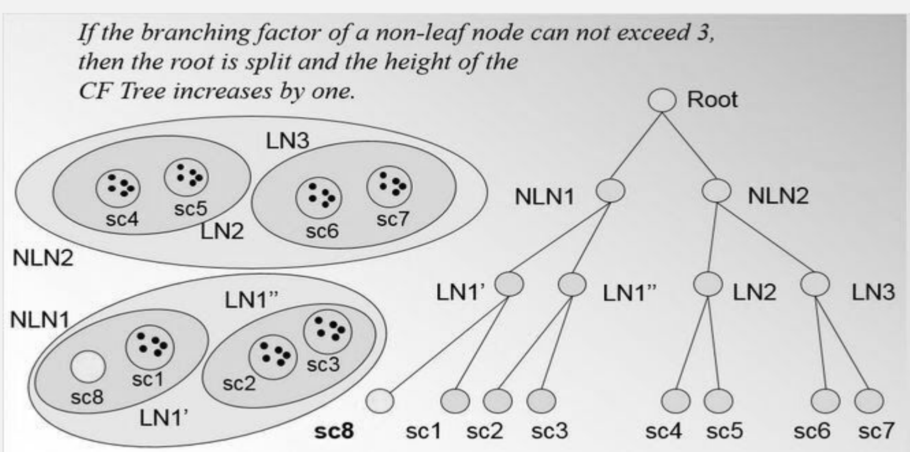
But now the condition of $B=3$ (Non-leaf node) is violated hence again splitting takes place.

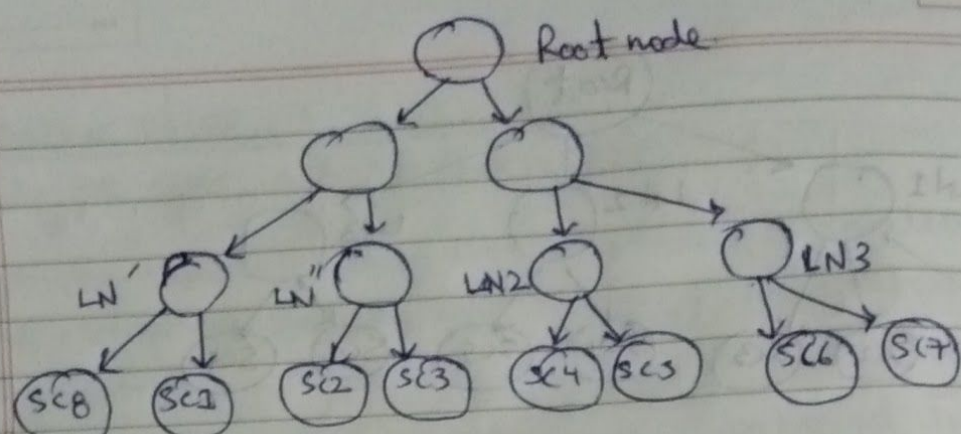


- We will find the two furthest CFs in all CF tuples in LN1 as seed CFs of these two new leaf nodes, and then we will save all sc1, sc2, sc3 in the LN1 node, and the new tuple sc8 of the new sample point Divide it into two new leaf nodes.

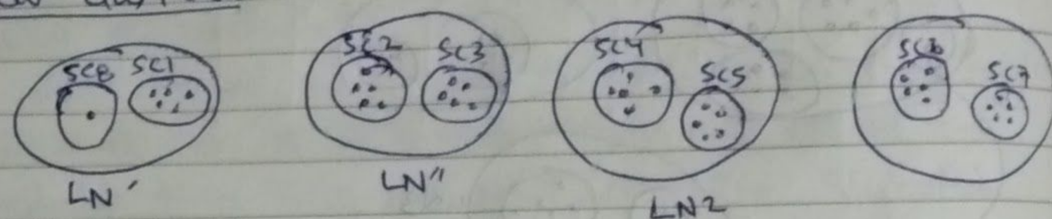
- 

If the maximum CF number of our internal nodes is B = 3, then splitting the leaf node into two will cause the maximum CF number of the root node to be exceeded, that is to say, our root node will now also split, the split method and The leaf nodes are split, and the split CF Tree is as follows:





New clusters:



Additive Property:

$$\boxed{CF_1 + CF_2 = CF_{net}}$$

here, $\boxed{CF_{LN'} = CF_{SC8} + CF_{SC1}}$

$$CF_{net} = (N_1 + N_2, LS_1 + LS_2, SS_1 + SS_2)$$

⇒ There is a threshold set for each type of cluster (Root) Non-leaf and leaf and, if the data entry does not satisfies any of it then it becomes an outlier or noise point.

1. Given a dataset of N-points
2. In first phase ⇒ goes through all datapoints and makes cluster feature tree

Cluster feature for N-points. means

$CF = (N, LS, S.S)$

$(2,2).(3,3).(4,4)$ → $N=3$

$LS = (9,9)$

$SS = (29,29)$

(CF) are organized in a CF tree.

2 parameters are there (B) and (T).

B → Branching factor

T → Threshold value

(3) Each node represents a cluster in hierarchy.

Intermediate nodes → super clusters
leaf nodes → actual clusters.

B: <u>Branching factor</u>:
maximum of children a node can have.

(4) From each node/cluster, cluster radii and center are calculated.

(5). Now each new point stands at the root and walks down the tree entering the sub-cluster with nearest center until ends at a certain leaf node

6. Once arrived at leaf node, new point is added to this cluster;

Given that the addition of this point does not increase the cluster radius value beyond a threshold (T).

7. If R_{new} > (T), then this point is not included in the cluster and a new cluster is created with this point as its only member.

⇓

This way parameter (T) Threshold controls the size of cluster.

8. Now if creation of this new cluster leads to more child nodes than (Br) factor then parent node is split.

9. Once all datapoints are submitted to birch centres of the final leaf clusters are in Global clustering phase where clustering algorithm such as agglomerative or K-means is used and parameter (K) to be input

(10.) Types of birch:

Full-Birch

Global clustering phase of leaf cluster centre occurs.
Basically this is the step of merging of cluster.
Given points (leaf cluster centre) and applying K-means.
After this process the final no. of clusters is what we desire.
Takes input from user for no. of cluster

Tree-Birch

No global clustering step performed.
Sub-clusters are returned the way they are
Here the no. of cluster is very large (500-600).
Does not take input from user for no. of clusters.

LSA, LDA, Birch

Summarize the insertion of CF Tree:

1. Find the **leaf node** closest to the new sample and the **closest CF node** in the leaf node from the root node
2. After the new sample is added, if the radius of the hyper-sphere corresponding to this CF node still satisfies the threshold T, **then all the CF triplets on the path are updated, and the insertion ends.** Otherwise, go to 3.
3. If the number of CF nodes of the current leaf node is less than the threshold L, create a new CF node, put in a new sample and the new CF node into this leaf node, update all CF triplets on the path, and insertion Ends. Otherwise, go to 4
4. Divide the current leaf node into two new leaf nodes, select the two CF tuples with the farthest hyper-sphere among all CF tuples in the old leaf node, and distribute as the first CF node of the two new leaf nodes. Put other tuples and new sample tuples into corresponding leaf nodes according to the principle of distance.

5. In turn, check whether the parent node is also to be split. If it needs to be split in the same way as the leaf node.

Summary

- The BIRCH algorithm **does not need to input the K value of the number of categories**, which is different from K-Means and Mini Batch K-Means.
- In addition to clustering, BIRCH can also do some additional outlier detection and preliminary data pre-processing according to category specifications.
- If the dimension of the data features is very large, such as greater than 20, BIRCH is not suitable. **At this time, Mini Batch K-Means performs better.**
- The complexity of the algorithm is O(n).
- *For parameter adjustment, BIRCH is more complicated than K-Means and Mini Batch K-Means, because it needs to adjust*

several key parameters of CF Tree, which have a great influence on the final form of CF Tree.

Advantages

1. Save memory, all samples are on disk, CF Tree only stores CF nodes and corresponding pointers.
2. **The clustering speed is fast**, and it only takes one scan of the training set to build the CF Tree, and the addition, deletion, and modification of the CF Tree are very fast.
3. Noise points can be identified, and preliminary classification pre-processing can be performed on the data set.

Disadvantages

1. Since CF Tree has a limit on the number of CFs per node, the clustering result may be different from the real category distribution.

2. The data clustering effect is not good on high-dimensional features. At this time, you can choose Mini Batch K-Means.
3. If the distribution cluster of the data set is not similar to a hyper-sphere or is not convex, the clustering effect is not good.

implementation:

https://colab.research.google.com/drive/1yqblvPsOy4IYTO9UECEL-HsAgu5p5sN2?usp=sharing