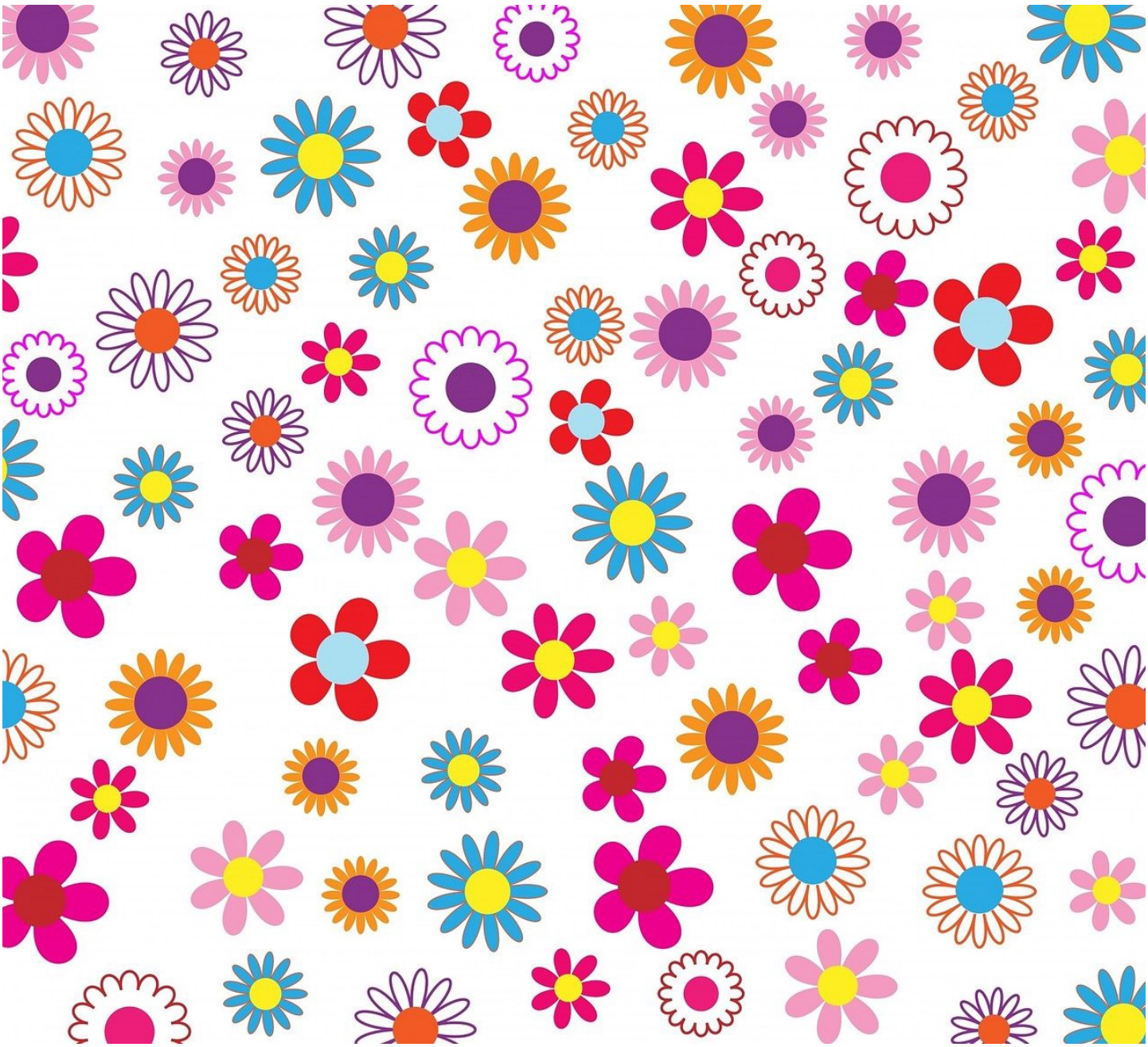
BIRCH Clustering Clearly Explained

Informative guide of Best Practices of BIRCH Clustering and discussion about insertion in CF-Tree with advantages and disadvantages

Pawan Jain
May 10 · 9 min read ★



source: kaz, via pixabay

Principle of BIRCH clustering algorithm

The BIRCH algorithm is more suitable for the case where the amount of data is large and the number of categories K is relatively large. It runs very fast, and it only needs a single pass to scan the data set for clustering. Of course, some skills are needed. Below we will summarize the BIRCH algorithm.

BIRCH overview

BIRCH stands for ***Balanced Iterative Reducing and Clustering Using Hierarchies***, which uses hierarchical methods to cluster and reduce data.

- BIRCH only needs to scan the data set in a single pass to perform clustering.

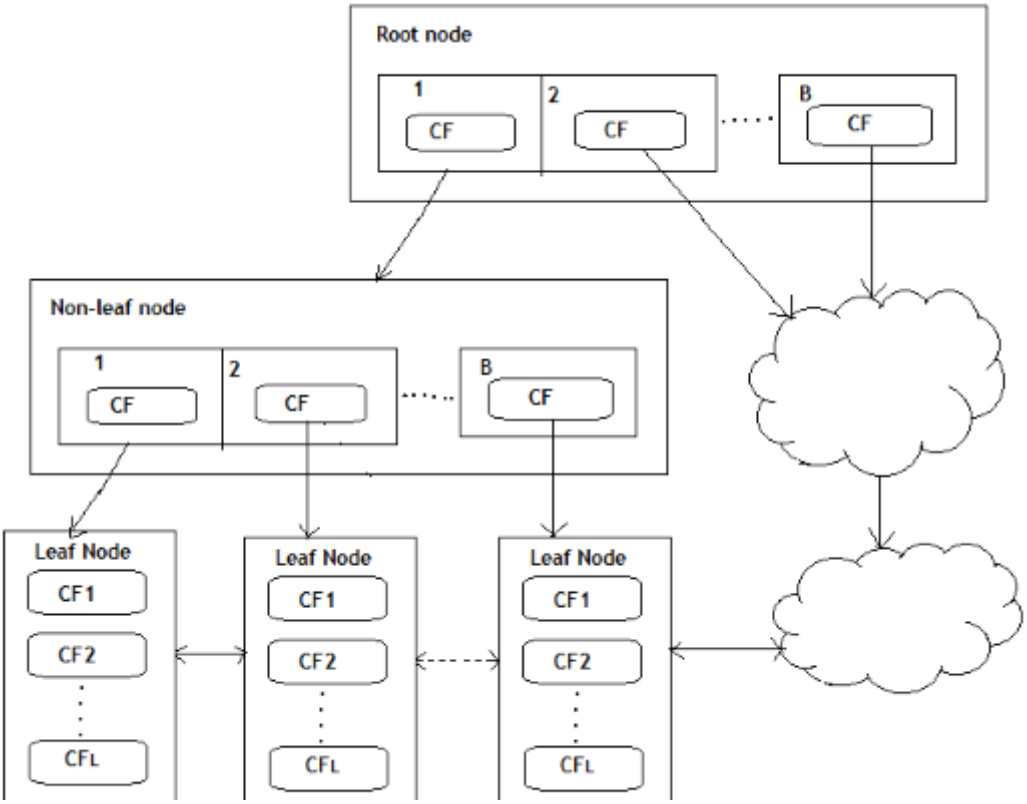
How does it work?

The BIRCH algorithm uses a tree structure to create a cluster. It is generally called the Clustering Feature Tree (CF Tree). Each node of this tree is composed of several Clustering features (CF).

> ***Clustering Feature tree structure is similar to the balanced B+ tree***

From the figure below, we can see what the clustering feature tree looks like.

Each node including leaf nodes has several CFs, and the CFs of internal nodes have pointers to child nodes, and all leaf nodes are linked by a doubly linked list.



From Research Paper by Nidal, Mahmoud & Wesam

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

In the clustering feature tree, a clustering feature (CF) is defined as follows:

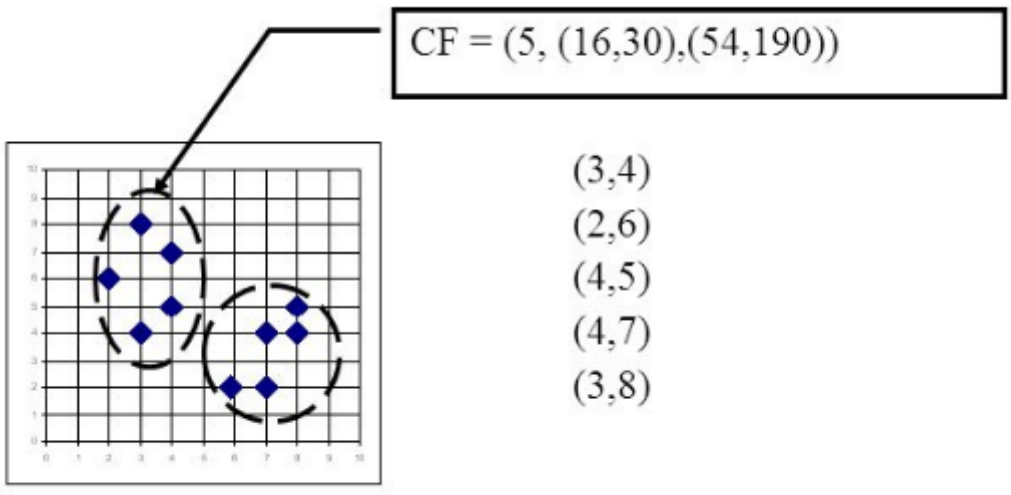
Each CF is a triplet, which can be represented by (N, LS, SS).

- 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

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

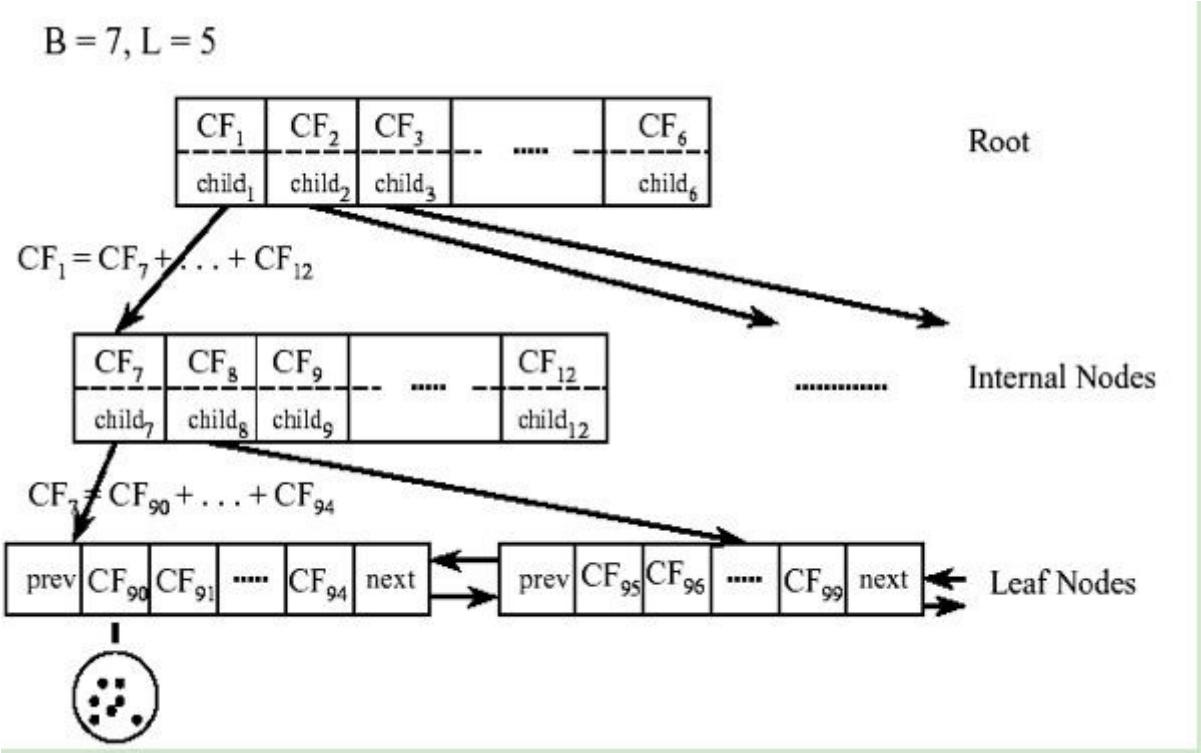
$$SS = 3^2+2^2+4^2+4^2+3^2+4^2+6^2+5^2+7^2+8^2$$



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.



From notes by By T, Zhang, R. Ramakrishnan

As can be seen from the above figure, the value of the triplet of CF1 of the root node can be obtained by adding the values of the 6 child nodes (CF7-CF12) that it points to. In this way, we can be very efficient when updating the CF Tree.

For CF Tree, we generally have several important parameters,

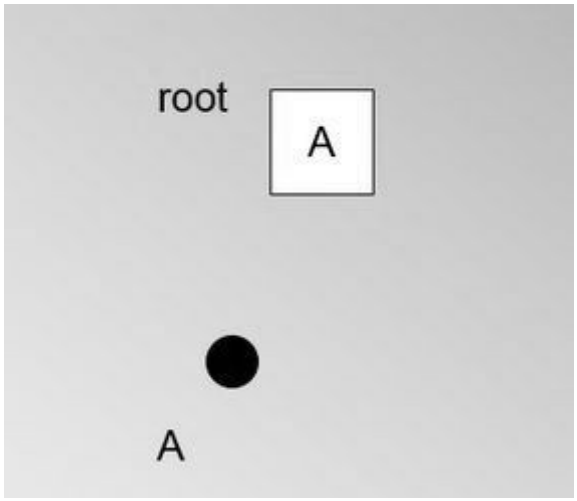
- The first parameter is the maximum CF number B of each internal node,
- The second parameter is the maximum CF number L of each leaf node,
- The third parameter is for the sample points in a CF in the leaf node. It is the maximum sample radius threshold T of each CF in the leaf node. That is to say, 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.

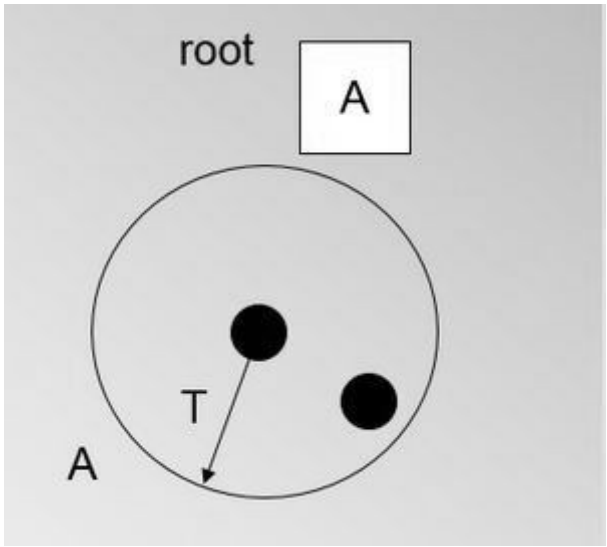
Generation of Clustering Feature Tree (CF Tree)

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.

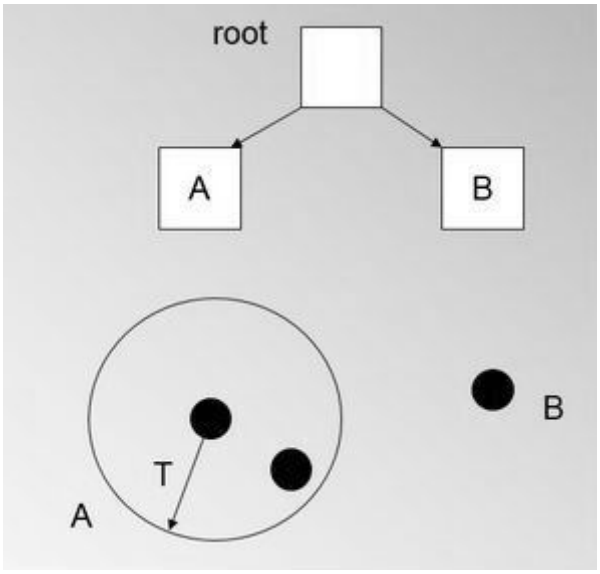


- Now we continue to read the second sample point, we find that this sample point and the first sample point A are within the range of a hyper-sphere with a radius T. Now, N=2 in the triplet of A.



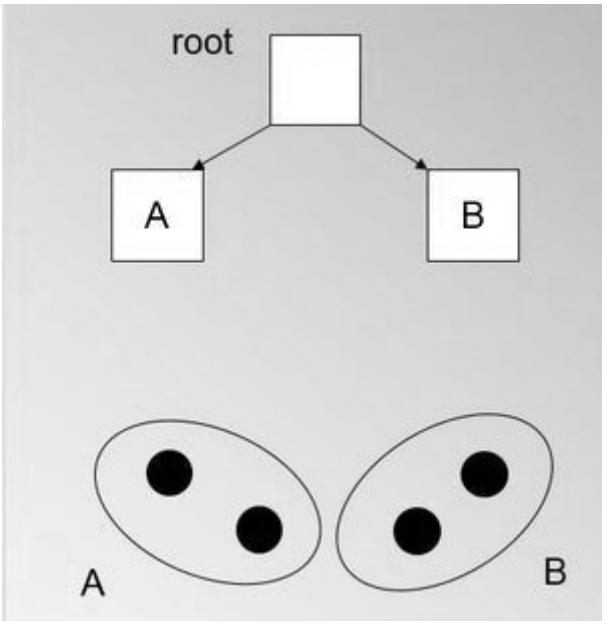
- At this point, the third node came, and we found that this node could not be integrated into the hyper-sphere formed by the previous node, that is, we needed a new CF triple B to accommodate this new value.

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



Source

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



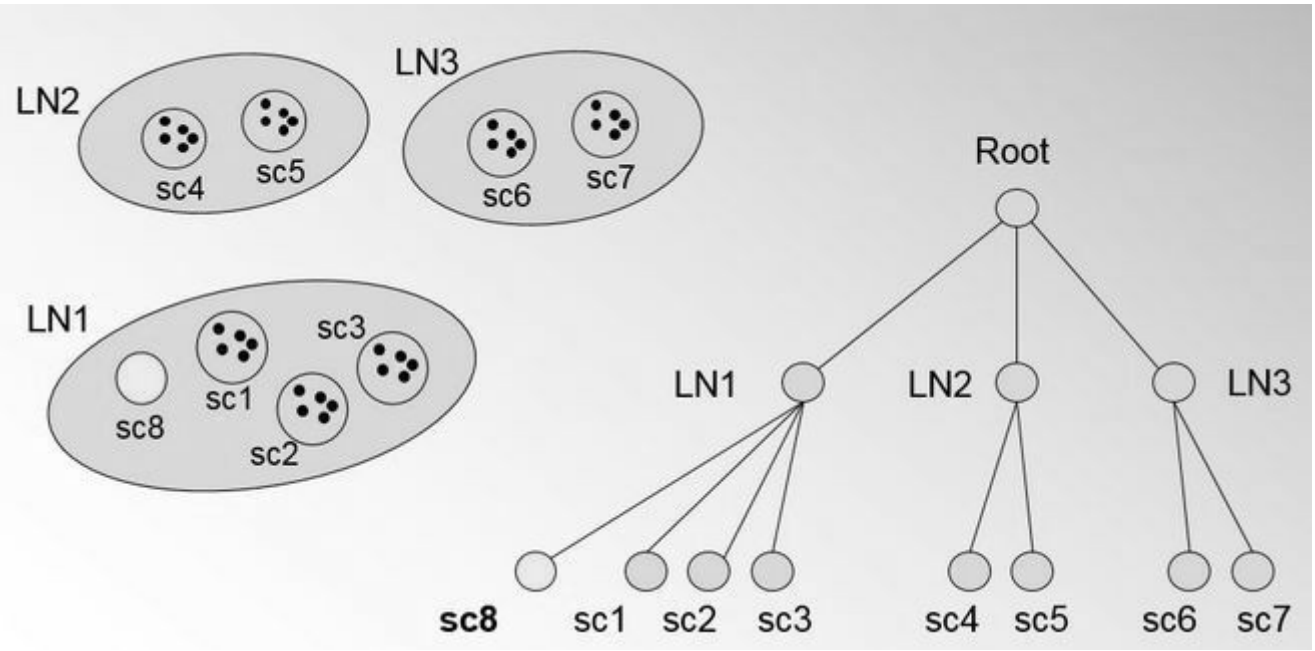
Source

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.

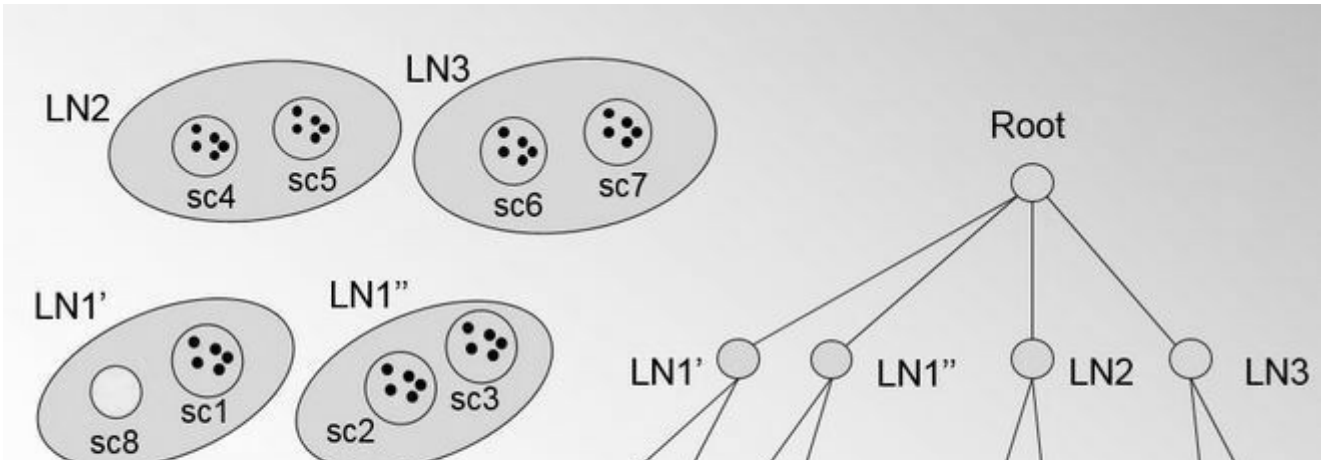
The maximum CF number of our leaf nodes is L = 3. At this time a new sample point is coming, we find that it is closest to the LN1 node, so we start to judge whether it is within the super sphere corresponding to the three CFs of sc1, sc2, sc3

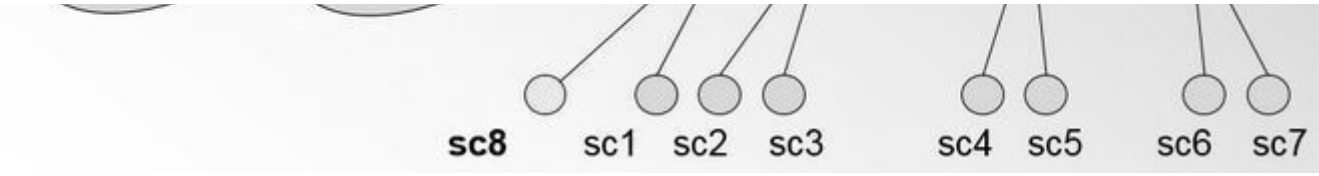
But, unfortunately, it is not, so it needs to create a new CF, sc8, to accommodate it. The problem is that our L = 3, which means that the number of CFs of LN1 has reached the maximum value, and no new CF can be created. What should I do? At this point, it is necessary to split the LN1 leaf node into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



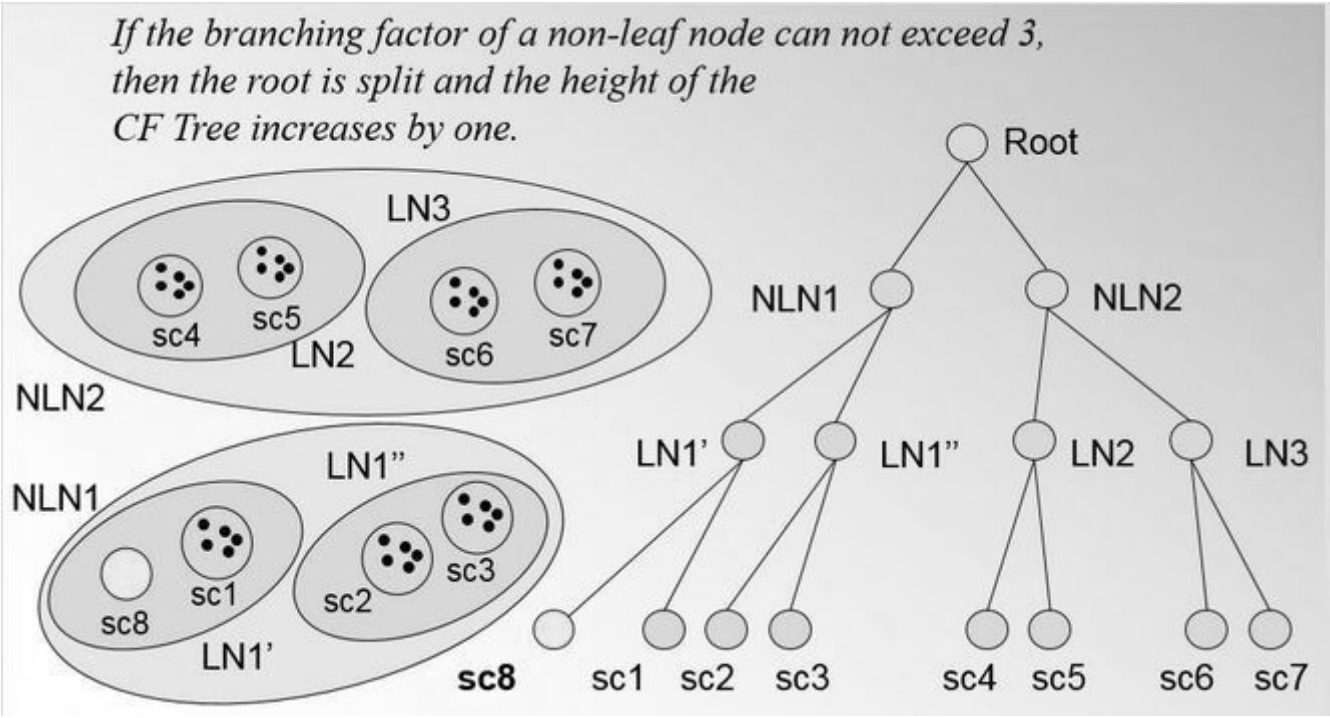
Source

- 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:



Source

Summarize the insertion of CF Tree

So I think that the whole process is somewhat tedious. Hence I decided to summarize it as follows

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.

BIRCH algorithm

All the training set samples are built into a CF Tree, and a basic BIRCH algorithm is completed. The corresponding output is several CF nodes, and the **sample point in each node is a cluster.**

> *In other words, the main process of BIRCH algorithm is the process of establishing CF Tree.*

In addition to building a CF Tree to cluster the real BIRCH algorithm, there are some optional algorithm steps. Now let's take a look at the process of the BIRCH algorithm.

- Read all the samples in sequence and create a CF Tree in memory. For the method of creation, refer to the previous section.

- (Optional) Filter the CF Tree created in the first step to remove some abnormal CF nodes.

> *Abnormal in the sense nodes having few sample points*

- (Optional) Use other clustering algorithms such as K-Means to cluster all CF tuples to get a better CF Tree.

The main purpose of this step is to eliminate problems caused by some tree structure splits **due to the limitation of the number of CF nodes**.

- (Optional) Using the centroids of all CF nodes of the CF Tree generated in the third step as initial centroid points, cluster all the sample points according to distance.

> As can be seen from the above, the key of the BIRCH algorithm is step 1, which is the generation

of CF Tree, and the other steps are to optimize the final clustering result.

Algorithm 1 BIRCH algorithm

Input: M data points
Output: N subclusters
repeat
Calculate the cluster feature $\vec{CF}$ of data points
Insert record to construct CF tree
until Insert all records into the CF tree

From Research Paper by Tang D., Dai R., and Tang, L.

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.
- If you do not enter the K value, the number of the last CF tuple is the final K, otherwise, the CF tuple will be merged according to the distance
- 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.

. . .

Other Clustering Algorithms to learn

Yes, we have made another clustering algorithm i.e. BIRCH Clustering algorithm.

GIF by Giphy

If you want to learn about Clustering and other clustering algorithms

* For understanding Clustering : **Clustering Clearly Explained**

* For clearing concepts of DBSCAN : **DBSCAN Clustering Best Practices**

* For Heirarchial Clustering : **Hierarchical clustering Clearly Explained**

* For GMM best practices : **Best clustering practice: Gaussian mixture model (GMM)**

Thank you for reading. Don't hesitate to stay tuned for more!