BIRCH Algorithm with working example

Vipul Dalal · Follow
9 min read · Feb 23, 2022

Listen Share

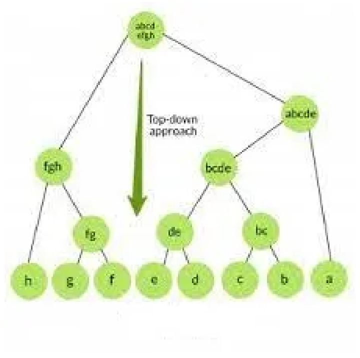
In Data Mining and Machine Learning domains, Clustering refers to the process of grouping the given objects together based on their similarity or dissimilarity. It is unsupervised learning process in which we have only unlabeled data available to us and we try to group them using a list of features into different categories (no predefined categories/labels/classes as opposed to the Classification process). There are many different types of Clustering algorithms are available. Some of the important types are:

1. Partitioning Based clustering
2. Hierarchical clustering
3. Density based clustering
4. Distribution based clustering
5. Fuzzy clustering and so many others

Today, we'll be talking about BIRCH algorithm which comes under the category of Hierarchical clustering. It is bit difficult to understand the working of this algorithm but I'll explain it with a simple example, step-by-step.

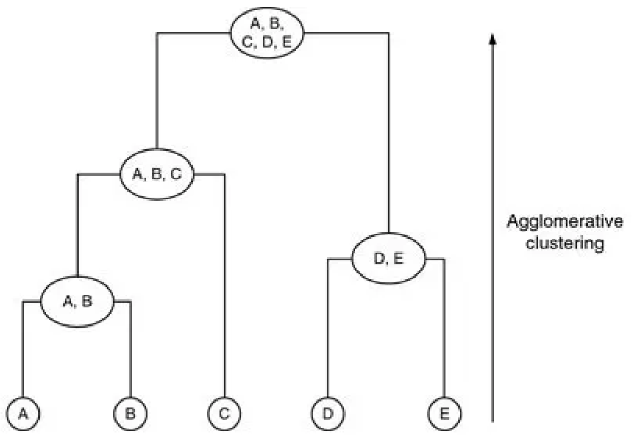
In Hierarchical clustering, as you might have already guessed, a hierarchy or a tree structure is created to represent the clustering process. A tree structure can be created either in top-down (root to leaf nodes) or in bottom-up (leaf nodes to root) fashion. So, accordingly, the Hierarchical clustering is either called Divisive (which follows top-down approach) or Agglomerative (which follows bottom-up approach) clustering.

In Divisive (or top-down) approach, we start with the given set of objects as a root node (single big cluster) of the tree and we go on decomposing or subdividing this node into smaller and smaller sub-clusters to form lower level nodes until all the nodes are left with single object or given termination criterion met.



Divisive Approach

In Agglomerative (or bottom-up) approach, we start with each object in its own cluster (leaf nodes) and we go on merging them into larger and larger clusters to form higher level nodes until all the objects are merged into a single node (root) or given termination criterion met.



Agglomerative Approach

The Agglomerative approach is relatively easy to implement because we can simply go on merging the clusters (nodes) having the smallest distance (during each iteration) into larger clusters. The distance between the clusters can be based on single link, complete link or average link approach. This will satisfy the main goal of the clustering process — minimize intra-cluster distance and maximize inter-cluster distance. But, there two main issues — first, inability to undo what was done in previous step, that is, once two clusters are merged into a single single cluster in a particular iteration then this cannot be undone in any further iterations. And second, scalability, that is, Agglomerative approach doesn't scale well.

The Divisive approach is difficult to implement, mainly because of two challenges — first, how to decide splitting criterion, and second, having a set of clusters at particular level in tree, which cluster to split. For both these challenges, the answer is select a cluster to split and split it such that error is minimized. But, it is not trivial.

The BIRCH algorithm solves these challenges and also overcomes the above mentioned limitations of agglomerative approach.

BIRCH stands for Balanced Iterative Reducing & Clustering using Hierarchy. It is multi-phase hierarchical clustering based on Clustering Features (CFs). It is designed for clustering large amount of numeric data by integrating hierarchical clustering in initial phase, called micro-clustering and other clustering methods in later phase, called macro-clustering.

The Clustering Features (CFs)

The most important characteristic of this algorithm is that it uses CF to summarize a cluster and CF tree to represent the clustering hierarchy. So, let’s first see what Clustering Feature (CF) is.

The Clustering Feature (CF) of a cluster is a 3-D vector summarizing information about clusters of objects. It is defines as,

CF = (n, LS, SS)

where n is the number of objects in the cluster, LS is the linear sum of the objects and SS is the squared sum of the objects.

$$LS = \sum_{i=1}^{n} x_i \quad \text{and} \quad SS = \sum_{i=1}^{n} {x_i}^2$$

For example, consider a cluster C1={9,12,10,8,11} then CF(C1)=(5,50,510) where n=5, LS=9+12+10+8+11=50 and SS=$9^2+12^2+10^2+8^2+11^2$=510

Another example with 2-D objects, C2={(1,1),(2,1),(3,2)} then CF(C2)=(3,(6,4),(14,6)) where n=3,LL=(1+2+3,1+1+2)=(6,4) and SS=($1^2+2^2+3^2$, $1^2+1^2+2^2$)=(14,6)

Using CF, we can derive many other useful statistics of a cluster as given below,

Cluster’s centroid, $$X_0 = \frac{LS}{n}$$

Cluster’s radius, $$R = \sqrt{\frac{\sum_{i=1}^{n}(x_i - X_0)^2}{n}} = \sqrt{\frac{n(SS) - 2LS^2 - n(LS)}{n^2}}$$

Cluster’s diameter, $$D = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\left(x_i - x_j\right)^2}{n(n-1)}} = \sqrt{\frac{2n(SS) - 2(LS)^2}{n(n-1)}}$$

The algorithm improves space efficiency as it needs to store only CFs of the clusters instead of detailed information about the individual objects within the clusters.

Another important property of the CFs is that they are additive. That is, two disjoint clusters C1 and C2 with CFs CF1=(n1,LS2,SS1) and CF2=(n2,LS2,SS2) respectively, the CF of the cluster formed by merging C1 and C2 is given as, CF1+CF2=(n1+n2,LS1+LS2,SS1+SS2)

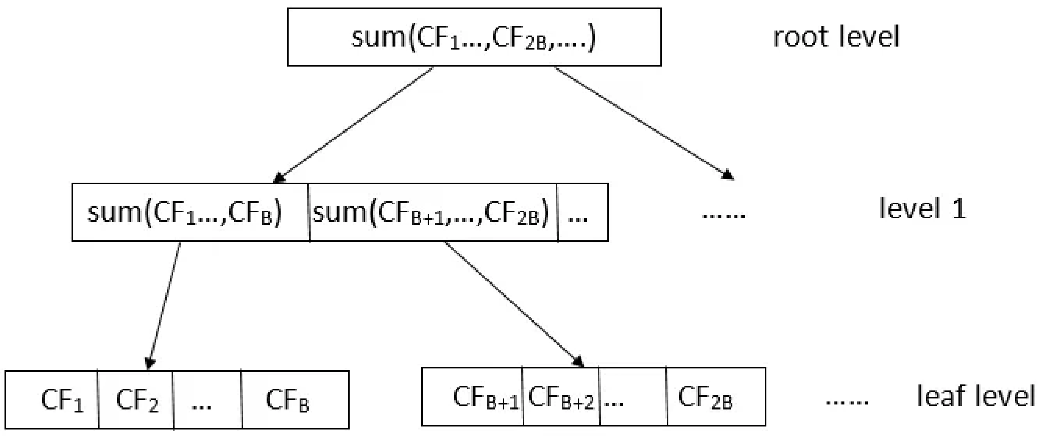
For example, C1={(2,5),(3,2),(4,3)} and C2={(1,1),(2,1),(3,1)} then

CF1=(3,(2+3+4,5+2+3),($2^2+3^2+4^2$,$5^2+2^2+3^2$))=(3,(9,10),(29,38)) and CF2=(3,(1+2+3,1+1+1),($1^2+2^2+3^2$,$1^2+1^2+1^2$))=(3,(6,3), (14,3)) now, if C3=C1UC2 then CF3=CF1+CF2=(6,(15,13),(43,41))

The CF tree

A CF tree is used to store CFs for hierarchical clustering. Let's see how it is created and maintained.

Each leaf node of CF tree stores CFs for a fixed number of clusters. Each non-leaf node stores sum of CFs of its child nodes.



A CF tree has two important parameters — Branching factor (denoted by B) and Diameter threshold (denoted by T). The Branching factor specifies the maximum number of children per non-leaf node and the Diameter threshold specifies the maximum diameter of sub-clusters stored at the leaf node. The overall size of the tree depends up on these two parameters.

A single scan of data creates basic clustering and one or more additional scans can optionally be used to improve the quality of the clusters.

The Phases

There are two primary phases of the algorithm:

Phase 1 — The algorithm scans the objects and constructs an initial in-memory CF tree, which can be viewed as multilevel compression of the data that tries to preserve the inherent clustering structure of the data.

Phase 2 — The algorithm uses a selected clustering method to cluster the leaf nodes of the CF tree.

During Phase 1, objects are dynamically inserted to build the CF tree. An object is inserted into the closest leaf entry. If the diameter of the sub-cluster stored at the leaf node, after insertion, is greater than the specified threshold T then the leaf node is split. After inserting the current object successfully, the information about the inserted object is passed towards the root of the tree. If the memory required to store the CF tree is larger than the available memory then the diameter threshold T is updated (with larger value) and the tree is rebuilt.

Once the CF tree is built, any cluster method, such as partitioning method, can be used with CF tree in Phase 2, to obtain the final clusters.

The most important advantage of this algorithm is that its time complexity is only O(n), where n is the number of objects.

Its disadvantage is that it works well only for spherical shape clusters and numeric attributes.

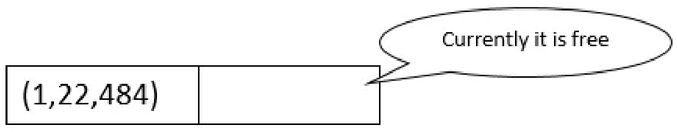
Now, it’s time to walk through an example to understand the working of the algorithm more clearly. As I said in the beginning of the post, we’ll be taking very simple example and the same can be extended to more complex cases, if required.

So, consider a set of 1-D objects {22,9,12,15,18,27,11,36,10,3,14,32}. Let the Branching factor B=2 and the Diameter threshold T=5

How are these parameters selected, I mean, based on what? — B depends up on pagesize in the memory. It is selected such that one node of CF tree fits in one page, so as to minimize I/O. T is selected such that the entire CF tree fits in the memory, so as to avoid multiple iterations to build the tree.

So, let’s start. Read the objects in the given order (the algorithm can also be used with data stream where the data/objects are received in real time) and keep inserting them into the tree in terms of CFs.

The first object is 22. It is directly inserted into a new node.



CF = (n, LS, SS) = (1, 22, 22^2) = (1, 22, 484)

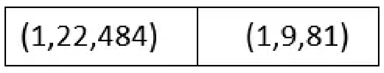
∴ Centroid = 22 & Diameter = undefined (as of now since only one object)

The next object is 9. First, try to insert it into the existing cluster.

So, CF=(n,LS,SS)=(2,22+9,22^2+9^2)=(2,31,565). Now calculate the diameter of the sub-cluster to see if it satisfies the threshold T (which is 5).

$$D = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\left(x_{i-x_j}\right)^2}{n(n-1)}} = \sqrt{\frac{2n(SS)-2(LS)^2}{n(n-1)}} = \sqrt{\frac{2*2(565)-2(31)^2}{2*1}} = 13$$

As we can see that the diameter is 13 which is greater than the given threshold (T=5), so the current object {9} cannot be inserted into the existing cluster and a new cluster is to be created.



∴ Centroid = 22 & 9

The next object is 12. First try to insert into the existing cluster with the closest centroid which is (1,9,81), centroid 9

CF=(2,9+12,9^2+12^2)=(2,21,225). Now calculate the diameter of the sub-cluster to see if it satisfies the threshold current object {12} is now actually inserted into the existing cluster and the CF is updated.

(1,22,484)	(2,21,225)

∴ Centroid = 22 (fist sub-cluster) and $LS/n = 21/2 = 10.5$ (second sub-cluster)

The next object is 15. First try to insert into the existing cluster with the closest centroid which is (2,21,225), centroid 10.5

CF=(3,21+15,225+15^2)=(3,36,450). The diameter of this sub-cluster will be, using the same equation, D=4.24 which is satisfying the threshold. So, {15} is inserted into this existing cluster and the CF is updated.

(1,22,484)	(3,36,450)

∴ Centroid = 22 (fist sub-cluster) and $LS/n = 36/3 = 12$ (second sub-cluster)

The next object is 18. Try to insert into the cluster with closest centroid 22

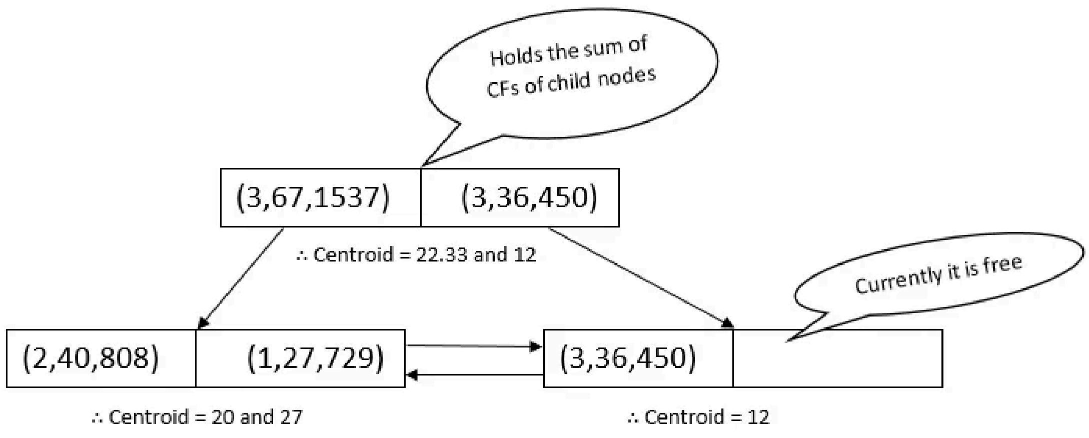
CF=(2,22+18,22^2+18^2)=(2,40,808). The diameter of this sub-cluster will be D=4 which satisfies the threshold.

(2,40,808)	(3,36,450)

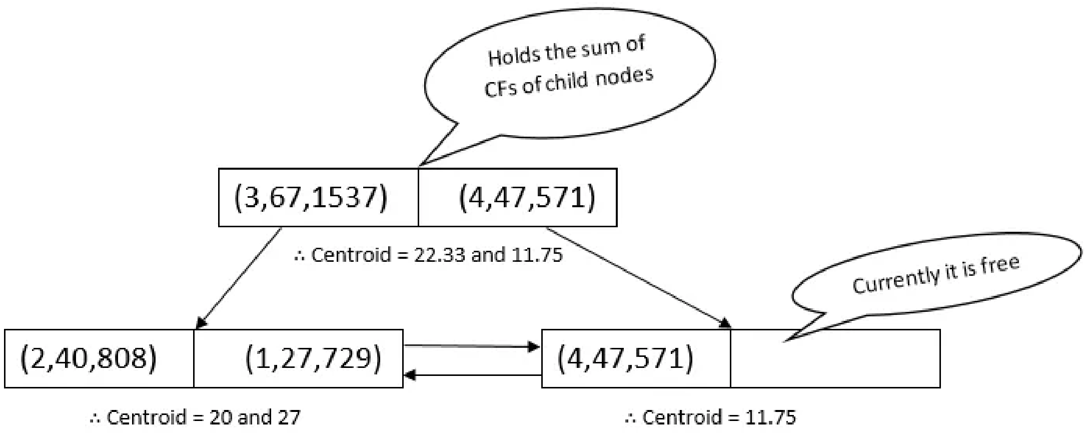
∴ Centroid = $LS/n = 40/2 = 20$ (fist sub-cluster) and $LS/n = 36/3 = 12$ (second sub-cluster)

The next object is 27 and the closest centroid is 20.

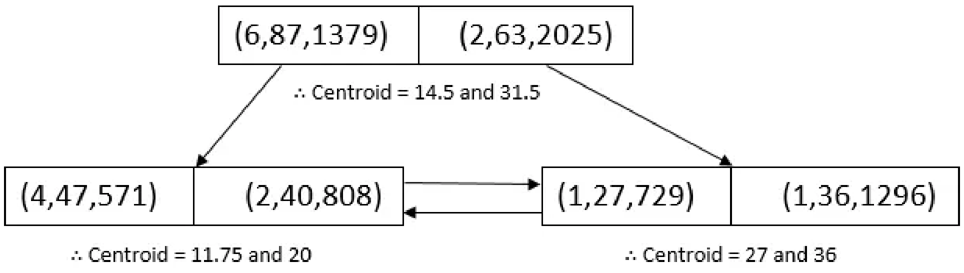
CF=(3,40+27,808+27^2)=(3,67,1537) & we get D=6.37 which doesn't satisfy the threshold. Therefore a new cluster to be created, but there is no space in the leaf node as it can hold only two CFs (recall that the Branching factor B=2). So, the leaf node is split and a new level is added to the CF tree.



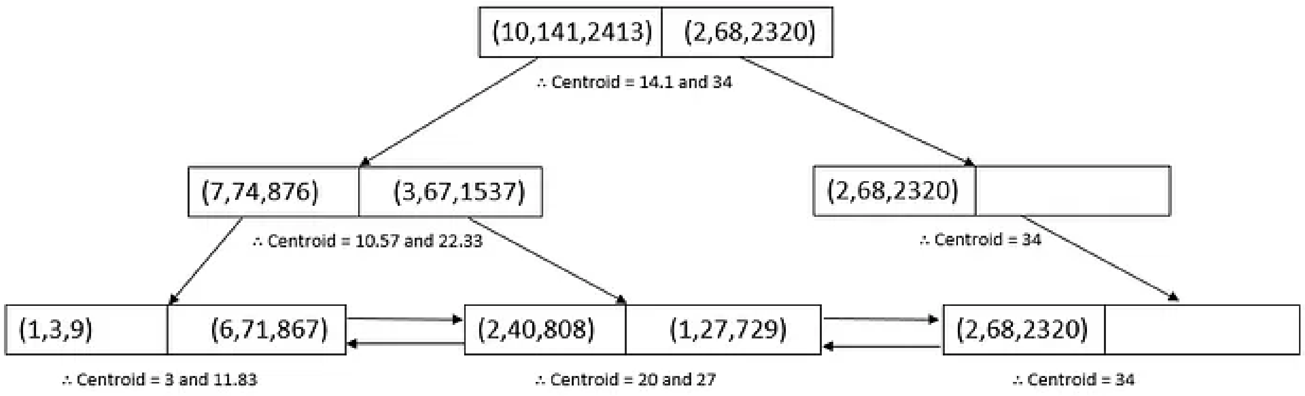
The next object is 11. Now, to determine the closest centroid in the leaf nodes, first compare it with centroids in the root. It is closest to 12, so follow the branch from this node which leads to leaf node with sub-cluster (3,36,450). Follow the same process and you get this CF tree after inserting {11}



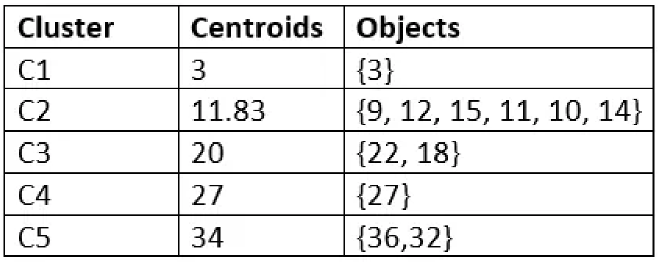
The next object is 36. The closest centroid in the root is 22.33, follow the branch and the closest centroid in leaf is 27. Up on calculating diameter, we find D=9. It is not satisfying the threshold, so there is a need to create new cluster, but there is no space in the leaf node, so there is a need to split it. But we can see that there is a space in another node in the leaf level. Therefore, to keep the tree height balanced, the CFs at the leaf level are redistributed, the current object is inserted in a new cluster and the corresponding CFs at root level are updated as follows,



By now, I have explained all the points required to build the CF tree. So, I leave it as exercise to the reader to complete this tree. The remaining objects are 10,3,14 and 32. The final CF tree should be as shown below.



The leaf nodes give the natural clustering structure of the given objects. This is end of Phase 1 which performed micro-clustering. So, if 5 clusters are desired then the leaf nodes can be used directly. The clusters are as follows,



Cluster	Centroids	Objects
C1	3	{3}
C2	11.83	{9, 12, 15, 11, 10, 14}
C3	20	{22, 18}
C4	27	{27}
C5	34	{36,32}

If more or less clusters are desired then agglomerative or partitioning method can be applied on leaf nodes in Phase 2 (macro-clustering)

I hope this helped you understand working of BIRCH algorithm. Please leave your queries/suggestions, if any. Thank you!!!

Clustering Birch Hierarchical Clustering

Follow

Written by Vipul Dalal

34 Followers

Machine Learning, NLP enthusiastic