BIRCHR PACKAGE FOR R

Akshay Kurapaty, Ankit Anand, Hemanth Devavarapu, Rohit Kata
Purdue University, Department of Management, 403 W. State Street, West Lafayette, IN 47907
akurapat@purdue.edu; anand57@purdue.edu; hdevavar@purdue.edu; kata@purdue.edu

ABSTRACT

Machine learning models are classified into supervised or unsupervised learning. Clustering is a unsupervised machine learning technique that involves grouping similar data points together. Performing cluster analysis is computationally expensive operation. As the clustering deals with unsupervised data, achieving the right cluster groups need multiple data scans to yield optimal solution. In the world where working with gigabytes of data is a norm, there is a dire need to develop algorithms that would yield accurate results with few data scans, that in turn reduces the computational effort. BIRCHR package developed for R is one such algorithm, which uses hierarchical tree-based algorithms to get the clusters. The key challenge in the implementation of this package was to design a work around mechanism to implement trees, as pointers are not supported in R. We used two dimensional lists in collaboration with a recursive algorithm to design self-organizing B-Tree, that always maintains the clusters at the leaf nodes and keeps all the leaf nodes at the same level.

INTRODUCTION

Birch, stands for balanced iterative reducing and clustering using hierarchies, was developed by Tian Zhang, Raghu Ramakrishnan and Miron Linvy[1]. Birch clustering algorithm is scalable and can work on large amounts of data with minimal time and space complexity. The hierarchical structure of the algorithm allows to create multiple processing threads to ease up the computational resources and gives the flexibility to scale up the resources in the era of cloud computing infrastructure such as Azure and AWS.

To decide on the clusters, birch clustering algorithm need not have to know all the data points before hand, it computes clustering features sequentially, which provides the fiddle with the memory constraint majority of the computers face.

BIRCH ALGORITHM

Birch clustering algorithm consists of two phases[2]. Building the CF tree and Performing global clustering. Because of this BIRCH is referred to as Two-Step clustering, because of the two phases mentioned above

BIRCH is a two-step clustering process. The process is listed below

- First, creation the CF Tree
- Second, Global clustering (existing clustering algorithm) is applied to leaves of the CF Tree

Creating CF

BIRCH clustering effectiveness is because of the employment of a small set of summary statistics to represent a larger set of data points. For clustering purposes, these summary statistics constitute a CF, and represent a sufficient substitute for the actual data.

A CF is a set of three summary statistics that represent a set of data points in a single cluster. These statistics are *count, Linear sum, and Squared sum*. Listed below is an example for better understanding of how elements of a CF Tree is created.

Considering two clusters: Clusters 1 and 2.
Cluster 1 contains data values (1, 1), (2, 1), and (1, 2)
Cluster 2 contains data values (3, 2), (4, 1), and (4, 2).

CF1, the CF for Cluster 1, consists of the following:
CF_1 = {3, (1 + 2 + 1, 1 + 1 + 2), (12 + 22 + 12, 12 + 12 + 22)}
= {3, (4, 4), (6, 6)}
For Cluster 2, the CF is
CF_2 = {3, (3 + 4 + 4, 2 + 1 + 2), (32 + 42 + 42, 22 + 12 + 22)}
= {3, (11, 5), (41, 9)}

Also, we know that BIRCH calls for merging of cluster under conditions, described later, for effective clustering of large data sets. Here, we use Additive theorem to combine the CFs. Thus, merging of clusters 1 and 2 would result a CF having following statistics.
CF_{12} = {3 + 3, (4 + 11, 4 + 5), (6 + 41, 6 + 9)} = {6, (15, 9), (47, 15)}

Creating CF Tree

Since now we have understanding of CF, we move ahead to create the CF Tree. Creating a CF tree depends on 3 parameters, as described below
• **Branching Factor *B***. Determines the maximum children allowed for a non-leaf node
• **Threshold *T***. *it* is an upper limit to the radius of a cluster in a leaf node
• **Number of Entries in a Leaf Node *L***

Creating of CF Tree is done using a sequential clustering approach, whereby the algorithm scans the data one record at a time and determines whether a given record should be assigned to an existing cluster, or a new cluster should be constructed.

The CF tree building process consists of four steps, as follows [2]:

1. For each given record, BIRCH compares the location of that record with the location of each CF in the root node, using either the linear sum or the mean of the CF. BIRCH passes the incoming record to the root node CF closest to the incoming record.
2. The record then descends down to the non-leaf child nodes of the root node CF selected in step 1. BIRCH compares the location of the record with the location of each non-leaf CF. BIRCH passes the incoming record to the non-leaf node CF closest to the incoming record.
3. The record then descends down to the leaf child nodes of the non-leaf node CF selected in step 2. BIRCH compares the location of the record with the location of each leaf. BIRCH tentatively passes the incoming record to the leaf closest to the incoming record.
4. Perform one of (a) or (b):

a. If the radius (defined later in the paper) of the chosen leaf including the new record does not exceed the Threshold *T*, then the incoming record is assigned to that leaf. The leaf and all of its parent CFs are updated to account for the new data point.

b. If the radius of the chosen leaf including the new record does exceed the Threshold *T*, then a new leaf is formed, consisting of the incoming record only. The parent CFs are updated to account for the new data point.

Global Clustering

Once we are done creating the CF tree, any existing standard clustering algorithm (we used k-means) may be applied to the sub-clusters (the CF leaf nodes), to combine these sub-clusters into clusters. Because there are many fewer sub-clusters than data records, the task becomes much easier (reduced computational expense) for the clustering algorithm in the cluster step.

IMPLEMENTATION

Implementation of this algorithm in R is a key challenge, as R do not support pointers, which are extensively used to implement tree data structures. Creating a work around to make this implementation using Two dimensional arrays allowed us to create an implementation.

We utilize a 2 dimensional list. Initially the list is empty. Each element in the list will function as the node and has the capability to store the address of parent node, child node and CF values of the data. The CF values that are stored are:

• Count: How many data values in the cluster

• Linear Sum: Sum the individual coordinates. This is a measure of the location of the cluster

• Squared Sum: Sum the squared coordinates. This is a measure of the spread of the Cluster

Mainlist which is a two dimensional list is created to hold all the data related Implementation of this algorithm is dealt using the following functions:

- MakeCFTree
- Compute_radius
- Calculatenearestnode
- splitNode
- rearrange
- createnewnodetop

- recalculateCF

MakeCFTree:

MakeaCFTree function takes data points x (point in n dimensional space) , branchingfactor (maximum number of child nodes allowed) and threshold (Maximum width of the cluster) as input. The first point becomes the root node and as further points are provided as input, the tree grows by utilizing the calculatenearestnode() function & Compute_radius() function. If the new point is within the threshold of an existing cluster, then it will be added to that cluster or else a new cluster is created. For new cluster creation, the branching factor condition is checked and if the number of child nodes are less than branching factor then the new cluster is created to the same parent node. If the number of child nodes are more, then we use the combination of splitNode() function and rearrange() function to split the nodes at required levels to accommodate the new cluster. Createnewnodetop() function is used in the scenario where we need to create a new level for splitting of the CF's. Once the new cluster is formed, recalculateCF() function will recalculate the CF's at all levels where updating is required.

Compute_radius:

For the nearest node calculation, it takes the calculated LS, SS and N of the CF nearest node along with the new point. With the updated LS, SS and N, the radius is calculated and is returned by the function which is used to check if the radius is within the threshold or not.

splitNode:

This function is called if the count of the number of clusters retuned by the leaf is equal to the branching factor. It takes the depth and index of the nearest node calculated and splits the node.

Rearrange:

The re-arrange function would take the depth and index of the node at which split has to take place and its child nodes are arranged such they the closer cluster stay in the same branch after splitting. This will ensure that the new points are going the node.

Createnewnodetop:

This is created in scenario when a new node has to be created when the branching factor is reached in multiple levels

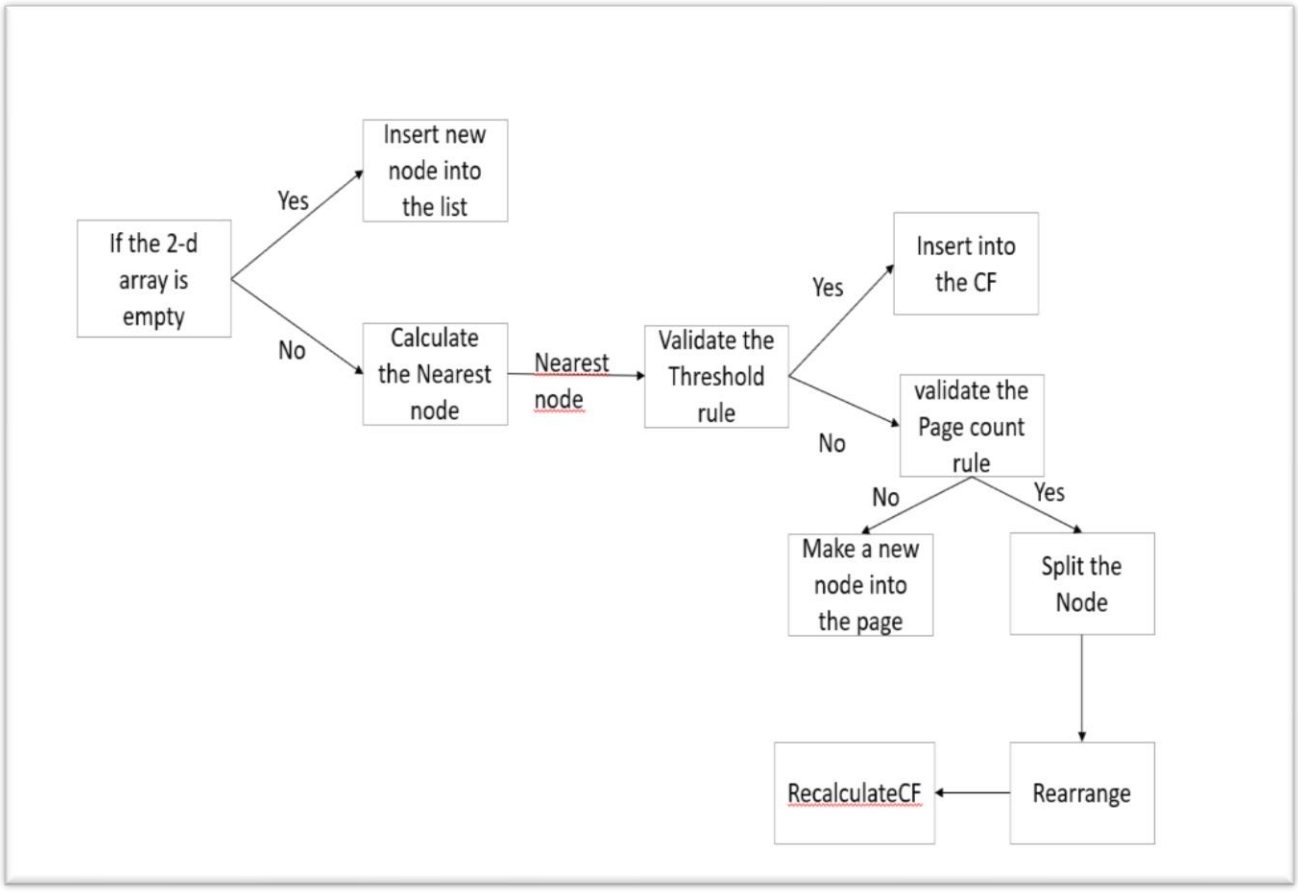
recalculateCF:

With the inclusion of a new point, the CF of the all the associated parent nodes CF's is updated using this function.

Figure 1 explains the flow of the algorithm in detail.

FIGURE -1

Flowchart of Algorithm Implementation in R



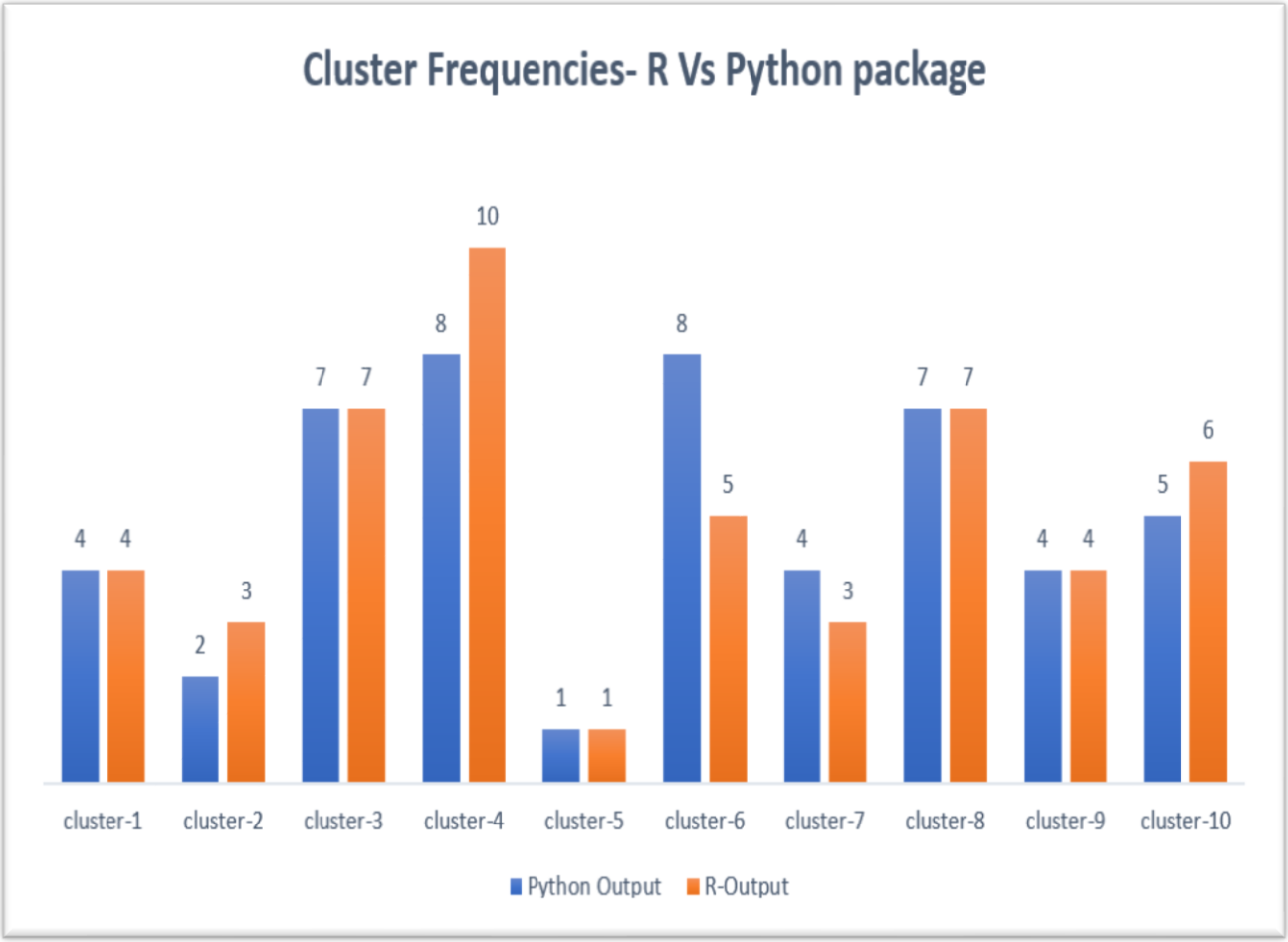
RESULTS

The clusters formed from the BirchR package are compared with the clusters formed from Birch algorithm from scikit learn library in python. The results proved to be completely similar with the results yielded from python. The cluster analysis has been performed on the cars dataset which is by default available in R and imported to Python to validate the accuracy of clusters.

Figure 2 is the chart which shows the frequency of datapoints in each cluster.

FIGURE -2

Cluster Frequency Comparission Between R and Python



REFERENCES

[1] Tian Zhang, Raghu Ramakrishnan, Miron Livny Birch. An Effcient Data Clustering Method for very large Databases, available at

https://www.cs.sfu.ca/CourseCentral/459/han/papers/zhang96.pdf

[2] Daniel T. Larose, Chantal D. Larose. Data Mining and Predicitve Analaytics, WILEY