Scalable Hierarchical Agglomerative Clustering

Anonymous Author(s)

ABSTRACT

We have some algorithms which are highly efficient but not scalable in nature when the data sets are more than a few million while some of the other algorithms are highly scalable, but they suffer from over-merging of the data. Hence, we represent the best of both the worlds: An agglomerative algorithm yet scalable method for hierarchical clustering. The main benefits are that the algorithm is efficient as well as scalable. Clustering is a process in which we group our given dataset into clusters which have similar features or characteristics. We will briefly go over HAC algorithm preview before we present a novel scalable and agglomerative clustering algorithm that is not affected by the size of the dataset and does not compromise the quality of clustering or the accuracy. Existing scalable hierarchical clustering methods sacrifice quality for speed and often lead to over-merging of clusters [2]. We also compare how the algorithm presented in the paper compares to classic hierarchical agglomerative clustering method.

1 INTRODUCTION

Clustering is a technique which is widely used in today's world to produce similarities in our dataset, analyze them and visualize them. There are two types of clustering techniques, Hierarchical and Flat Clustering. Hierarchical Clustering fundamentally creates a tree like structure where each level of the tree is a cluster, thus creating a hierarchy of clusters. Flat Clustering has no structure or hierarchy, where we choose the number of categories or cluster into which the data should be divided. We have seen many clustering algorithms in the years before like K-Means, DP-Means, HAC (Hierarchical Agglomerative Clustering), etc. For Example, given a dataset for the COVID cases in a country, which has information like the location, the severity of each case, we can use clustering algorithms to find out which areas of the country has more COVID cases depending on the maximum number of data points in each cluster for a given number of clusters. Thus, clustering is widely used and beneficial to various industries in practice.

Non-hierarchical clustering or flat clustering, in which K-means, a centroid-based algorithm, is the most used algorithm as it is simple and efficient. We are given a set of clusters K and centroid values initially. The dataset is then divided into K clusters using the minimum distance calculation technique like Euclidean distance, Manhattan distance, Mahalanbois distance etc. between a data point and the centroid. Until the centroid values do not vary we keep iterating. It has a use case in horticulture, astronomy, image processing, market division, weather forecasting, bioinformatics and computer vision. [1]

Hierarchical clustering, in which the leaves correspond to data points and the internal nodes correspond to clusters of their descendant leaves, can be useful to represent clusters of multiple granularities or to automatically discover nested structures. Hierarchical clusterings represent multiple alternative tree consistent partitions. [2] Each node of tree in hierarchical clustering can be related to the flat clustering of data i.e., we can produce a flat clustering if we partition our tree at regular intervals and get the relevant nodes from that partition. Therefore, it is beneficial to extract a flat clustering from a hierarchical clustering instead of carrying out the flat clustering on our dataset directly. The most used clustering algorithm is the bottom-up Hierarchical Agglomerative Clustering (HAC), but the downside of this algorithm is its time complexity and scalability. The approximate runtime of HAC is

$$O(N^2 log(N))$$

for N points in our dataset. Initially, every data point is its own cluster. This is because at each step we merge the data based on the shortest distances and then merge them. Hence, each iteration takes O (N log N), and this process is carried out until we are left with the single cluster, hence, the total complexity of the algorithm is

$$O(N^2 log(N))$$

The main problem of the algorithm is still scalability. The existing bottom-up agglomerative algorithm is inherently sequential in nature. Competing methods attempt to achieve better scalability by operating in an online manner [2].There are some algorithms which achieve scalability (randomized/parallel/ distributive methods) but they do at the cost of accuracy. One of the examples is the Affinity algorithm. Our paper is going to present the algorithm which interpolates between the two well-known (said) algorithms. This new algorithm is called Sub-Cluster Component Algorithm (SCC).

2 SUB-CLUSTER COMPONENT ALGORITHM

The Sub-Cluster algorithm works similarly to HAC, it has a certain number of rounds and in each round, it figures out the points which should be together in a single cluster. The type of tree that HAC produces is a Compact, binary tree which is not the case with SCC. It produces a compact, non-binary tree. However, HAC is not suited for large data-sets due to its sequential nature and exceptionally large number of internal nodes. SCC allows for independent decisions at each level. The hierarchical structure that both the algorithm produces can be seen as a tree structure where leaves are the data points, and the internal nodes are the clusters formed after each round.

The algorithm starts as all the data points are its own clusters and there is an input received which is monotonically increasing sequence of thresholds. Every node connects to its nearest neighbor and then we prune those edges based on the threshold value. If the distance between any two nodes is greater than the threshold, we remove such edges. We pop the lowest thresholds and merge the clusters from the previous round such that distance between the two clusters is less than or equal to the popped threshold value. For the remaining graph we merge the connected components and then advance on to the next round. We continue as such until we find that there is no change in consecutive rounds and then we advance our threshold to the next large value. We continue in the above manner when we have no more clusters remaining to merge. Therefore, it is crucial to have this threshold value which avoids over-merging of clusters. The intuition behind how this works is that the increasing thresholds will allow first all points in the inner cluster to merge while keeping points from one cluster separate from points of another cluster. Like HAC, SCC inherits the best-first

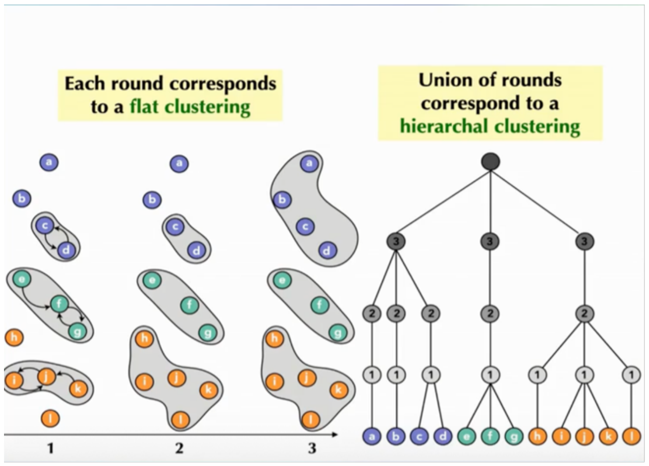
Each round corresponds to a flat clustering
Union of rounds correspond to a hierarchal clustering
a
b
c
d
e
f
g
h
i
j
k
l
1
2
3

Figure 1: [2] The Sub-Clustering Component Algorithm. We illustrate SCC on a small dataset. The formation of sub-clusters is shown with black arrows for pairs of points satisfying Def. 3. The direction indicates the nearest neighbor relation- ship (Def. 3, condition 2). Red edges indicate the nearest neighbor relationships that are above the distance thresholds. The grey circles indicate the sub-cluster components created in that round. Best viewed in color.

manner, it puts together points in a cluster in rounds. The sequence of rounds begins with the decisions that are "easy to make" (e.g., points that are clearly in the same cluster) and prolongs the later, more difficult decisions until these confident decisions have been well established. [2] But, HAC has certain drawback which we will see using the illustration in the Figure 2.

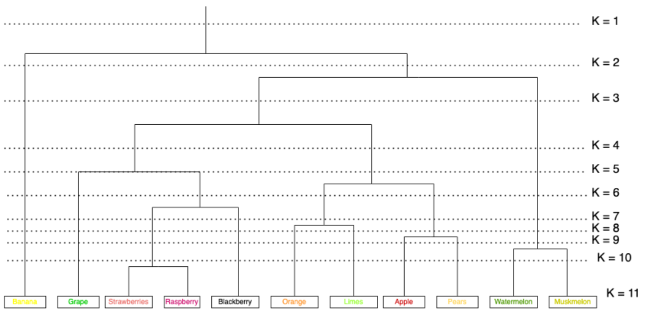

Figure 2: HAC representation of an arbitrary fruits' dataset.

Let's take a dataset of fruits and cluster them using their types/family like Berries, Tropical, Citrus, Melons, etc. The fruits belonging to the same family should belong in the same cluster. HAC creates a binary tree making decisions as it goes from bottom to top, while this algorithm is not particularly suited for large datasets as due to its sequential nature and large internal node count. Binary trees only accommodate up to two children and this increases the space complexity of the algorithm. Therefore, HAC has limited scalability to larger and more complex datasets.

Whereas on the other hand, SCC uses non-binary tree structure and is much more scalable compared to HAC. We will illustrate this in the figure 3 using the same dataset only with some extra data but the same clustering criteria and find that the tree has a lot less internal nodes and is much compact.

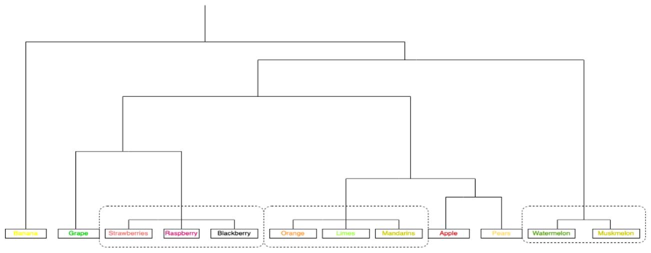

Figure 3: SCC representation of a dataset similar to the fruit dataset used to visualize HAC. The Dotted Rectangles indicated that the datapoints are in a subcluster.

3 HIERARCHICAL CLUSTERING ANALYSIS

Hierarchical and Sub-Cluster Component Algorithm (SCC) needs to be compared to see the differences in their performances, one of them is Dendrogram Purity. Dendrogram is a branching diagram which is used to represent the similarity between group of entities. Purity in dendrogram is considered as a measure of range to which clusters contain a single class. In other words, Dendrogram purity can be given as the mean, over all pair of points from the same ground cluster, of the purity of the least common ancestor of the pair. Most clustering algorithms, try to maximize dendrogram purity as it the main goal of clustering is to construct a tree that puts similar points closer together in the tree. Among some of the taken datasets the SCC algorithm performs best by attaining maximum dendrogram purity for all but one considered dataset. For this algorithm, we can consider either linear or geometric progression for threshold. After noticing the algorithm output or performance an observation is made where SCC performs a bit finer in terms of geometric progression compared to linear. In addition to that SCC can accomplish top standard results in just few rounds.

From the Figure 4 graphs and experiments, the observations are much clearer. HAC has quadratic time complexity and grows at a faster rate as the size of dataset increases, whereas on the other hand, SCC follows a close to linear characteristics. But still, the dendrogram purity of the SCC trees are pretty much similar to those produces by HAC. As the only difference between the tree structure are HAC produces binary trees and SCC produces non-binary trees.

4 APPLICATIONS

We examine the use of SCC for clustering our fruits data for example. Here's how to cluster's look after we examine the result of SCC on

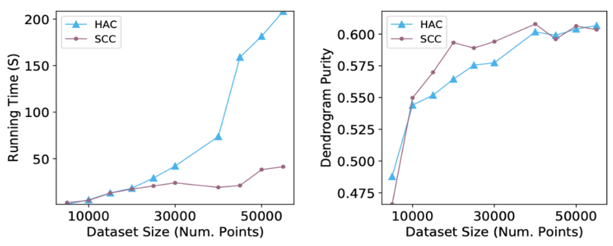

Figure 4: [2] Comparison to HAC: We report the running times and Dendrogram purity of SCC compared to HAC, on a synthetic dataset.

fruits data in Figure 3. But the SCC algorithm prefers to work on large datasets, so let's take the IMDB dataset which has about 8 million entries in their movies dataset which has movies of different genres like Comedy, Action, Sci-fi, etc. Using SCC, we extract and visual the dataset by dividing each movie into sub-cluster of genres. SCC will generate cluster which are accurate, clear and coherent as shown in Figure 5.

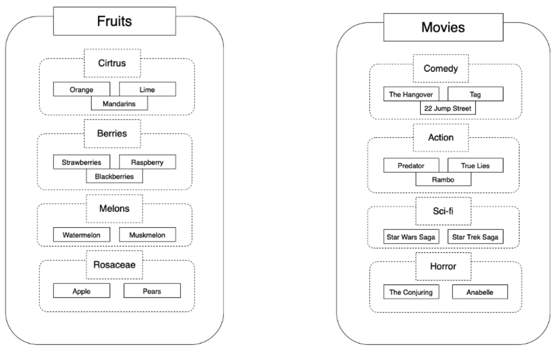

Figure 5: Hierarchy inferred on fruit dataset using SCC.

[2] We represent the hierarchy using rectangular boxes. The root with header is represented by the outer rectangular box (solid line). The second level of the hierarchy, with header, is shown in dashed (– –) rectangle withing the outer box. Finally, the third level from the root is shown as the inner most dotted (...) rectangle. Plain text within each of the dotted rectangle are the families that belong to that cluster. For example, Citrus is a sub-cluster and orange, lime and mandarins are the fruits which belong in that cluster because they are citrus. Similarly, we cluster berries, Melons and Rosaceae. We can add as a note that there will be many sub-clusters depending on the dataset and the selection criteria.

5 CONCLUSION

In this paper, a new algorithm Sub-Cluster Component algorithm (SCC) for scalable clustering was introduced which uses best-first, round based, agglomerative method. In each round, the distance threshold is increasing which dictates which points can be merged to make a subcluster. Due to merging, the tree structure of SCC is compact and consume less space because the trees are non-binary leading to less levels compared to the tree for HAC. However, there exists some threshold value for which no internal merging occurs and the trees generated by SCC is similar to that of HAC. We later compared, the performance of HAC and SCC in terms of their running times on large dataset size and dendrogram purity. The performance of SCC was found to be increasing in at a slower rate to HAC, whereas the dendrogram purity is similar.

6 ACKNOWLEDGEMENTS

We would like to thank Prof. Carlos R. Rivero for introducing us to the KMeans algorithm and Clustering in general which led us to dive deep into different clustering techniques like Hierarchical Clustering. To extend our gratitude to the authors of this paper, Nicholas Monath, Kumar Avinava Dubey, Guru Guruganesh , Manzil Zaheer , Amr Ahmed , Andrew McCallum , Gokhan Mergen , Marc Najork, Mert Terzihan , Bryon Tjanaka , Yuan Wang , Yuchen Wu, for introducing us to this novel clustering algorithm. We are grateful to S. Patel, S. Sihmar and A. Jatain for providing us with a deep study on hierarchical clustering algorithms.

REFERENCES

[1] K. B. A comparative study on k-means clustering and agglomerative hierarchical clustering. *International Journal of Emerging Trends in Engineering Research*, 8: 1600–1604, 05 2020. doi: 10.30534/ijeter/2020/20852020.

[2] N. Monath, K. A. Dubey, G. Guruganesh, M. Zaheer, A. Ahmed, A. McCallum, G. Mergen, M. Najork, M. Terzihan, B. Tjanaka, Y. Wang, and Y. Wu. Scalable hierarchical agglomerative clustering. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery amp; Data Mining*, KDD '21, page 1245–1255, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383325. doi: 10.1145/3447548.3467404. URL https://doi.org/10.1145/3447548.3467404.