Literature Review:Performance Comparison of Centroid Based Clustering Algorithms

Aagyapal Kaur
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
aagyapalkaur@cmail.carleton.ca

October 6, 2021

1 Introduction

Enterprises today are dealing with the massive size of data, which have been explosively increasing. The key requirements to address this challenge are to extract, analyze, and process data in a timely manner. Clustering is an essential data mining tool that plays an important role for analysing the big data. However, large scale data clustering has become a challenging task because of the large amount of information that emerges from technological progress in many areas, including finance and business informatics. Map Reduce is one of the most famous frameworks, and it has attracted great attention because of its flexibility, ease of programming, and fault tolerance. Map Reduce is a programming paradigm that is used for computation of large datasets.

Cluster Analysis

Clustering basically deals with grouping of objects such that each group consists of similar or related objects. The main idea behind clustering is to maximize the intra-cluster similarities and minimize the inter cluster similarities. The data set may have objects with more than attributes. The classification is done by selecting the appropriate attribute and relate to a carefully selected reference and this is solely dependent on the field that concerns the user. Classification therefore plays a more definitive role in establishing a relation among the various items in semi or unstructured data set. Cluster analysis is a broad subject and hence there are abundant clustering algorithms available to group data sets. Very common methods of clustering involve computing distance, density and interval or a particular statistical distribution. Depending on the requirements and data sets we apply the appropriate clustering algorithm to extract data from them.

Clustering has a broad spectrum and the methods of clustering on the basis of their implementation can be grouped into

- Connectivity Technique Example: Hierarchical Clustering
- Centroid Technique Example: K-Means Clustering
- Distribution Technique Example: Expectation Maximization
- Density Technique Example: DBSCAN
- Subspace Technique Example: Co-Clustering

In this project, I mainly focused on performance analysis of Centroid Technique based clustering algorithms that include K-means, K-means++ and mrk-means and would like to find out which one of these is effectively performed on the datasets.

2 Literature Review

2.1 The Basic Principle of K-means Algorithm

The K-means algorithm is one of the most well-known clustering algorithms. It has been successfully utilized to solve variety of problems during the last few years because it is simple and straightforward.

Algorithm 1. The k-means algorithm.

1: **procedure** K-MEANS (M, k)
2: Randomly select k initial centers $\hat{C} = \hat{c}_1, \hat{c}_2, ..., \hat{c}_k$
3: **repeat**
4: Assign each data item to the nearest center, and update each $\hat{c}_i$
5: **until** there are no changes in $\hat{C}$.
6: **return** $\hat{C}$
7: **end procedure**

Figure 1: [2]

The traditional K-means algorithm partitions the samples into separated clusters by minimizing a measure between the samples and the centroid of the clusters.Steps of K-means can be defined as:

i) Select k points as initial centroids arbitrarily.

ii) Calculate the distance (Euclidean Distance) between the rest data and k centroids. Arrange each data into the nearest cluster. According to Euclidean Distance formulae, the distance between the two points in the plane with coordinates (X1, Y1) and (X2, Y2) is given by:

d= $\sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}$

iii) After processing all the points, compute the average of all the records in each same cluster.

iv)The average of each cluster is used as the new centroid.

v) Calculate the difference between the new centroid with the original one in the same cluster. If the difference is smaller than the threshold or the number of iterations of the algorithm has been reached for the maximum, the algorithm is over. Otherwise, the new centroid is substituted for the original ones. Return ii) and continue until the convergence is achieved and obtain the final cluster [1].

Limitation of K-means:

In theory, the main loop of k-means is repeated until the set of centers converges and there is no changes. This procedure is also known as Lloyd's algorithm. However, in practice this condition may never be satisfied or its satisfaction may need many iterations. Hence, the main loop of k-means is usually bounded to a fixed number.[2] Moreover , the problem of initialization sensitivity occurs owing to pick the initial centers randomly from the data points. This problem tends to affect the final formed clusters.The final formed clusters depend on how intial centroids were picked.

2.2 To overcome the limitation of Kmeans, we use K-means++ Algorithm:

The K-means++ algorithm is identical to k-means except the initialization of centers. K-means++ tries to choose a set of carefully selected initial centers instead of random initialization[2] . This algorithm ensures a smarter initialization of the centroids and ameliorate the quality of clustering. If we assume D'(x) to be the distance of a data item x to its nearest center that is already chosen, the initialization step of k-means++ can be defined as:

i)Pick the first centroid point C1 randomly.

ii) Compute the distance of all the points in the datasets from the selected centroid. The distance of xi data point from the farthest centroid can be computed by:

di= max(j:1→m) $[x_i - C_j]^2$

where di: distance of xi point from the farthest centroid m: number of centroid already picked

iii) Make the point xi as the new centroid that is having a maximum probability proportional to di

iv) Repeat the step ii) and iii) till you find k centers.

v) Proceed as with the standard k-means algorithm.

Algorithm 2. k-means++ centers initialization.

1: **procedure** K-MEANS++(M, k)
2: Choose a center $\hat{c}_1$ randomly from dataset.
3: $\hat{C} \leftarrow \{\hat{c}_1\}$.
4: **while** $|\hat{C}| \neq k$ **do**
5: Choosing an item $\vec{m}_i \in M - \hat{C}$ with probability of $\frac{D'(\vec{m}_i)^2}{\sum_{\vec{m}_j \in M} D(\vec{m}_j)^2}$
6: $\hat{C} \leftarrow \hat{C} \cup \{\vec{m}_i\}$.
7: **end while**
8: **return** $\hat{C}$
9: **end procedure**

Figure 2: [2]

2.3 Map Reduce design

Map Reduce is a programming model and an associated implementation for processing and generating large data sets, which was introduced by Google to solve certain kinds of distributable problems. Programs written in MapReduce's functional style can be automatically parallelized and executed on a high performance computing cluster of a large number of low-cost ordinary personal computers. In Map Reduce, the parts of a dataset are processed separately by map functions (known as mappers also). The reduce functions (known as reducers also) are provided with the results from the mappers. In this way, Map Reduce process a large dataset by partitioning the jobs between mappers and reducers . The dataset input to the Map Reduce must first transform into key and value pairs as the mappers and reducers can only work in this format.

Mapper: (k_1, v_1)ß$[(k_2, v_2)]Reducer : (k_2, |v_2|)$ß$[(k_3, v_3)$

where, k1 k2 are in-key and out-key respectively, and v1 and v2 are in-value and out-value respectively. k3 and v3 are final keys and final value respectively. —v2— is the out value list. The mappers execute in parallel on different data splits of an input dataset and output intermediate pairs of keys and values. The reducers obtain these values from mappers and calculate the final value for each key[3]. Fig. 3 provides the working style of Map Reduce with mappers and reducers.

The traditional K-means clustering is simple, but the time complexity is too high and the efficiency is too low when it is used to process the massive data. When handling the massive data, the memory of a single node also can be a restriction. As a consequence it has not been used previously with large data sets. So, in order to enhance the performance of K-means , this traditional algorithm is combined with Map Reduce framework to implement the mrk-means algorithm.

2.4 The divide and conquer approach – mrk-means in Map Reduce

To define our divide and conquer approach, it is necessary to define the algorithm that is applied on every chunk and the final algorithm that is applied on the set of intermediate centers. For both phases we use k-means++. For each chunk, k-means++ is applied and k centers are extracted. Therefore, if we have c chunks, after applying k-means++ on each chunk, there will be a set of k * c items as the intermediate set. To reach a better accuracy, we keep the number of assigned items to each extracted center, too. That is to say, we extract k centers and the number of items assigned to each center, from each chunk. In addition to higher accuracy, keeping centers weights helps us to prove the optimality of mrk- means.

Fitting the proposed divide and conquer approach into the Map Reduce framework is straight

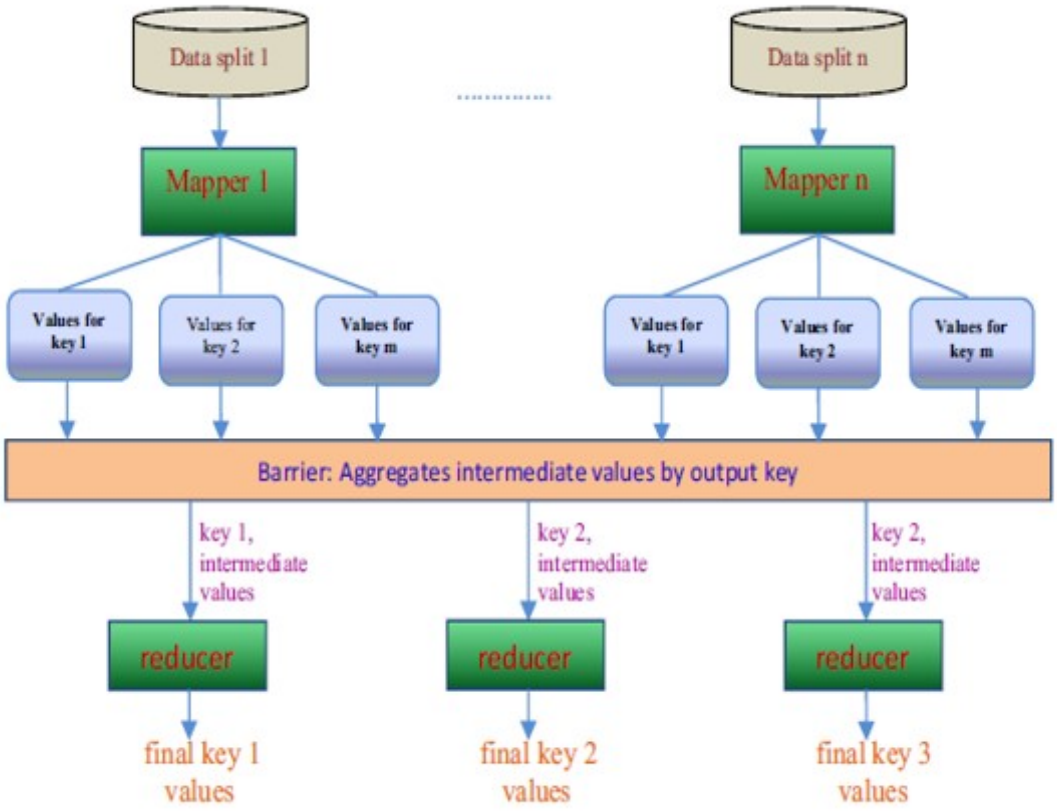
Data split 1
Data split n
Mapper 1
Mapper n
Values for key 1
Values for key 2
Values for key m
Values for key 1
Values for key 2
Values for key m
Barrier: Aggregates intermediate values by output key
key 1, intermediate values
key 2, intermediate values
key 2, intermediate values
reducer
reducer
reducer
final key 1 values
final key 2 values
final key 3 values

Figure 3: [3]

Algorithm 3. Divide and conquer mrk-means in MapReduce.

procedure MRK-MEANS M, k

THE MAP PHASE:
Divide the dataset M into a set of αchunks $B = B_1, B_2, \ldots, B_\alpha$
for each B_i **in** B **do**
  Spawn a map task and run the k-means algorithm using k-means++ initialization on B_i to get a set of k weighted centers T_i
  Emit T_i as the output of the map task.
end for

THE REDUCE PHASE:
let C_w be the set of intermediate centers outputted from the map phase.
Run the weighted k-means algorithm using k-means++ initialization on $\hat{C}_w$ to get a set of k final centers $\hat{C}_f$
return $\hat{C}_f$

end procedure

Figure 4: [2]

forward. Since processing of each chunk is independent from others, each chunk can be processed in a dedicated map task. That is to say, each map task picks a chunk of data set and performs the k-means++ algorithm on that chunk, and then emits the resulted k centers and their weights as the output. After the map phase is completed, a set of intermediate weighted centers is formed as the output of the map phase, and this set of centers is given to a single reduce task. The reduce task gets the set of intermediate centers and their weights and performs a final k-means++ on them and returns the final centers as the output[2]. The map and reduce functions are given in Algorithm3.

2.5 References

[1]Q. Liao, F. Yang and J. Zhao, "An Improved parallel K-means Clustering Algorithm," in 2Beijing University of Posts and Telecommunications, Beijing 100876, China, Beijing, 2013.

[2]S.Shahrivari and S. Jalili, "Single-pass and linear-time k-means clustering based," Elsevier, pp. 1-12, 2016.

[3]Tanvir Habib Sardar and Z. Ansari, "Partition based clustering of large datasets using MapReduce framework:An analysis of recent themes and directions," Elsevier, pp. 247-261, 2018.

[4]H.-G. Li, G.-Q. Wu, X.-G. Hu, J. Zhang, L. Li and X. Wu, "K-Means Clustering with Bagging and MapReduce," in Department of Computer Science, University of Vermont, Burlington, VT 50405, U.S.A., Burlington, 2011.