MR-KClust: An efficient Map Reduce based clustering Technique

Snigdha Agarwal[1] and Rituparna Sinha[2]

[1] Heritage Institute of Technology, Kolkata
[2] Heritage Institute of Technology, Kolkata
rituparna.sinha@heritageit.edu

Abstract- The world is rich in data, however efficient storage and processing of the data is the utmost need. As the data grows big, processing the data and applying algorithms to extract meaningful and hidden patterns efficiently, becomes a challenge to the research community. Keeping this in mind, an efficient map reducea clustering algorithm, named MR-KClust, has been designed. It is a variant of the well known k-means clustering, where a polynomial regression has been applied for the initial selection of centroids and algorithms. In the Map Reduce design of the algorithm, the data has been splitted and each input split is provided to the mappers, where the map function calculates the distances from each sample to each centroid point and emits relabeled cluster ID as a key, and data ID coordinates as value. The combiner function combines samples with the same key together for each machine and loads in what the mapper emits. The performance of the algorithm as compared with the existing variants of k means, outperformed the others with respect to the mean squared error and has a decent runtime.

Keywords: Hadoop Distributed File System, MapReduce, Big Data, K Means, Clustering

1. Introduction

The exponential increase in the generation of data over the past decades, together with advances in computational efficiency of machines, has led to the development of technologies specifically created to manage, store and analyze humongous amounts of data known as Big Data.[1] It has three main characteristics, namely, 1) velocity 2) volume 3) variety, which refers to the wide-ranging rate, amount and type of data respectively. As a result, it cannot be managed by traditional database management systems.[2] Hadoop, open-source software was developed to overcome the computational challenges of Big Data like heterogeneity, timelessness, scale, privacy etc and enable us to effectively reserve and operate petabytes of data. So, rather than storing and processing all the data on a single system, we may disperse it among many processors, each of which is capable of processing massive volumes of data concurrently, thereby reducing computation time. Hadoop Distributed File System (HDFS) [3] and MapReduce architecture [4] make up its two main processing and storage components.[5] A mapper and a reducer are parts of the MapReduce. It allows us to work on large datasets in parallel. The data is parsed by the mapper, who also turns it into usable information. After which, the output generated by the mapper is fed to the reducer. The reducer then merges it accordingly, reducing it into smaller subsets. To enhance the performance of clustering techniques, the MapReduce framework is leveraged.[6]

The K Means algorithm, also known as Lloyd's algorithm, [7] is a heuristic algorithm that solves the NP-hard problem of partitioning a dataset of unlabeled points into clusters.[8] It appears in a wide variety of applications, being one of the most simple, and so has been convenient for many analysts. But this standard algorithm's iterations take O(nkd) time, and for the algorithm to converge, it can take quite a number of interactions.[9] Furthermore, the main problem with the standard k-means algorithm is its accuracy. With its average computational time, the accuracy that it provides is very low, producing arbitrarily chosen bad clusterings. Many variants were suggested to improve the accuracy of the k-means algorithm.[10] One of them was, k-means++ was introduced. [11] In this algorithm, a set of initial means that are well separated from one another was chosen, which led to an improvement in final error compared to the standard k-means process. Although the accuracy improved, the major problem with this method was the computational time because of the initialization overhead. Each point initialization iteration takes O(|M|nd) time, with M increasing by one every iteration until it reaches k. The total complexity of k-means++ is O(k2nd), plus O(nkd) per iteration once the standard k-means method starts.[11] Thus increasing the computational time taken by the algorithm. This issue is sorted to some extent using Hadoop MapReduce by introducing k means** and k means+*, [12] which were the MapReduce versions of the k means and k means++ algorithm respectively. While the k means** improved the time taken by the standardized k means the problem in accuracy of the model still remains intact. In case of the k means+* the computational time taken improved but it was not significant and was still a major issue.

In this work, a new approach has been proposed for the initialization of centroids by introducing the polynomial regression curve that runs through the clusters. The polynomial regression curve with appropriate degrees can fit much better on a spread-out dataset, and initialization of centroids around this curve can help achieve convergence much more quickly and efficiently. This approach is then further combined with the MapReduce technique to further reduce the computational time taken, to come up with a new algorithm, MapReduce K Means. In addition, preliminary experimental data is also provided which shows that, in practice, MR-KClust outperforms k-means and its variants in terms of accuracy and computational speed, often by a significant margin.

2. Methodology

In the standard k-means algorithm, for n number of data points A = { a_1, a_2, …, a_n}, each having dimension m. The main goal is to find k number of means M = { μ_1, μ_2, …, μ_k} which minimizes the function

$$f(M) = \sum_{a \in A} min_{\mu \in M} ||a - \mu||^2$$

that is to reduce the total sum of squared distances between every point in the dataset and the mean closest to that point.[10] In MR-KClust algorithm, this task is divided into 3 parts, namely: the mapper, the combiner, and the reducer, according to Hadoop MapReduce Systems. The data is divided into multiple subsets and fed to multiple mappers, which then feed their output to the reducer. This goes on for multiple iterations till the optimal solution is reached. By going in with this approach, we try to enhance the standard k-means clustering algorithm by transforming its initialization step and amplifying its efficiency using the distributed Hadoop MapReduce Cluster environment. The initialization step introduces the polynomial regression curve that runs through the clusters. The idea being, the regression curve has a tendency to pass through the center of the overall cluster. When the initial points are assigned around this curve randomly, it provides a great chance for the algorithm to converge efficiently, giving a sort of head start to the main algorithm to work on. This provides a more sensible way of choosing the initial cluster on most datasets, as different clusters are segregated from each other in most cases.

The use of Polynomial Regression over Linear Regression is more suited to this algorithm as linear regression creates just a single straight line, whereas polynomial regression creates a curve which can not only be a straight line but also curve to different degrees. The data points in datasets are all over the place, not centered around a single straight line. So a polynomial regression line with appropriate degrees can fit much better on a spread-out dataset. Thus, the curve should pass through the centers of most clusters, where the final centroids should ideally be.

2.1 Algorithm

For the main algorithm, Python's MRJob library framework has been used to write and execute our algorithm in the Hadoop MapReduce environment. This framework provides a Runner which runs a map-reduce job written on a Class that inherits MRJob class from the mrjob library. This class needs to override Mapper, Reducer, and Combiner methods to implement the required logic. It also includes a steps() method for labeling the different methods as Mapper, Combiner, or Reducer, as well as allowing the developer to specify the number of iterations the Mapreduce job should run.

The code in MRJob can run on n-dimensional data points. Libraries like pandas, NumPy, matplotlib and sklearn libraries to process data, calculate efficiently for a range of values and get a visual representation of our results.

Initialization of Centroid

First, the parameters equation degree and k are defined, and then the input data is read. Furthermore, before running polynomial regression, it sorts and preprocesses the data to remove any outliers. Finally, the k centroids located around the generated polynomial regression curve are initialized.

Algorithm 1 CenInit : Centroid initialization algorithm to assign k centroids around the regression curve

---

INPUT:
A dataset of CSV file format, with each line of format (x-coordinate,y-coordinate). This is represented by F.

OUTPUT:
An Initial Centroid file with a single line, containing all the k space-separated Centroid Coordinates of the format (μ_1, μ_2, μ_3,... μ_k) and a text file containing all the data points in the format (DataID| |Coordinates)

Pseudo Code:

```
1. Set dataset = read_csv(F)
2. Set model = PolynomialRegression()
3. model.fit(dataset)
4. Set initialCentroids= k coordinates around the Regression Curve
5. writeToCentroidFile(centroids)
6. writeToFormattedFile(dataset)
```

MapReduce

The pseudocode for the MapReduce algorithm is mentioned below:

Mapper:

Algorithm 2 MapMeans: Calculates the distances from each sample to each centroid point and emits relabeled cluster ID as a key, and data ID, Coordinates as value.

INPUT:
A String of Values of the form "DataID| |Coordinates". Each of these Strings is represented by S.

OUTPUT:
A <key,value> pair of the form (ClusterID,DataID,Coordinates), where key = CusterID and value = DataID, Coordinates. Each of these <key,value> is represented by X.

Pseudo Code:

```
1. Set coordinates = S.coordinates
2. Set centroidsArr = getCentroidsFromFile()
3. Set distance = EuclideanDistancecentroidsArr,coordinates
4. Set ClusterID = min(distance)
5. Generate <key,value> pair of the form (ClusterID, DataID, Coordinates), where key =
   CusterID and value = DataID, Coordinates
```

Combiner:

Algorithm 3 ComMeans: Combine samples with the same key together for each machine and load in what the mapper emits. Outputs Cluster ID as the key, members ID of that cluster, combination of samples with the same key, member coordinates of that cluster as value.

INPUT:
A <key,value> pair of the form (ClusterID,DataID,coordinates), where key = CusterID and value = DataID, Coordinates. Each of these <key,value> is represented by X

OUTPUT:
A <key,value> pair of the form (ClusterID,coordinatesSum,DataIdSet,coordinatesSet), where key = CusterID and value = coordinatesSum,DataIdSet,coordinatesSet. Here coordinatesSum is the sum of all the coordinates and, coordinatesSet and DataIdSet are the set of coordinates and DataID respectively for the given ClusterID

Pseudo Code:

```
1. Set coordinatesSum = {0.0, 0.0}
2. Set DataIdSet,coordinatesSet = {}
3. for each X.DataID, X.coordinates
           do
           coordinatesSet.insert(X.coordinates)
           DataIdSet.insert(X.DataID)
           coordinatesSum = coordinatesSum + X.coordinates
   end for
4. Generate <key,value> pair of the form
   (ClusterID,coordinatesSum,DataIdSet,coordinatesSet), where key = CusterID and value =
   coordinatesSum,DataIdSet,coordinatesSet
```

Reducer:

Algorithm 4 RedMeans:Centroid for each cluster and outputs exactly the same format of data as input file except centroid, cluster ID has been updated.

INPUT:
A <key,value> pair of the form (ClusterID,coordinatesSum,DataIdSet,coordinatesSet), where key = CusterID and value = coordinatesSum, DataIdSet, coordinatesSet. Each of these <key,value> is represented by X

OUTPUT:
A String of Values of the form "DataID|ClusterID|Coordinates". Each of these Strings is represented by S.

Pseudo Code:

```
1. Set finalCoordinatesSet, finalDataIdSet = {}
2. finalCoordinatesSum = {0.0, 0.0}
3. For each X.coordinatesSum, X.DataIdSet, X.coordinatesSet
           do
           finalCoordinatesSet.add(X.coordinatesSet)
           finalCoordinatesSum = finalCoordinatesSum + X.CoordinatesSum
           finalDataIdSet = finalDataIdSet + X.DataIdSet
   end for
4. Set n = finalDataIdSet.length
5. Set newCentroid = finalCoordinatesSum / n
6. writeCentroidsToFile(newCentroid)
7. for each DataID in DataIdSet
           do
           Generate a String of Values of the form "DataID|ClusterID|Coordinates"
```

3. Results

3.1 Data Set

The STARNET Dataset
The Thundering dataset contains long-range lightning detection data from the High Altitude Ice Crystals-High Ice Water Content (HAIC-HIWC) project, which took place in Cayenne, French Guiana. [13] The Sferics Timing and Ranging Network (STARNET) lightning detection network is operated by the University of Sao Paulo in Brazil, at the Laboratorio de Sensoriamento Remoto de Tempestades STORM-T, Department of Atmospheric Sciences, IAG, and contains 136,188 data points, 29 of which were extracted and saved in CSV format.

Simulated Dataset
A dataset of 10,000 data samples with dimension 25 was manually generated, centered on three centroids. Three centroids were arbitrarily selected in the dataset, and data points were fairly spread around those centroids. As a result, obtaining multiple well-separated clusters to aid in determining optimal clustering. It contained 9,999 data points and was saved in a CSV file.

3.2 Comparative Study

Our algorithm, as well as the simple k-means and its variants k means, k means++, k means*+, and k means**, was run on both datasets with two different values of k; k = 5 for the simulated dataset and k = 10 for the STARNET dataset. For each dataset, ten full runs of the algorithm and k-value combinations were executed, with the average error, minimum error, and average runtime recorded.
The following tables summarize the findings.

Algorithm	Average Error	Minimum Error	Average Time
K MEANS	17.60	10.12	5.20
K MEANS ++	28.09	10.44	4.96
K MEANS**	19.87	10.18	3.65
K MEANS+*	14.32	10.09	4.54
MR-KCLUST	10.95	10.04	4.87

Table 1. Simulated dataset, k = 5, n = 10,000, d = 25

Algorithm	Average Error	Minimum Error	Average Time
K MEANS	72.48	47.63	47.61
K MEANS ++	56.09	46.24	82.02
K MEANS**	154.37	49.02	49.44
K MEANS+*	58.43	46.54	54.90

MR-KCLUST	47.01	44.31	57.89

Table 2. STARNET dataset, k = 10, n = 136,188, d = 29

In the case of the K MEANS algorithm, an average efficiency in terms of runtime and error is observed.. It takes the most time in a smaller dataset with mediocre performance on average and a minimum error rate. Its runtime improves and it is the fastest with larger datasets, but it has a poor average and minimum error metric. The error performance metrics for the K MEANS++ algorithm are better in larger datasets. In larger datasets, while the average and minimum error rates are acceptable, the time required is significantly greater. The average and minimum error metrics are high in the case of a smaller dataset, with an average time taken. For the K MEANS** algorithm, it has good performance when it comes to computational time but produces inefficient results in error metrics. Although it is one of the quickest algorithms, the results have a high average and minimum error. In the case of the K MEANS*+ algorithm, it performs well in both error and runtime metrics. Although K MEANS*+ outperforms K MEANS++ in both larger and smaller datasets, K MEANS++ is more accurate in larger datasets. As a result, this algorithm still has loopholes and can be improved. The MR-KCLUST algorithm provides an ideal balance of error and runtime metrics. It has the lowest average error in both the datasets. It outperforms most of its competitors in terms of error metrics and speed on smaller and larger datasets.

4. Conclusion

A new approach to MR-KClust was presented, which adopted a different initialization technique to better predict the centroids of the clusters. Furthermore, the Hadoop MapReduce Cluster environment was used, which divided the data into subsets and fed them through a number of mappers, each of which fed its output to a reducer in a series of iterations until the best solution was found. Due to this parallel computing, there is a considerable difference in computational time when more clusters are added to the multi-node Hadoop system. MR-KClust along with standard K-Means and their variants were implemented and tested on two different datasets. The distributed computing versions performed similarly to their standard counterparts. Only MR-KClust, however, could produce a well-balanced result. In terms of average and minimum errors, it outperformed all other models and ranked among the fastest algorithms.

References

[1] Kalbandi, Ishwarappa & Anuradha, J.. (2015). A Brief Introduction on Big Data 5Vs Characteristics and Hadoop Technology. Procedia Computer Science. 48. 319-324. 10.1016/j.procs.2015.04.188.

[2] Bhosale, Harshawardhan S., and Devendra P. Gadekar. "A review paper on big data and hadoop." International Journal of Scientific and Research Publications 4.10 (2014): 1-7.

[3] K. Shvachko, H. Kuang, S. Radia and R. Chansler, "The Hadoop Distributed File System," 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), 2010, pp. 1-10, doi: 10.1109/MSST.2010.5496972.

[4] Jens Dittrich and Jorge-Arnulfo Quiané-Ruiz. 2012. Efficient big data processing in Hadoop MapReduce. Proc. VLDB Endow. 5, 12 (August 2012), 2014–2015. https://doi.org/10.14778/2367502.2367562

[5] Lee, Kyong-Ha & Lee, Yoon & Choi, Hyunsik & Chung, Yon & Moon, Bongki. (2011). Parallel Data Processing with MapReduce: A Survey. SIGMOD Record. 40. 11-20. 10.1145/2094114.2094118.

[6] Beakta, Rahul. (2015). Big Data And Hadoop: A Review Paper. international journal of computer science & information te. 2.

[7] Stuart P Lloyd. Least squares quantization in pcm. Information Theory, IEEE Transactions on, 28(2):129–137, 1982.

[8] Aristidis Likas, Nikos Vlassis, Jakob J. Verbeek,The global k-means clustering algorithm, Pattern Recognition,Volume 36, Issue 2,2003,Pages 451-461,ISSN 0031-3203,https://doi.org/10.1016/S0031-3203(02)00060-2.

[9] M. K. Pakhira, "A Linear Time-Complexity k-Means Algorithm Using Cluster Shifting," 2014 International Conference on Computational Intelligence and Communication Networks, 2014, pp. 1047-1051, doi: 10.1109/CICN.2014.220.

[10] S. Na, L. Xumin and G. Yong, "Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm," 2010 Third International Symposium on Intelligent Information Technology and Security Informatics, 2010, pp. 63-67, doi: 10.1109/IITSI.2010.74.

[11] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[12] Bodoia, Max. "MapReduce Algorithms for k-means Clustering." accessed online at: https://stanford. edu/~ rezab/classes/cme323 S 16.

[13] Sferics Timing and Ranging Network (STARNET) long-range lightning detection data from the High Altitude Ice Crystals - High Ice Water Content (HAIC-HIWC) project that took place in Cayenne, French Guiana. https://data.ucar.edu/dataset/starnet-lightning-data-dat