Improvements and Implementation of Hierarchical Clustering based on Hadoop

Jun Zhang[1, a], Chunxiao Fan[1], Yuexin Wu[2,b], Ao Xiao[1]

[1] School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing, 100876 China

[2]Institute of Sensing Technology and Business, BUPT, Wuxi, China

[a]email: aedime@yeah.net, [b]email: wuyuexin@bupt.edu.cn

Keywords: Hierarchical Clustering; Hadoop; MapReduce

Abstract. As the traditional agglomerative hierarchical clustering has a higher number of iterations which makes low efficiency of parallel realization on Hadoop, we propose an improved hierarchical clustering method: when the between-class distance is monotonically increasing, by changing the clustering order of hierarchical clustering without changing the final clustering result, aggregate multiple classes in a MapReduce operation, to reduce the number of iterations then enhance the computational efficiency. The experiments show compared to traditional hierarchical clustering algorithm implemented in Hadoop, the improved algorithm implemented in Hadoop has greatly reduces the number of iterations and the computation time.

Introduction

Clustering is a fundamental data mining algorithms and hierarchical clustering is a commonly used clustering algorithm. Hierarchical clustering has the advantages of intuitive, no need to set the number of clustering, only need the similarity between the clustering object, e.t. When cluster large date on a single machine, it’s easy to meet the problems of memory bottleneck and long processing time, which can be solved by using parallel realization.

Hadoop is a software framework for distributed storage and processing of very large data sets. It has the advantages of highly scalable, cost effective, efficient calculation, resilient to failure. The core of Hadoop consists of Hadoop Distributed Files System (HDFS) and MapReduce. In HDFS, data is cut into small blocks, then each block will be replicated and stored on three different nodes, which can improve the reliability and move the calculative ability to data to save time. MapReduce is an efficient distributed computing framework, users don’t need to consider the underlying implementation details, only need to write map and reduce functions. MapReduce use the key-value pairs to process and transfer data in map and reduce. The map function splits large data into small pieces and distributed them to different nodes to process, and the intermediate results are submitted to reduce function in form of key-value pairs to get the final result.

However, as compared to other clustering, hierarchical clustering algorithm has higher iterations, while Hadoop is not friendly to the iterative computation, and the efficiency of the traditional hierarchical clustering algorithm is lower in Hadoop. In this paper, under the precondition of not changing the clustering results and the between-class distance is monotonically increasing, by changing the order of clustering, aggregate multiple classes in an aggregating process to reducing the number of iterations, then enhance the computational efficiency. The experiments show that the improved hierarchical clustering algorithm implemented in Hadoop compared to unimproved hierarchical clustering algorithm implemented in Hadoop has less calculating redundancy and is significantly improved in terms of time efficiency.

Hierarchical Clustering and the Problem with Hadoop

Hierarchical clustering (HC) decomposes each object based on the distance between each object. There is two kinds of Hierarchical clustering, agglomeration and division. The agglomerative hierarchical clustering is a "bottom up" approach: each object is treated as a separate category, then continued cluster the nearest couple object into one class until meet the stopping condition. Where the divisive hierarchical clustering is a "top down" approach: it considers all the object as one class and divides it into small classes. Consider the agglomerative hierarchical clustering applications more, this paper discusses only the agglomerative hierarchical clustering.

The basic steps of original agglomerative hierarchical clustering (i.e. single-class hierarchical clustering) are as follows:

Step 1. Regard each object as one class, calculated the distance between each two classes.

Step 2. Find the nearest two classes in all classes and merge them into one class.

Step 3. Recalculate the distance between each two classes.

Step 4. If meet the stopping condition, then stop, otherwise continue with step 2.

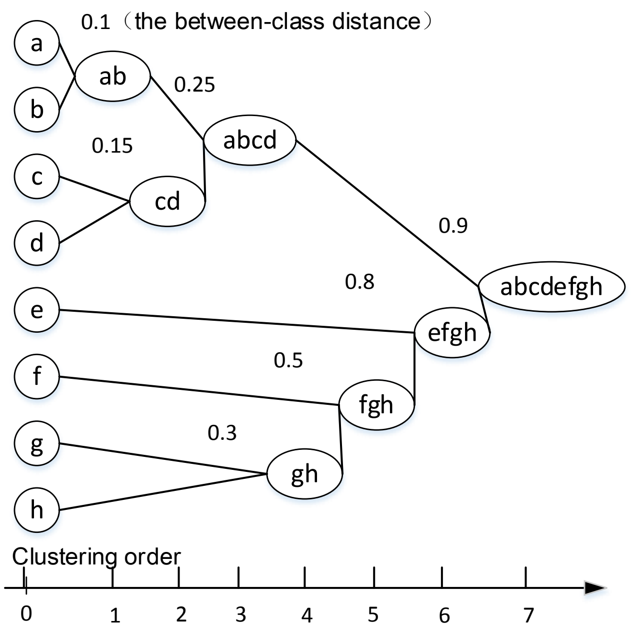


Fig. 1 Unimproved HC's clustering tree and order

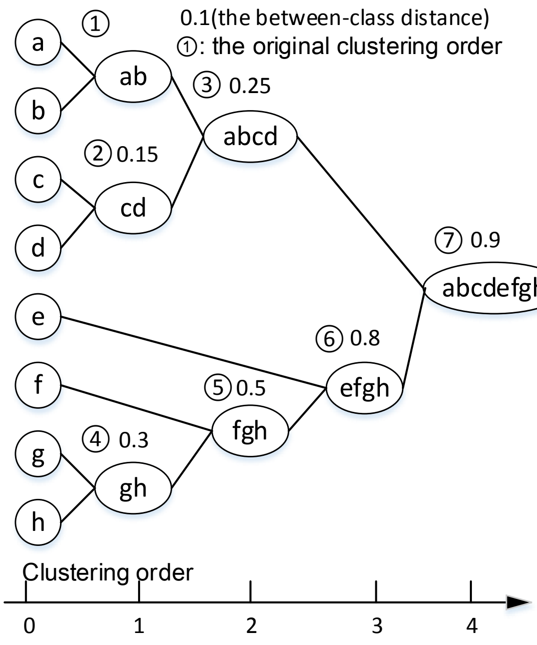


Fig. 2 Improved HC's clustering tree and order

Figure 1 is the clustering tree and clustering sequence of hierarchical clustering, from the clustering tree we can very clearly see the relationship between the classes. As can be seen from the figure, if the initial number of classes is n, we will need n-m times clustering to merge the classes into m classes. Generally speaking, n-m is a large number, and the bigger the initial point number is the more times of iterations is required, as a result the original hierarchical clustering needs a large number of iterations.

But excessive iterative calculations will reduce the computational efficiency for hierarchical clustering achieved base on Hadoop, which is due to 1, for Hadoop, each iteration includes one or more job, each job has to made communication, resource allocation and data transfer which is essential in Hadoop, but is not necessary to the calculation. When the number of iterations is large, the total job's start-up cost of time cannot be ignored. 2, in one hierarchical clustering process, most of the distances between the classes doesn't change, but still need to read, transmit and re-storage them, which will cost a lot of time. 3, since in HDFS, the data is divided into small blocks and storage distributed, and HDFS doesn't support update partial data, these made us can't design the data storage structure to reduce the computational redundancy.

Therefore, to improve the computational efficiency of hierarchical clustering algorithm on Hadoop, one is reducing the time spend on each iteration by optimize computing architecture according to the principle of Hadoop; the second is to reduce the number of iterations by follow the principle of the algorithm. This paper use the second way to design a new method of hierarchical clustering Hadoop implementation, which can significantly reduce the the number of iterations and improve computational efficiency.

Improvement of Hierarchical Clustering Algorithm

From the basic idea of hierarchical clustering and analyzing Figure 1 we can conclude that:

If the between-class distance is monotonically increasing, then:

Class A and B are each other nearest $\Leftrightarrow$ Class A and B will first be aggregated into AB, then, aggregated with other classes. (1)

Simple proof of Eq. 1 are given below:

Necessity: from the clustering condition of hierarchical clustering, easy knowing it's true.

Sufficiency: Analysis: just need to prove if at the start time, class A and B are each other nearest, until they are merged, class A and B are still each other nearest. Since class A and B are each other nearest, they won't merge with other class and the distance is monotonically increasing, so the minimum distance between all the classes will continue to increase until the A and B became the nearest pair then aggregate into AB.

Let: D (A, B) represent the distance between class A and B; C, D are two different classes except A, B, and CD represents the class C and D aggregate into. D (CD, A) represents the distance between class A and CD.

Proof: because the class A and B are each other nearest $\Leftrightarrow$ D (A, C), D (B, C) $\geqslant$ D (A, B)

And because the between-class distance is monotonically increasing, which means D (CD, A) $\geqslant$ min {D (C, A), D (D, A)}

Therefore, D (CD, A) $\geqslant$ min {D (C, A), D (D, A)} $\geqslant$D (A, B)

Thus, D (CD, A) $\geqslant$D (A, B), proving by the same methods D (CD, B) $\geqslant$D (A, B)

So in the process of clustering in, A, B are still each other nearest.

Therefore, classes A and B will first be aggregated into AB, then, aggregated with other classes. QED.

When the distance is not monotonically increasing, there may be D (CD, A) <min {D (C, A), D (D, A)}, and D (CD, A) <D (A, B), which making the Eq. 1 does not hold. So the improved hierarchical clustering method is only applicable to the class distance is monotonically increasing. When use the shortest distance method, the longest distance method, the group average method (i.e. the average distance method), and the variable sum of squared residuals method, the between-class distance is monotonically increasing[5] and can use the improved method, while the middle distance method and gravity method are non-monotonic[5] and can't use the improved method.

Since the method with monotonous distance have more number, better performance and more commonly used, the proposed method has a large applicable scope.

Based on the above analysis, we redesigned the clustering process of hierarchical clustering, as shown in the Figure 2, it has the same clustering objects and clustering results with the Figure 1, the difference is in each clustering process, all the nearest classes of each other will respectively cluster into one class, so in a calculation process can aggregate multiple classes, then reducing the number of iterations. Although we change clustering sequence, but did not change the results of clustering: by recording each clustering distance, we can easily get the original clustering order to generate clustering tree.

When the between-class distance is monotonically increasing, without changing the clustering results, this paper presents an improved hierarchical clustering (i.e. multiple-class hierarchical clustering) algorithm and step as follow:

Step 1. Regard each object as one class, calculate the distance between each two classes.

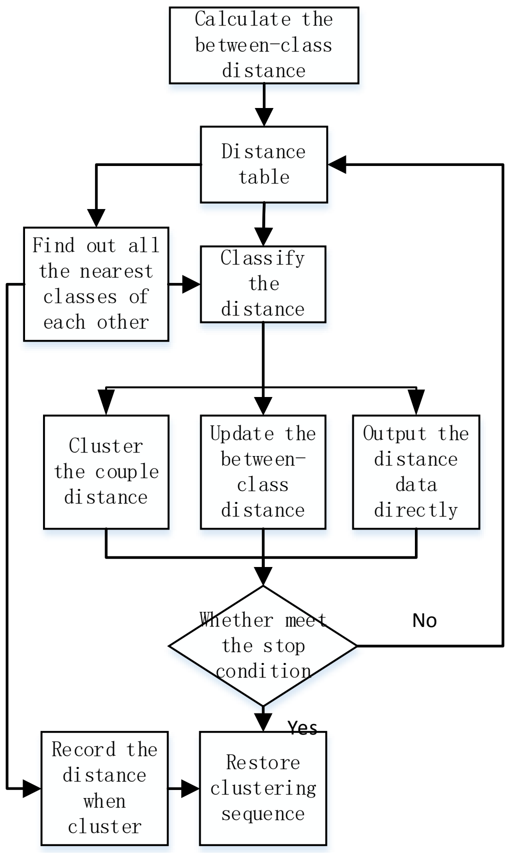
Step 2. Find all the nearest classes of each other in all classes.

Step 3. Merge all the nearest classes of each other respectively into one class, record the distance when merge. Recalculate the distance between each two classes.

Step 4. If meet the stopping condition, then stop, otherwise continue with step 2.

The Improved Hierarchical Clustering Achieved on Hadoop

The overall framework of the program shown in Figure 3, the initial module will generate the raw data into distance table, according the table find out all the nearest classes of each other, use it to divide the distances in distance table into three kind: 1, need to aggregation; 2, need to update the distance; 3, do nothing, then process them separately and generate a new distance table. Continue to iterate until satisfy the stop condition.



Calculate the between-class distance
Distance table
Find out all the nearest classes of each other
Classify the distance
Cluster the couple distance
Update the between-class distance
Output the distance data directly
Whether meet the stop condition
No
Yes
Record the distance when cluster
Restore clustering sequence

Fig. 3 The improved HC's block diagram

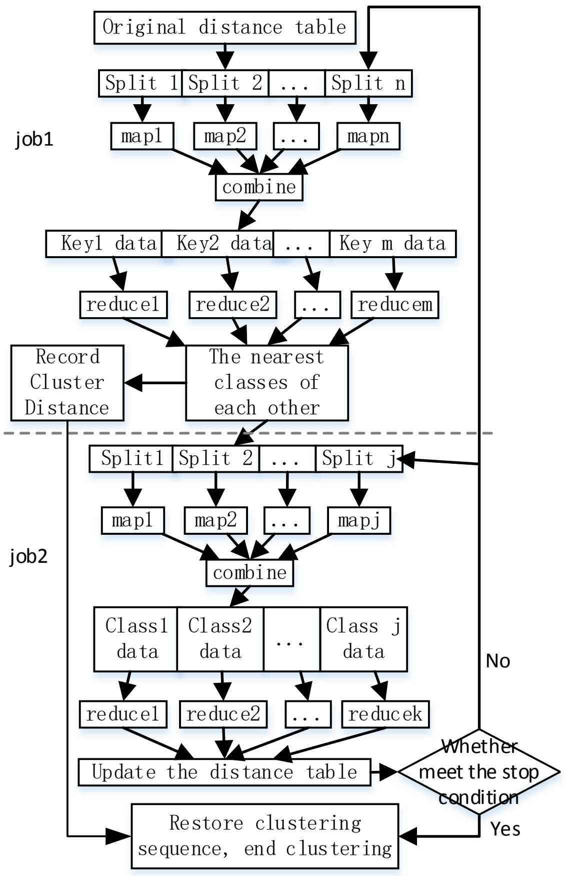


Fig. 4 The improved HC's MapReduce flow

The improved hierarchical clustering's MapReduce flow are shown in Figure 4. The entire program consists job 1, job 2, circulation control procedures etc. Job 1 complete search all the nearest classes of each other, job 2 complete cluster and update the distance table.

Job 1's Map function find each class's local nearest class in each data piece. The Reduce function gather the Map output, then find the each class's nearest class and find out all the nearest classes of each other.

Job 2's Map function will divide the distance table's distance into three categories according the all the nearest classes of each other table of job 1. According to the classification, the Reduce function will respectively merger classes, updating distance or direct output the distance.

Job 2's pseudo-code of Map function and Reduce functions are as follows:

Job 2's Map function's pseudo-code:

Judge each row of distance table by use the nearest classes table of job 1's output, if two classes: class A, class B are both in the table (means A and B will be merge with some classes, but not necessary A and B together), then key = 0; if only one of them is in the table, key = 1; otherwise the key is Random number greater than or equal to 2 (random number is for load balancing). Made the classes name as the key, the distance between the classes and the number of the points within the class as the value, output the <key, value> to the Reduce function.

Job 2's Reduce function's pseudo-code:

Judge the key value, if the key = 0, store the data into hash table, traversing value lists, use Eq. 3 to calculate the distance between the merged classes, merge combination, update distance; if key =1, store the data into hash table, traversing value lists, use Eq. 2 to calculate the distance between the merged class and non-merge class, merge combination, update distance; If the key is greater than or equal to 2, direct output data.

There is a corresponding recurrence formula to calculate between-class distance, so we can calculate the distance of between (AB) and C(C is merged by C_1, C_2, C_3) by using D_{AC} and D_{BC}, will not need the D_{AC1}, D_{AC2}, … D_{BC3}. It can reduce the calculation. For example, the average distance method's ordinary recurrence formula is Eq. 2 and to adapt to the improved algorithm, the transformation is Eq. 3. The other types of between-distance calculation method are similar to these.

$$D_{(AB)C} = (N_A N_C D_{AC} + N_B N_C D_{BC})/(N_A N_C + N_B N_C). \tag{2}$$

$$D_{(AB)(CD)} = (N_A N_C D_{AC} + N_B N_C D_{BC} + N_A N_D D_{AD} + N_B N_D D_{BD})/(N_A N_C + N_B N_C + N_A N_D + N_B N_D). \tag{3}$$

In the above formula, D_{AC} refers to the distance between class A and B, (AB) refer to the class which A and B merged into, $D_{(AB)C}$ refer to the distance between class AB and C, N_C refer to the original class number of class C(for here, it's 3). Eq. 2 is to calculate the between-class distance of only one class is new aggregate, while the other is old. We generalize Eq. 2 to Eq. 3 to calculate the distance when both classes are new aggregate. For the other method, it's easy to get the generalized formula.

Experiment and Analysis

In this paper, the experimental environment is Aliyun' ODPS (Open Data Processing Service), which is built on Hadoop framework and has the similar programming and implementation. The following data is measured at the same hardware environment, so the data is comparable. The data use to cluster is processing the user's Internet browse and purchase records to get the similarity between each goods brand and use the similarity's reciprocal as the between-class distance. Since we have only the distance between brands, so cannot use K-means clustering algorithm, etc., so select the hierarchical clustering algorithm to get the clustering result between brands.

We implement two procedures of hierarchical clustering algorithm in Hadoop, one is the original, and the other is the improved hierarchical clustering, then compare and analyzed their performance. The original hierarchical clustering on Hadoop is simple in algorithm and procedure, which job 1 finds the minimum between-class distance, namely the the nearest classes of each other, job 2 updates the distance table, and we won't do detailed instructions here.

Figure 6 analyzes the once aggregation time, average time of aggregate one class and the performance improved times between the single-class hierarchical clustering (i.e. the original hierarchical clustering) and multiple-class hierarchical clustering (i.e. the improved hierarchical clustering) when in different data size, i.e., the class number need to cluster. In the multiple-class hierarchical clustering, once aggregation aggregate 500 classes. The average time of aggregate one class is for the multiple-class hierarchical clustering and is equal to the once aggregation time divide the class number of once aggregation aggregate (inhere, it is 500). In figure 6, the average time of aggregate one class is too small and can hardly see from the figure, but we have number there. When the total class number is about 37 thousand, there are 0.35 billion row data (Since removed some too big distance and missing data, the number is less than the theoretical value).

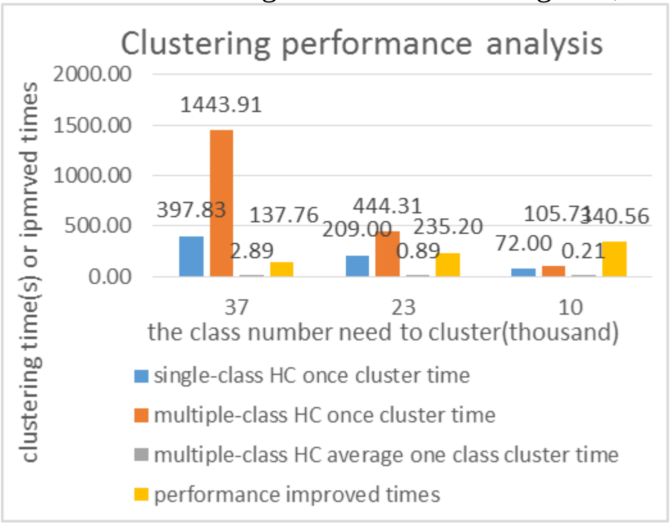


Fig. 5 HC procedure performance comparison

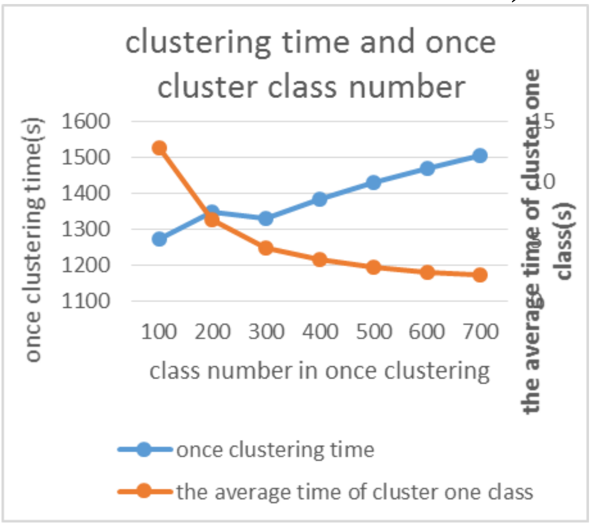


Fig. 6 Once cluster number and time

As can be seen from figure 6, multiple-class hierarchical clustering take more time in once aggregation time, but in average time of aggregate one class is much less and is second level, which is single-class hierarchical clustering unable to achieve because each job will cost necessary time but have nothing to do with calculate and it is at least several seconds. So the multiple-class hierarchical clustering has obvious advantages in time efficiency compared to single-point hierarchical clustering on Hadoop.

In figure 7, we control the class number of once aggregation aggregate, the total class number is about 37 thousand, the other things being equal. We can see that the once aggregation time is increased with aggregate number increasing, but the average time of aggregate one class is reduce. When the aggregate number is 700, the reduction of average time is already small, because it has been closer to the limit of the system and procedures, further increase the number of aggregate to reduce the time it would not be very helpful.

In this test data, when the total class number is about 37 thousand there can find out more than one thousand pairs of the nearest classes of each other, and in the process of clustering the number has remained relatively stable, which can guarantee there are enough classes to aggregate in once clustering, so efficiency of the improved clustering algorithm is guaranteed.

Conclusion

Through the above analysis and experimental testing can be seen, this paper proposed multiple-class hierarchical clustering after analysis the advantages and disadvantages of Hadoop and MapReduce, and also the principle of hierarchical clustering algorithm. In between-class distance is monotonically increasing and do not change the premise of clustering results, design and implement the clustering algorithm to optimize the aggregate sequence: in an aggregating process aggregate multiple classes. Compared to the Hadoop implementation of the single-point hierarchical clustering (i.e. the original hierarchical clustering), the multiple-class hierarchical clustering (i.e. the improved hierarchical clustering) greatly reduces the number of iterations, improve the computational efficiency of the program, reducing the computation time obviously.

Acknowledgement

The work presented in this paper was supported by the National Great Science Specific Project (Grants No. 2014ZX03002002-004) and National Natural Science Foundation of China (Grants No. NSFC-61471067).

References

[1] Sun T, Shu C, Li F, et al. An efficient hierarchical clustering method for large datasets with map-reduce[C]//Parallel and Distributed Computing, Applications and Technologies, 2009 International Conference on. IEEE, 2009: 494-499.

[2] Jin C, Patwary M M A, Agrawal A, et al. DiSC: A Distributed Single-Linkage Hierarchical Clustering Algorithm using MapReduce[J]. work, 2013, 23: 27.

[3] TU Jinjin, YANG Ming, GUO Lina. A density-based hierarchical clustering algorithm of gene data based on MapReduce [J]. Journal of University of Science and Technology of China, 2014, 7: 001.

[4] ZHOU Ting, ZHANG Junying, LUO Cheng. Realization of K-means Clustering Algorithm Based on Hadoop [J]. Computer Technology and Development, 2013, 23(7): 18-21.

[5] Shao Fengjing, Yu Zhongqing. Principle and algorithm of data mining [M]. Science Press, 2009.