An Improved K-means Distributed Clustering Algorithm Based on Spark Parallel Computing Framework

Xin Lu1, Huanghuang Lu1*, Jiao Yuan2 and Xun Wang2

1 School of software, University of Electronic Science and Technology of China, Chengdu, Sichuan, 610054, China
2 China Railway Eryuan Engineering Group Co.Ltd, Chengdu, Sichuan, 610031, China
* luhuanghuang@qq.com

Abstract. Traditional K-means distributed clustering algorithm has many problems in clustering big data, such as unstable clustering results, poor clustering results and low execution efficiency. In this paper, a density based initial clustering center selection method is proposed to improve the K-means distributed clustering algorithm. The algorithm uses the sample density, the distance between clusters and the cluster compact density, defines the product of the three as the difference weight density, and finds the sample point with the maximum difference weight density as the initial cluster center, so as to solve the problem of randomness and low quality of initial cluster center selection. At the same time, this paper uses spark parallel computing framework to implement the improved algorithm to further improve the processing performance of the algorithm in big data clustering. The experimental results show that the improved k-means distributed clustering algorithm based on spark parallel computing framework has higher execution efficiency, accuracy and good stability in big data clustering analysis.

1. Introduction

With the deepening of social informatization, big data is becoming more and more important in all walks of life. Big data mining analysis can discover the potential rules and relations of data, so as to provide good suggestions for the business development of customers. Clustering analysis is the most commonly used methods in data mining[1]. It gathers data into several clusters composed of similar objects through unsupervised learning, so as to realize data clustering processing. The common clustering analysis methods mainly include partition based clustering (such as k-means[2]), hierarchical clustering (such as birch[3]), density based clustering (such as DBSCAN[4]), grid based clustering (such as sting[5]), etc. With the deepening of the era of big data, the type and amount of data are increasing, and the traditional centralized clustering algorithm has been unable to meet the requirements of users for the accuracy and efficiency of big data clustering analysis. Improving the existing clustering algorithm and realizing distributed parallel computing has become a research hotspot in the field of clustering algorithm [6].

Density based K-means distributed clustering algorithm (DK-means for short) is a kind of algorithm which is suitable for clustering analysis of data with uniform density distribution. The core idea of the algorithm is to calculate the average distance between the samples in the dataset, and calculate the density of each sample through the density function, select the sample with the largest density as the first initial clustering center, and then remove the samples within its average distance
from the dataset, and repeat the above steps until K initial clustering centers are found. The algorithm calculates the density of data samples to find the sample with the largest density as the initial clustering center to improve the accuracy of clustering. Then, the parallel algorithm is realized by combining MapReduce model under Hadoop platform to improve the execution efficiency of clustering. However, when DK-means is used to cluster the data sets with different density distribution, it is easy to select several sample points in the same high-density cluster as the initial clustering center, which is obviously not the optimal initial clustering center. At the same time, the unsolved k-means algorithm causes a lot of I/O consumption when accessing the dataset many times in the iterative process, which results in the low clustering efficiency.

Aiming at the problems of low quality of initial cluster center selection and low efficiency of clustering process in DK-means, this paper proposes an improved k-means distributed clustering algorithm based on spark parallel computing framework (MDDK-means for short). MDDK-means firstly selects K optimal initial clustering centers by density based initial clustering center selection method, which introduces the difference weight density when calculating the sample density. By calculating the difference weight density of samples, the sample point with the maximum value is selected as the next initial cluster center, which helps to solve the problem of low quality of DK-means initial cluster center selection, makes the selection of initial cluster center more reasonable, and lays the foundation for improving the accuracy and execution efficiency of subsequent clustering. At the same time, based on the improved k-means algorithm and spark parallel computing framework, the algorithm is modified to be suitable for parallel computing, which can improve the performance of the algorithm.

2. Related work

2.1. K-means distributed clustering algorithm

K-means clustering algorithm is widely used because of its simple principle, easy implementation and fast convergence speed. However, the shortcomings of traditional K-means algorithm are also very obvious: the random selection of initial clustering center makes the clustering results easy to fall into the local optimal solution and cause the clustering results to be unstable, the implementation efficiency of big data clustering analysis is low. In view of the above problems, many scholars have been improving and optimizing K-means clustering algorithm. Several improved methods are as follows: selecting the optimal initial cluster center based on density[7] and particle swarm optimization[8], selecting the optimal cluster center in the iterative process based on adaptive method[9], changing the distance calculation formula based on specific data set[10], combining with distributed parallel computing framework to optimize and other improved methods.

Xie Xiujuan[7] et al. proposed a density based K-means initial clustering center selection algorithm. It establishes the similarity matrix by calculating the similarity between data objects, and then calculates the average similarity between each data object and other data objects. The data object whose average similarity is higher than the set similarity threshold is regarded as the core object. Take the first core object as the first initial cluster center, and then take the core object which is the most dissimilar as the second initial cluster center, and so on to find K initial clustering centers, and improve the accuracy of clustering by looking for high-quality initial clustering centers. However, this method has high requirements for the setting of similarity threshold. The quality of the setting directly determines the quality of the initial cluster center selection. At the same time, a different similarity threshold is needed for different data sets, which is not universal.

Judith[8] et al. proposed particle swarm optimization (PSO) algorithm to find the best quality heart for k-means clustering, improve the accuracy of clustering, and use the distributed computing and storage provided by Hadoop and MapReduce framework to improve the execution efficiency. The algorithm is easy to fall into the local optimal solution, and it needs to solve the problem of how to determine the parameters. At the same time, the k-means algorithm needs to be iterated for many times,
and each iteration needs to access the data set, while MapReduce framework does not provide memory storage, so the degree of efficiency improvement is limited.

Wang Bo[9] et al. proposed a parallel K-means clustering algorithm with adaptive cuckoo search. By improving the cuckoo search algorithm, it can adjust the search step size adaptively in the search process, and integrate it into the K-means clustering process. In the process of K-means iterative solution, the optimal clustering center is found and the clustering is completed. At the same time, the algorithm is improved by combining MapReduce framework under Hadoop platform. Line parallel design. The algorithm needs to input the number of clusters to be clustered, the maximum number of iterations, the probability of discovery, the maximum step size and the minimum step size. The quality of these input values determines the quality of clustering. There are human factors and the random influence of initial cluster center selection.

Tang[10] et al. proposed an improved k-means algorithm (imr-kca). By analyzing the shortcomings of traditional k-redundancy, the mean algorithm proposed a selection model to simplify the calculation with multiple cluster centers. At the same time, the Euclidean distance is replaced by Manhattan distance to be suitable for clustering analysis of medical data, and the parallel calculation is carried out with MapReduce framework. This improved algorithm is not universal and more suitable for clustering analysis of medical data.

2.2. Spark parallel computing framework

Apache spark[11] is a widely used distributed parallel computing framework. Its most significant advantage is the concept of distributed memory abstraction RDD (resilient distributed dataset). It is an elastic distributed data set to support the application of working set. The distributed cache of data sets in the memory of each work node, the working node only needs to read the working set in memory when executing tasks, and can reuse the results of the working set in the subsequent query. This greatly reduces the large amount of I/O consumption caused by repeatedly writing and reading disk in the calculation process, and greatly improves the efficiency of calculation. This feature of spark is very suitable for many algorithms, such as machine learning and data mining, which need to do a lot of repetitive work. Combining with spark, the efficiency of such algorithms can be greatly improved. The framework of spark distributed parallel computing is shown in Figure 1.

![Spark operation architecture](image)

The computing tasks are deployed to several work nodes in spark framework[12]. They are scheduled by sparkcontext object in the main driver, and connected with the cluster manager to manage and control each work node. Each work node executes tasks independently and in parallel. After the task is completed, the results are summarized and returned to the driver. The detailed running process of spark distributed parallel computing framework is as follows: firstly, the driver creates a sparkcontext, which applies for resources, allocates and monitors tasks. The cluster manager starts the executor process and allocates resources to it. Sparkcontext builds DAG diagram according to RDD dependency, and DAG diagram is parsed to generate stage. A stage is a taskset, and the taskset is
submitted to the taskscheduler for management and monitoring. After that, the executor applies for the task to the sparkcontext, issues the task to the executor through the taskscheduler, and provides the application to the executor. Tasks run in parallel in executors on multiple work nodes. After the task runs in parallel, the running results are fed back and summarized, and the resources are released.

3. Improved k-means distributed clustering algorithm based on density

DK-means needs further improvement in clustering accuracy and execution efficiency. Therefore, this paper proposes an improved k-means distributed clustering algorithm based on density MDDK-means, which introduces the maximum difference weight density to select the optimal initial clustering center. The following gives the definition of the model.

3.1. Definition of K-means distributed clustering model

Suppose that the sample data set is \( X = \{X_1, X_2, X_3, ..., X_N\} \), including \( N \) data objects, where \( X_i = \{x_{i1}, x_{i2}, x_{i3}, ..., x_{iq}\} \), \( q \) represents the number of attributes of data objects in the sample data set.

Definition 1: the Euclidean distance between any two data objects. Its calculation is shown in Formula 1.

\[
\text{dist}(X_j, X_i) = \sqrt{\sum_{n=1}^{q} (x_{jn} - x_{in})^2}
\] (1)

Euclidean distance is used to represent the similarity between two data objects. The larger the Euclidean distance between them, the more dissimilar they are. On the contrary, it means that the two are more similar.

Definition 2: the average Euclidean distance between all data objects in the data set. Its calculation is shown in formula 2.

\[
\mu \text{ dist} = \frac{1}{N(N-1)} \sum \text{dist}(X_j, X_i)
\] (2)

Definition 3: the density of data objects, and its calculation is shown in Formula 3.

\[
\text{density}(X_j) = \sum_{j=1}^{N} \lambda(\text{dist}(x_i, x_j) - \mu \text{ dist})
\] (3)

Where \( \lambda(y) \) is the function defined in this paper, it is 0 if \( Y > 0 \), and 1 if \( Y \leq 0 \). Represents the number of other data objects within the average Euclidean distance of the data object.

Definition 4: use the cluster center distance to calculate the distance between clusters, and the calculation is shown in formula 4.

\[
\text{disp}(X_i) = \sum_{j=1}^{N} \text{dist}(X_i, Y_j)
\] (4)

\( Y \) is the selected initial clustering center set. The distance between two clusters is taken as the distance between clusters. The larger the distance between clusters, the greater the difference between the two clusters. The total difference between the cluster and the selected initial clusters is obtained by adding the distances between the cluster and the selected initial clusters.

Definition 5: the average distance within the cluster, and its calculation is shown in formula 5.

\[
\text{edist}(X_i) = \frac{1}{N_{i}} \sum \text{dist}(X_{ia}, X_i)
\] (5)

\( X_{a} \) is the data object in the cluster centered on \( X_i \), and \( N_i \) is the total number of data objects in the cluster.

The average distance between each data object and the center is taken as the average distance in the cluster. The smaller the average distance within the cluster, the more similar the data objects in the cluster, and the greater the cluster density. Therefore, the reciprocal of the average distance within a cluster is taken as a measure of cluster compactness.

Definition 6: cluster compact density, its calculation is shown in formula 6.

\[
\text{Ccom}(X_i) = \frac{1}{\text{edist}(X_i)}
\] (6)
Definition 7: difference weight density, its calculation is shown in formula 7.
\[
\text{Mdd} \ (X_i) = \text{density} \ (X_i) \times \text{disp} \ (X_i) \times C\text{com} \ (X_i)
\]  
(7)

In order to solve the problem that the quality of selecting the initial cluster center is not high in DK-means, this paper uses the maximum difference weight density to find the optimal initial cluster center. The improved idea is to find the points with higher density, more compact density and greater distance between clusters as the initial clustering center in the data set, and defines the product of density, distance between clusters and compact density as the difference weight density, the point with the maximum difference weight density is the optimal initial clustering center.

The Euclidean distance between any two data objects in the sample data set is calculated by formula (1), and the average Euclidean distance of the sample data set obtained by formula (2) is taken as the search radius, and the points within the search radius form a cluster with each data object as the center. The density of each data object is calculated by formula (3), and the data object with the largest density is selected as the first initial cluster center. For the rest of the data objects, the distance between clusters and the cluster compact density formed by the determined initial cluster center are calculated by formula (4) (5) (6), and the difference weight density of data objects is calculated by formula (7), and the data object with the maximum value is selected as the next initial cluster center. And so on until K initial cluster centers are selected.

MDDK-means firstly makes the optimal selection of the initial cluster center, which makes the selection of the initial cluster center no longer random, and improves the stability of clustering. The core idea of clustering algorithm is to cluster similar objects into a cluster, so that the objects in the same cluster are as similar as possible, and the objects in different clusters are not as similar as possible. Therefore, the distance between clusters is introduced into the selection of initial clustering centers, which makes the selected initial clustering centers as dissimilar as possible, which conforms to the core idea of clustering algorithm and improves the accuracy of clustering results. At the same time, for k-means algorithm, the sum of squares of errors (SSE) is used as the objective function to measure the clustering quality, which is to calculate the sum of squares of distances between the samples in the cluster and the center. The smaller the SSE, the better the clustering quality and the higher the compactness of clustering. Therefore, the cluster compact density is introduced into the selection of the initial cluster center, which makes the selected initial cluster center form clusters with higher cluster density, reduces the number of clustering iterations, and improves the clustering execution efficiency.

3.2. MDDK-means algorithm
In order to realize the initial clustering center selection method based on the maximum difference weight density, this paper gives the pseudo code description of MDDK-means as follows.

Input: Data ← sample data set, K ← cluster number
Output: Centers ← a set of K initial clustering centers

\[\text{dist} \ (X_i, X_j) \leftarrow \text{calculates the Euclidean distance between any two objects in the data}\]
\[\mu \text{dist} \leftarrow \text{calculate the average Euclidean distance of data}\]
for i 1 to the number of data in the sample dataset by 1 do
\[\text{density} \ (X_i) \leftarrow \text{calculates the density of each data object}\]
\[\text{cdist} \ (X_i) \leftarrow \text{calculates the average distance within the cluster for each object}\]
\[\text{Ccom} \ (X_i) \leftarrow \text{computes the cluster density of each object}\]
end

Centers ← data objects with the highest density
for n 1 to K-1 by 1 do
\[\text{disp} \ (X_i) \leftarrow \text{calculates the cluster distance between each data object that does not belong to Centers}\]
\[\text{and the objects in Centers}\]
\[\text{Mdd} \ (X_i) \leftarrow \text{calculates the difference weight density of each data object that does not belong to}\]
\[\text{Centers}\]
\[\text{Centers} \leftarrow \text{Centers} + \text{data objects that do not belong to Centers and have the largest Mdd}\]
4. Implementation of MDDK-means algorithm based on spark parallel computing framework

Through the analysis of the implementation process of K-means algorithm, we can see that the traditional K-means algorithm is inefficient when clustering large-scale data. The main reason is that each iteration needs to read and write data sets and calculate the distance between each data object and the cluster center in the process of iterative solution, resulting in a lot of I/O consumption and calculation consumption. Therefore, the key to improve the execution efficiency of K-means algorithm is to parallelize the iterative process and reduce the read and write operations on disk. In this paper, MDDK-means is implemented in parallel with spark distributed parallel computing framework. The parallel algorithm can be implemented in two stages: searching for the initial clustering center and K-means clustering.

4.1. Definition of K-means distributed clustering model

In the stage of finding the initial cluster center, the optimal initial cluster center is found by using the weight density of maximum difference degree, and the phase is parallelized. The flow chart is shown in Figure 2.

![Figure 2. Flow chart of finding initial cluster center.](image)

In the stage of searching for the initial cluster center, the parallel implementation is as follows: first, spark cluster reads the data set used from HDFS to form the initial RDD. The initial RDD calls the map operator to convert the initial data set into key value pairs. At the same time, the dense method of vectors object is executed to further process the key value pairs into vector data key value pairs used for subsequent calculation to form RDD1. RDD1 starts the map operator to calculate the density and cluster density of each data object in the dataset to form RDD2 and RDD3, which are saved as key value pairs. After that, RDD2 starts the sortby sorting to get the data object with the maximum density and adds it to CT (the initial cluster center set).

RDD1 starts the map operator to calculate the distance between each data object and the data object in CT to form RDD4, which is saved as a key value pair. RDD4 starts the join operator, merges RDD2 and RDD3, and the form of RDD4 key value pair is \(<\text{data object, (density of the data object, cluster density of the data object, distance between the data object and the selected initial cluster center)})>. RDD4 starts reducebykey to calculate the difference weight density of data objects, and gets the data object with the maximum difference weight density by sorting, and adds it to the initial cluster center set CT. Repeat the above steps until the number of data objects in CT is the set number of clusters K, and the optimal initial cluster center selection phase is completed.

4.2. K-means clustering stage

In the K-means clustering stage, the data objects selected in the stage of searching for the initial cluster center are used as the initial clustering center to cluster. The flow chart is shown in Figure 3.
At the beginning of K-means clustering stage, the cluster obtains the initial cluster center set CT and broadcasts it to all working nodes. RDD1 starts the map operator, calculates the distance between each data object and each cluster center in CT, and allocates each data object to the nearest cluster center to form RDD_cd, saved as <cluster center, data object> key value pair. RDD_cd starts groupbykey to collect data objects with the same cluster center to form RDD_cdg. RDD_cdg starts reducebykey, recalculates the cluster center, and judges whether the clustering result converges. If it converges, the algorithm ends; if not, it uses the new cluster center to repeat the above process until the clustering result converges.

5. Algorithm experiment and analysis

Compare MDDK-means with traditional K-means and DK-means to verify whether the algorithm is improved in accuracy and performance. The accuracy and speedup ratio are used to compare the algorithm. Among them, accuracy = the amount of data correctly clustered / the total amount of data in the data set. Speedup ratio = execution time of the same dataset in stand-alone mode / cluster mode.

5.1. Experimental scheme

In the algorithm experiment, build a spark cluster platform. It includes one master node and four slave nodes. Each node is configured with 16GB memory and 100GB hard disk, while Hadoop version is 3.1.1, and spark version is 2.3.2. This paper will compare and analyze the performance of each algorithm in accuracy through the following two experiments, and verify the parallel execution performance of the algorithm.

(1) In Experiment 1, three common test data sets of UCI machine learning website[13]: Ionosphere, Iris and Wine were used to compare the accuracy of each algorithm. The relevant parameters of the experimental data set are shown in table 1.

| Data set  | Number of rows | Number of attributes | Number of clusters |
|-----------|----------------|----------------------|-------------------|
| Ionosphere| 351            | 34                   | 2                 |
| Iris      | 150            | 4                    | 3                 |
| Wine      | 178            | 13                   | 3                 |

(2) In Experiment 2, the concurrent execution performance of the algorithm is tested. The algorithm uses Ionosphere data set, and the performance experiments are carried out according to the data volume of 500000 rows (0.31 GB), 1 million rows (0.62 GB), 2 million rows (1.25 GB), and 4 million rows (2.5GB).
5.2. Analysis of experimental results

(1) In Experiment 1, 10 experiments were conducted with Ionoshpere, iris and wine datasets on UCI. The average accuracy and average iteration times of these 10 experiments are taken as the experimental result data, as shown in Table 2.

Table 2. Comparison of accuracy and clustering iteration times of each algorithm.

| Data set | K-means | DK-means | MDDK-means |
|----------|---------|----------|------------|
|          | Ionoshpere | Iris | Wine | Ionoshpere | Iris | Wine | Ionoshpere | Iris | Wine |
| Accuracy | 66.9% | 83.6% | 89.5% | 69.8% | 88.0% | 95.5% | 70.9% | 88.7% | 96.1% |
| Number of iterations | 8.6 | 6.9 | 8.3 | 9 | 5 | 8 | 6 | 4 | 7 |

From the result data, we can see that the accuracy of MDDK-means is higher than that of traditional K-means and DK-means. In addition, MDDK-means has a smaller number of iterations, that is, it has faster convergence speed and higher execution efficiency. Therefore, the proposed MDDK-means is superior to the traditional K-means and DK-means in clustering accuracy and execution efficiency.

(2) In Experiment 2, MDDK-means uses four data sets of different sizes to cluster on spark platform with 1-4 nodes. To speed up the performance of this algorithm. The curve of the experimental data is shown in Fig.4.

6. Conclusion

In this paper, MDDK-means is proposed, which introduces the cluster compact density and the distance between clusters into the initial cluster center selection based on the density, which ensures that each initial cluster center has greater density, larger distance between clusters and greater cluster compact density, and improves the quality of initial cluster center selection. At the same time, MDDK-means is parallelized with spark parallel computing framework, which can be applied to big data scenarios. Experimental results show that the proposed algorithm has good accuracy, fewer iterations and higher parallel execution performance.

Acknowledgments

Thanks to the China Railway Eryuan Engineering Group Co.,Ltd for providing research project support, as well as colleagues from the Academy of Science and Technology for their encouragement and help.
References
[1] Altman, N., Krzywinski, M. (2017) Points of Significance: Clustering. J. Nature Methods., 14(6):545-546.
[2] Wu, M., Li, X., Liu, C., et al. (2018) Robust global motion estimation for video security based on improved k-means clustering. J. Journal of Ambient Intelligence & Humanized Computing., 10(2): 439-448.
[3] Lorbeer, B., Kosareva, A., Deva, B., et al. (2018) Variations on the Clustering Algorithm BIRCH. J. Big Data Research., 11:44-53.
[4] Zhang, P., Wang, Y., Liang, L., et al. (2020) Short-Term Wind Power Prediction Using GA-BP Neural Network Based on DBSCAN Algorithm Outlier Identification. J. Processes., 8(2):157.
[5] Bureva, V., Sotirova, E., Popov, S., et al. (2017) Generalized Net of Cluster Analysis Process Using STING: A Statistical Information Grid Approach to Spatial Data Mining. In:International Conference on Flexible Query Answering Systems. Univ Westminster, London, ENGLAND. 239-248
[6] Zhang, Y., Wang, J., Han, D., et al. (2017) Fuzzy-Logic Based Distributed Energy-Efficient Clustering Algorithm for Wireless Sensor Networks. J. Sensors., 17(7):1554.
[7] Xie, X., Li X., Mo L. (2018) Microblog public opinion analysis based on improved k-means algorithm. J. Computer engineering and science., 40 (01): 155-158.
[8] Judith, J.E., Jayakumari, J. (2017) Distributed Document Clustering Analysis Based on a Hybrid Method. J. China Communications., 14(02):131-142.
[9] Wang, B., Yu, X. (2018) Parallel K-means clustering algorithm for adaptive cuckoo search. J. Computer application research., 35 (3): 675-679.
[10] Tang, Z., Liu, K., Xiao, J., et al. (2017) A parallel k-means clustering algorithm based on redundancy elimination and extreme points optimization employing MapReduce. J. Concurrency and Computation., 29(20):e4109.1-e4109.18.
[11] Fan, W., Han, Z., Wang, R. (2018) An Evaluation Model and Benchmark for Parallel Computing Frameworks J. Mobile Information Systems., 2018:1-14.
[12] Zhang, X., Khanal, U., Zhao, X., Ficklin, S. (2018) Making Sense of Performance in In-memory Computing Frameworks for Scientific Data Analysis: A Case Study of the Spark System. J. Journal of Parallel and Distributed Computing., 120: 369-382.
[13] UCI.(2015-03-30) UCI Machine learning repository [DB/OL].http://archive.ics.uci.edu/ml/.