 TSP, vol.1, no.1, pp.1-5, 2018

SMK-means: An Improved Mini Batch K-means Algorithm Based on Mapreduce with Big Data

Bo Xiao[1], Zhen Wang[2], Qi Liu[3, *] and Xiaodong Liu[3]

Abstract: In recent years, the rapid development of big data technology has also been favored by more and more scholars. Massive data storage and calculation problems have also been solved. At the same time, outlier detection problems in mass data have also come along with it. Therefore, more research work has been devoted to the problem of outlier detection in big data. However, the existing available methods have high computation time, the improved algorithm of outlier detection is presented, which has higher performance to detect outlier. In this paper, an improved algorithm is proposed. The SMK-means is a fusion algorithm which is achieved by Mini Batch *K*-means based on simulated annealing algorithm for anomalous detection of massive household electricity data, which can give the number of clusters and reduce the number of iterations and improve the accuracy of clustering. In this paper, several experiments are performed to compare and analyze multiple performances of the algorithm. Through analysis, we know that the proposed algorithm is superior to the existing algorithms.

Keywords: Big data, outlier detection, SMK-means, Mini Batch *K*-means, simulated annealing.

1. Introduction

Nowadays, electric energy has become the main part of energy utilization. With the rapid growth of China's economy, social electricity consumption has also increased year by year, but the supply of power energy still cannot fully meet the needs of economic development. How to improve the efficiency of power energy utilization and explore the potential of power energy utilization is the most important part of the data mining in the power energy. In our country, because of many reasons, such as the obsolescence of public electricity or household electric equipment, and the weak consciousness of energy

[1] School of Environmental Science and Engineering, Nanjing University of Information Science and Technology, 219 Ningliu Road, Nanjing Jiangsu, 210044, China.

[2] School of Computer and Software, Nanjing University of Information Science and Technology, 219 Ningliu Road, Nanjing Jiangsu, 210044, China.

[3] School of Computing, Edinburgh Napier University, 10 Colinton Road, Edinburgh, EH10 5DT, UK

[*]Corresponding Author: Qi Liu. Email: q.liu@napier.ac.uk.

saving of consumers, the power energy has not been fully utilized. In addition, according to the statistics of the State Grid, it has been found that the theft of electricity in recent years has resulted in the loss of tens of millions of dollars, and the way of stealing electricity from the original barbaric violent electricity theft has developed into a more specialized means, more covert behavior and a large-scale through the use of intelligent equipment. The way of implementation of these abnormal power consumption will bring a lot of economic losses to business users and family users [Gurusamy and Subramaniam (2015)].

Therefore, in or der to use the power energy more effectively and protect the rights and interests of the consumers of the power users, the abnormal analysis of the household electricity data will be effectively realized, and the corresponding measures are taken to reduce the waste of the energy consumption according to the identified anomalies, and the abnormal recognition operation of the electrical data for the family users is carried out. The application of data mining to the abnormal recognition of household electricity consumption will save a lot of unnecessary losses, and it will also play a role in promoting the management of the State Grid. With the increasing amount of electricity used by home users, a large number of data are produced. The cloud platform provides a distributed storage system for storing massive data. By mining and analyzing the data, we detect the anomalies of the data set, and then identify the abnormality in the process of the user's electricity use.

Cloud computing and data mining have received extensive attention at home and abroad. Because of the increase in the demand for electricity and the consumption of electricity, the power data also increases rapidly. Therefore, the excavation of power data will also consume more computing resources [Shanmugam (2017); Yildiz (2015)]. Many distributed computing frameworks in cloud computing, such as Hadoop MapReduce, Apache Spark, Apache Flink etc., have obvious advantages for the calculation of real time or offline massive data, and the cloud computing is fault tolerant, and is based on the high YARN cluster management. Availability is to prevent the node from downtime. Therefore, cloud computing has a huge role in the excavation of power data in [Chen (2017); Gurusamy (2015)].

With the rapid development of computer technology and Internet technology, cloud computing technology has also emerged. With the powerful computing power brought by its distributed platform, cloud computing has ushered in a completely new computing experience for the processing of massive data. Data mining will release more potential under the cloud environment. This combination not only provides us with distributed storage and sharing of distributed file system (Hadoop Distributed File System, HDFS) files, but also provides fast distributed computing. Efficient data processing [Kumari, Kapoor and Singh (2016)]. In addition, there are many distributed computing frameworks for processing massive data, such as Hadoop MapReduce, Spark, Strom, Flink, etc. For processing in distributed computing, a job is divided into many tasks, each task consists of one or multiple computer nodes perform calculations, which are highly efficient for the processing and calculation of offline mass and real time data. Distributed computing has the advantages of resource sharing and load balancing, which can not only reduce the computing burden of the server, but also reduce the burden on the server. MapReduce

uses "Map" and "Reduce" to handle the distributed processing of large data sets.

The Mini Batch *K*-means algorithm has an impact on the clustering effect. SMK-means is proposed, which is an optimization of the Mini Batch *K*-means algorithm. It is not only suitable for processing massive data, but also improves clustering. The effect is to avoid the algorithm being trapped in a local optimal solution. SMK-means algorithm is mainly divided into the following steps, the first step is to bring about the simulated annealing algorithm for reducing the number of iterations and the second step is to utilize parallelization Mini Batch *K*-means algorithm on Hadoop and the third step is to calculate outlier scores by distance function.

In this paper, the contribution of the improved algorithm is shown as follows:

- A novel algorithm is proposed for anomaly detection.
- The improved algorithm combines the probability based algorithm and clustering algorithm. That is, the Mini Batch *K*-means algorithm based on simulated annealing algorithm.
- Based on cloud platform, an improved anomaly detection algorithm is implemented, and SMK-means algorithm is parallelized and distributed.

The rest of this paper is structured as follows: Section II summarizes Research status of outlier detection. The presented method in this paper is introduced in details in Section III. In Section IV, experimental environment, algorithm implementation and related performance analysis and comparison, the experimental studies and evaluation of methods are reported, while conclusion and future work are covered in Section V.

2. Related Work

With the advent of the third scientific and technological revolution, electronic technologies, atomic energy technologies, and bioengineering technologies have also continued to develop. As a result, the demand for electrical energy has become increasingly strong. To date, electrical energy is still an indispensable energy source in human life. At the same time, domestic and foreign research on electric energy has generated a great deal of interest. Whether it is industrial electricity, domestic electricity, or biological electricity, there are many research outputs [Yan (2015); Li (2013)]. In this section，we describe the research status of household electricity outlier detection and introduced relevant research work and introduce the research of clustering algorithm for outlier detection mainly, the already proposed algorithm may have more high computational complexity or less computation time, but these methods do not have better performance for high dimensional dataset and cannot reduce calculation time while ensuring accuracy.

To avoid anomalies such as California's power crisis of 2000 and 2001, the authors attempted to predict abnormalities using advanced machine learning algorithms, particularly the Electricity Price Change Point Detection (CPD) algorithm during the California power crisis. In order to solve the expensive calculation of a large amount of data at the time of application, the one-dimensional time series data Gaussian process

(GP) is accelerated. This algorithm effectively makes it possible to use the hourly price data to calculate change points during the California power crisis [Gu, Choi, Gu et al. (2013)].

A novel self-adaptive data shifting based method for one-class SVM (OCSVM) hyper-parameter selection, is proposed to generate a controllable number of high-quality pseudo outlier data around target data by efficient edge pattern detection and a "negative shifting" mechanism, which can effectively regulate the OCSVM decision boundary for an accurate target data description.

A large scale network traffic monitoring and analysis system based on Hadoop in 2014 was proposed, which is an open source distributed computing platform for commodity hardware big data processing. The system has been deployed in the core network of a large cellular network and has been widely evaluated. The results show that the system can effectively handle 4.2 TB of data from the 123 Gb/s link each day with high performance and low cost [Liu, Liu and Ansari (2014)].

A new incremental and distributed classification based on the popular nearest neighbor algorithm was proposed. This method is implemented in Apache Spark and includes distributed metric space ordering to perform faster searches. In addition, an efficient incremental data instance selection method has been proposed for continuous update of large scale data streams and the elimination of outdated examples from case libraries. This alleviates the high computational requirements of the original classifier, making it suitable for the problem under consideration. Experimental studies conducted on a set of real mass data streams demonstrate the effectiveness of the proposed solution [Ramírez-Gallego, Krawczyk, García et al. (2017)].

In [Song, Rochas, Beze et al. (2016)], the authors compare the different algorithms with the KNN algorithm based on MapReduce, and evaluate them through the combination of theory and time. To be able to compare solutions, we identified three general steps for KNN computation on MapReduce: data preprocessing, data partitioning, and calculations. Analyze each step from the aspects of load balancing, accuracy, and complexity. Various data sets were used in the experiment. The influences of data volume, data dimension and k value were analyzed from multiple angles of time and space complexity and precision. The experimental part brings new advantages and disadvantages to each algorithm.

In this paper, we employ SMK-means algorithm, which combines this algorithm with the simulated annealing algorithm. Meanwhile, SMK-means optimize the objective function, not only reduces the calculation time but also improves the accuracy of the algorithm.

3. The Proposed Method

In this section, we mainly introduce the method proposed in this paper. The first part introduces data preprocessing and feature engineering. The related concepts elaborated are described in details in second part. The SMK-means algorithm is explained in detail in third part.

3.1. Data Preprocessing

Feature engineering： When the data preprocessing is completed, it is necessary to select a meaningful feature input to the selected model for training modeling [Panigrahy, Santra and Chattopadhyay (2017)]. In general, select features from two perspectives:

- Whether the characteristic is divergence: If a characteristic is not divergent, for example, the variance is close to 0, that is, the sample is basically not different in this feature, which is not useful for the discrimination of the sample.
- Relevance of features and objectives: This is more obvious, and features that are highly relevant to the target should be preferred

A preliminary introduction has been made to whether or not the existing features are selected. Next, because the data features are limited and cluster related algorithms require more data features, it is suitable for clustering data sets. Therefore, before modeling the data, according to the characteristics of the model, the characteristics of the time characteristics are constructed according to the time characteristics of the data set. This is also the discretization of the data. For example, whether or not weekends, workdays, seasons, time periods, and the like are related to the power consumption of home users. Built up data features such as Tab. 1.

Table 1: Data characteristics table

Field Name	Represents
Id	Data number
Date	Date
Time	Time
Sub_metering_1	Sub-meter1
Sub_metering_2	Sub-meter2
Sub_metering_3	Sub-meter3
apparent_power	Power
Weekend_No	Current time is not weekend
Weekend_Yes	Current time is the weekend
Season_0	Spring
Season_1	Summer
Season_2	Autumn
Season_3	Winter
Tfd_0	0-4 hour
Tfd_1	4-8 hour
Tfd_2	8-12 hour
Tfd_3	12-16 hour
Tfd_4	16-20 hour
Tfd_5	20-24 hour

Data Preprocessing： Due to the difference of data indexes and data magnitudes of

different attributes, a plurality of data indexes or multiple data levels appear in one data set, so it is necessary to standardize the operation of the data set. The main way to achieve this is to scale the data according to a certain proportion of the EU, so that it falls into a range of characteristics. Especially when comparing or evaluating indicators, the unit attributes of the data need to be weakened to convert the data into dimensionless pure data values. In order to compare and weight the data indicators of different units or different orders of magnitude. We handle original dataset to accommodate our proposed method before we apply it. By intercepting partial sample data from original data set, removing the noise and employing normalized processing of sample data set. In this paper, we use standard deviation to process sample data. Standard deviation, the processed data is in accord with the standard normal distribution. Standardized data set as shown in Fig. 1.

Id	Date	Time	b_meteringg_meteringg_meteringparent_powWeekend_NoWeekend_Ye	Season_0	Season_1	Season_2	Season_3	Tfd_0	Tfd_1	Tfd_2	Tfd_3	Tfd_4	Tfd_5
0	1/1/2008	0:00:00	-0.31359 -0.296508 1.6611325 0.5551121 1 0	1	0	0	0	1	0	0	0	0	0
1	1/1/2008	1:00:00	-0.31359 -0.190266 1.6471356 0.5835702 1 0	1	0	0	0	1	0	0	0	0	0
2	1/1/2008	2:00:00	-0.31359 -0.296508 1.6844607 1.5367044 1 0	1	0	0	0	1	0	0	0	0	0
3	1/1/2008	3:00:00	-0.31359 -0.186331 1.668131 1.8058327 1 0	1	0	0	0	1	0	0	0	0	0
4	1/1/2008	4:00:00	-0.31359 -0.296508 1.6494684 1.4652053 1 0	1	0	0	0	1	0	0	0	0	0
5	1/1/2008	5:00:00	-0.31359 -0.288638 1.6401372 1.5147638 1 0	1	0	0	0	1	0	0	0	0	0
6	1/1/2008	6:00:00	-0.31359 -0.1942 -0.048824 0.6393894 1 0	1	0	0	0	1	0	0	0	0	0
7	1/1/2008	7:00:00	-0.31359 -0.296508 -0.844316 0.2475693 1 0	1	0	0	0	1	0	0	0	0	0
8	1/1/2008	8:00:00	-0.31359 -0.186331 -0.844316 0.3010773 1 0	1	0	0	0	1	0	0	0	0	0
9	1/1/2008	9:00:00	-0.31359 -0.296508 -0.844316 -0.04381 1 0	1	0	0	0	1	0	0	0	0	0
10	1/1/2008	10:00:00	-0.31359 -0.268963 -0.844316 -0.514417 1 0	1	0	0	0	1	0	0	0	0	0
11	1/1/2008	11:00:00	2.239453 1.3482761 1.0569322 1.7490841 1 0	1	0	0	0	1	0	0	0	0	0
12	1/1/2008	12:00:00	2.4278695 -0.174526 0.4737273 1.0952387 1 0	1	0	0	0	1	0	0	0	0	0
13	1/1/2008	13:00:00	-0.31359 6.6249628 -0.573709 2.1249578 1 0	1	0	0	0	1	0	0	0	0	0
14	1/1/2008	14:00:00	-0.31359 0.750173 1.693792 0.8103077 1 0	1	0	0	0	1	0	0	0	0	0
15	1/1/2008	15:00:00	-0.31359 -0.04861 0.5717057 -0.031433 1 0	1	0	0	0	1	0	0	0	0	0
16	1/1/2008	16:00:00	-0.31359 -0.296508 -0.844316 -0.269341 1 0	1	0	0	0	1	0	0	0	0	0
17	1/1/2008	17:00:00	-0.31359 -0.186331 -0.844316 1.0661673 1 0	1	0	0	0	1	0	0	0	0	0
18	1/1/2008	18:00:00	-0.31359 -0.296508 -0.844316 1.9813467 1 0	1	0	0	0	1	0	0	0	0	0
19	1/1/2008	19:00:00	-0.31359 -0.229615 0.2287813 2.138001 1 0	1	0	0	0	1	0	0	0	0	0
20	1/1/2008	20:00:00	-0.31359 -0.253224 1.5794839 1.9733994 1 0	1	0	0	0	1	0	0	0	0	0
21	1/1/2008	21:00:00	2.9318835 -0.296508 1.2248953 1.7995752 1 0	1	0	0	0	1	0	0	0	0	0
22	1/1/2008	22:00:00	2.7387566 -0.186331 -0.844316 0.2763377 1 0	1	0	0	0	1	0	0	0	0	0
23	1/1/2008	23:00:00	-0.31359 -0.296508 -0.844316 -0.756142 1 0	1	0	0	0	1	0	0	0	0	0
24	2/1/2008	0:00:00	-0.31359 -0.186331 -0.844316 -0.764495 1 0	1	0	0	0	1	0	0	0	0	0
25	2/1/2008	1:00:00	-0.31359 -0.296508 -0.844316 -0.791396 1 0	1	0	0	0	1	0	0	0	0	0
26	2/1/2008	2:00:00	-0.31359 -0.296508 -0.844316 -0.832693 1 0	1	0	0	0	1	0	0	0	0	0
27	2/1/2008	3:00:00	-0.31359 -0.186331 -0.844316 -0.767736 1 0	1	0	0	0	1	0	0	0	0	0
28	2/1/2008	4:00:00	-0.31359 -0.296508 0.3710833 -0.234303 1 0	1	0	0	0	1	0	0	0	0	0
29	2/1/2008	5:00:00	-0.31359 -0.190266 -0.844316 -0.775978 1 0	1	0	0	0	1	0	0	0	0	0
30	2/1/2008	6:00:00	-0.31359 -0.296508 -0.844316 0.0696449 1 0	1	0	0	0	1	0	0	0	0	0

Figure 1: Standardized data set

3.2. Basic clustering algorithm

The idea of the basic *K*-Means algorithm is simple. The constant *k* is determined in advance. The constant *k* means the number of the final clusters. The initial point is randomly selected as the centroid and the similarity between each sample and the centroid is calculated (Euclidean distance), the sample points are assigned to the most similar class, and then the centroid of each class is recalculated (the class center). This process is repeated until the centroid is not changed, and finally each sample is determined. The category they belong to and the centroid of each class. Since the similarity between all samples and each centroid is calculated every time, the convergence speed of *K*-Means algorithm is slow on large scale data sets [Yang (2017); Joshi (2017)].

Mini Batch *K*-Means algorithm is used as a variant of the standard *K*-Means clustering

algorithm [Cho (2014); Newling (2016); Feizollah (2015)]. Through the idea of "divide and conquer", the data is logically divided into multiple small batch data subsets. In other words, the algorithm does not need to perform calculation on all data samples in the calculation process, but instead randomly extracts subsets of data each time the algorithm is trained. This can greatly reduce the computation time for data. At the same time, Mini Batch *K*-Means also tries to optimize the objective function. The objective function is as follows:

$$SSE = \sum_{i=1}^{K} \sum_{j \in C_m} dis(c_i, j)^2 \tag{1}$$

k represents *k* clustering centers, c_i represents the *i* center, *j* represents the sample points, and *dist* represents Euclidean distance. By calculating the Euclidean distance, the optimization function is calculated, which is the sum of squared errors (even if Sum of the Squared Error, SSE). We compare the principles of the above two algorithms as shown in Fig. 2.

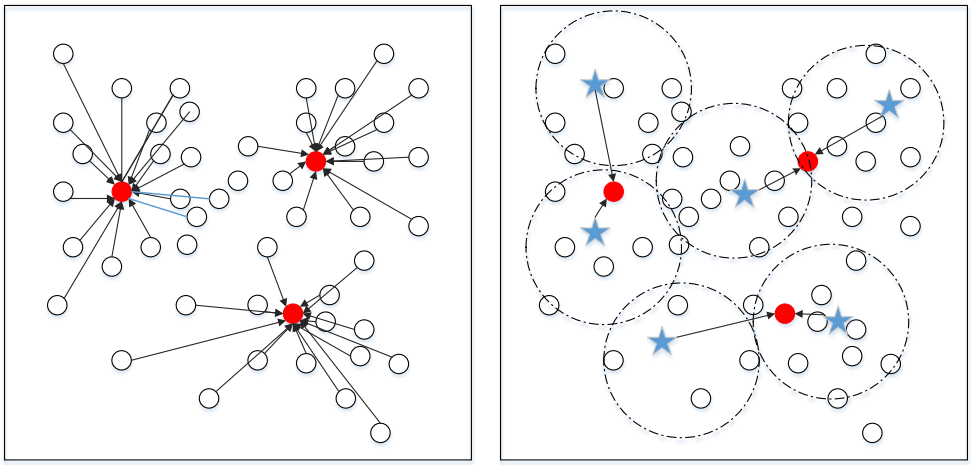

Figure 2: Two different clustering algorithm calculations: (a) *K*-means, (b) Mini Batch *K*-means

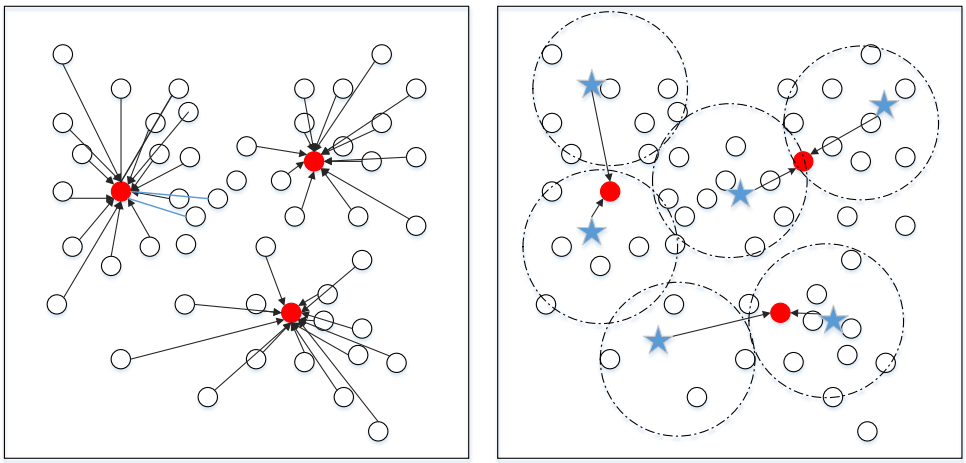

Figure 3: Two different clustering algorithms: (a) *K*-means algorithm; (b) Mini Batch *K*-means.

Compared with other related algorithms, Mini Batch *K*-Means reduces the convergence time of *K*-means. The effect of this algorithm is slightly worse than the standard algorithm. However, the Mini Batch *K*-Means algorithm implements further clustering by performing corresponding mathematical statistics on small batches of data in advance. That is to say, the standard clustering algorithm is that the center point is updated in a single point. As shown in Fig. 3, the differences between the two algorithms are shown from the intuitive and calculation methods respectively, so that the knowledge of the Mini Batch *K*-Means ratio is obtained. Clustering algorithms have faster convergence speeds and are more suitable for processing massive data.

Mini Batch *K*-means is mainly divided into the following steps:

(1) divide data sets into multiple batches randomly, and regard small batch as a whole.

(2) set the number of initial cluster clusters.

(3) allocate small batch data to the nearest cluster center.

(4) update the cluster center iteratively until the cluster center doesn't change any more.

Compared with the *K*-means algorithm, the Mini Batch *K*-means algorithm is based on every small sample set instead of a single data point. The operation object is also a small batch of data, and for each small batch of data, the update clustering center is realized by calculating the average value, and the small batch of data is allocated to the new cluster center. With the gradual increase of the number of iterations, the cluster center is gradually stable, until the cluster center no longer changes. Then the calculation is stopped.

Mini Batch *K*-means algorithm algorithm is simple, easy to understand and implement, the effect of clustering and the speed of the standard clustering algorithm are few, and the concept of batch type is introduced to speed up the data clustering speed, and the operation time can be reduced on the premise of maintaining the accuracy of the data. Because it is a batch processing data set, it does not need to calculate all the data samples

in the calculation process, and extracts some samples from different categories of data to calculate. The amount of data needed to be calculated is reduced much, so the running time will be reduced accordingly, so the performance will be shown when the data is large. It is superior to the standard clustering algorithm.

3.3 SMK-means algorithm

This section makes corresponding improvements to the shortcomings of the Mini Batch *K*-means algorithm. When referring to Mini Batch *K*-means, we must mention the *K*-means algorithm. The Mini batch *K*-means itself is an optimization algorithm of the *K*-means algorithm, which largely retains the advantages of the standard *K*-means algorithm. Improved the shortcomings of the long computation time of the *K*-means algorithm. Although the Mini Batch *K*-means is suitable for the calculation of massive data sets, the calculation time is greatly reduced, and the k-value convergence is accelerated. However, because this algorithm is a small-batch calculation, it reduces the accuracy of the algorithm. In order to improve the accuracy of the algorithm, it also improves the accuracy of detection of power anomalies for home users. The SMK-means algorithm draws on the idea of the simulated annealing algorithm. The simulated annealing algorithm is a stochastic algorithm and does not necessarily find the global optimal solution. It can quickly find the approximate optimal solution to the problem, combined with the Mini Batch *K*-means. The advantage of not only can improve the accuracy of clustering, but also greatly reduce the predicament trapped in the local optimum. In addition, the algorithm is deployed based on the cloud environment, so it is more advantageous for processing large amounts of data. Therefore, the SMK-means clustering algorithm is proposed. In order to better understand the core idea of the SMK-means algorithm, some detailed explanations will be given.

The implementation of the algorithm based on the cloud environment, the computing framework adopted in this paper is MapReduce, which implements parallelization and distributed computing through MapReduce. In Hadoop, the MapReduce computing framework takes each task as a Job. Each Job is divided into two execution stages, the Map execution stage and the Reduce execution stage. The Map function interface implements data filtering and distribution. The Reduce function interface implements the consolidation of the results on the Map side. The Combine process is also nested in the middle. Combine links Map and Reduce. It is distributed in the Map and Reduce stages of MapReduce. Each Map generates a large amount of output. Combine The role is to first do a merge on the Map side output to reduce the amount of data transferred to Reduce, Combine the most basic function is to achieve the integration of the local key to reduce computation time. The output of Combine is the input of Reduce. In fact, Combine is also a special Reduce. MapReduce splits the data set stored on the HDFS distributed file system and splits it into the Map stage to map the split slice data set through Combine. Realize the initial merge function of data, and then enter the Reduce stage to complete the final merge of data.

Fig. 4 shows the parallelization of the SMK-means algorithm. The two dashed lines in the figure are the two parallel parts of the SMK-means algorithm. The parallelization of

the simulated annealing algorithm implements the initial center point of the clustering. The selection also prevents clustering from falling into the local optimal solution; the Mini Batch *K*-means parallelization implementation is based on the simulated annealing algorithm, which eliminates the steps of iteratively finding cluster midpoints and the data batching, greatly improving the computation of the algorithm.

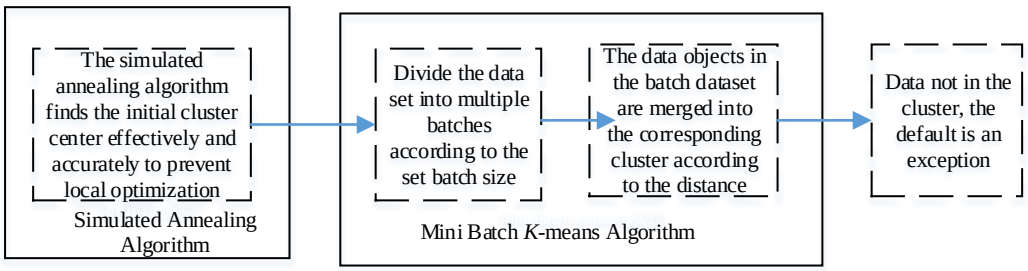

Figure 4: Parallel implementation of SMK-means algorithm

4. Experiments and Evaluation

4.1. Experiments Analysis

The implementation of the distributed SMK-means clustering algorithm based on cloud environment is mainly the design and implementation of the Map and Reduce functions. The SMK-means algorithm based on the MapReduce computing framework can be divided into multiple subtasks, namely selecting *k*-values and Mini Batch *K*-means algorithm. Choosing the *k* value is to combine the advantages of the simulated annealing algorithm to select the number of clusters in advance and effectively avoid falling into the local optimum solution. The main task of the Mini Batch *K*-means algorithm is to calculate the distance between the data object and the cluster center. The abnormal points can be selected by clustering and the outlier scores can be calculated for these abnormal points. These two parts are all realized through distributed parallel computing. Whether it is *k*-value or Mini Batch *K*-means algorithm, the distance calculation between data object and cluster center is independent and does not affect each other. Hadoop's MapReduce computing framework completes the functions of these two parts. The simulated annealing algorithm can predetermine the number of initial clusters. This is also a value needed by the Mini Batch *K*-means algorithm, so reducing the number of iterations of the algorithm.

The experimental environment of this paper is implemented on Hadoop cluster with 9 nodes, including one master node and 8 slave nodes. Fig. 5 describes the layout of the experimental environment in this paper.

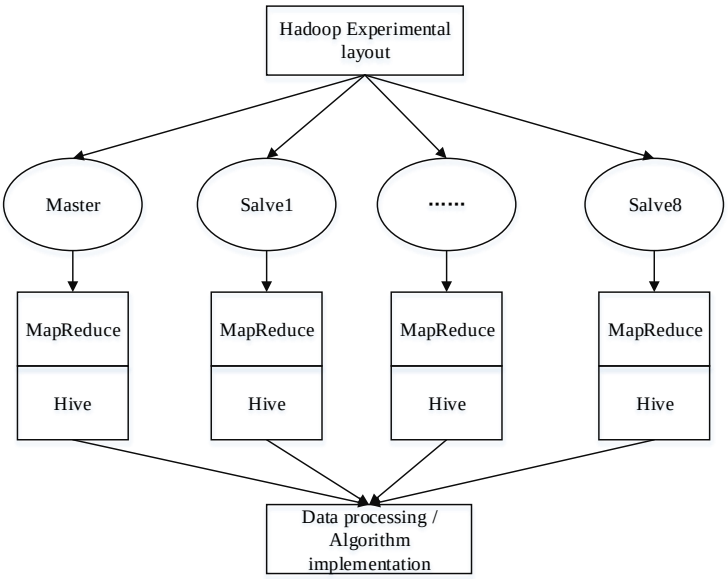

Figure 5: Experimental environment layout

The implementation of the MapReduce parallelization of the Mini Batch *K*-means algorithm is basically the same as that of the parallel algorithm of the standard algorithm. This paper is based on the simulated annealing algorithm to determine the cluster center in advance, reduce the number of iterations, and store it on the HDFS. The data is divided into multiple batches, and the batch is split to enter the Map stage. The preliminary data mapping is completed. The data is reduced locally through the Combiner process, and the Reduce process is finally completed. The resulting intermediate results are stored in the In HDFS, the MapReduce parallelization of the Mini Batch *K*-means algorithm is shown in Fig. 6.

4.2 Performance Evaluation

This section parallelizes the algorithm we propose. To test the accuracy and algorithm runtime of different algorithms, we use *K*-means algorithm, Mini Batch *K*-means algorithm, and SMK-means algorithms. The three algorithms perform parallel experiments. This experiment was performed on 9 Hadoop cluster. There are 1 master node and 8 slave nodes.

- Comparison of the accuracy of the algorithm

The contour coefficient is an evaluation index to measure the quality of the clustering algorithm. Assume that the data is clustered. For a sample point x in a cluster, the formula for calculating the contour factor is as follows:

$$S(\mathrm{x}) = \frac{m(\mathrm{i}) - \mathrm{a}(\mathrm{i})}{Max\{\mathrm{m}(\mathrm{i}), \mathrm{a}(\mathrm{i})\}} \tag{2}$$

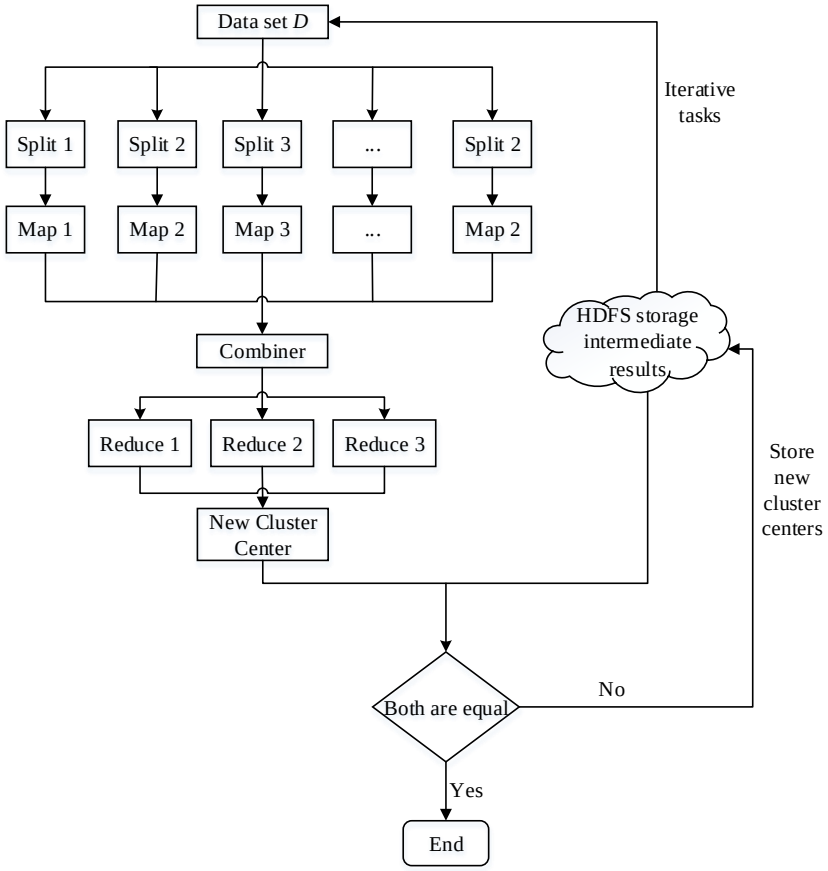

Figure 6: MapReduce parallelization flow chart of SMK-means algorithm

Where a(*i*) = average (the distance of sample x to all other points in the cluster it belongs to), find the average of the distance of sample x from all other points in the cluster; m(i) = min (sample x to all The average distance of the points other than the cluster in which it is located) finds the minimum value of the distance from the sample *x* to the non-cluster midpoint. It can be seen that the range of the contour coefficient is [-1, 1]. The closer the value is to 1, the higher the degree of cohesion in the cluster is, and the higher the degree of separation between clusters and clusters. The comparison of the accuracy of the algorithm as shown in Fig. 7.

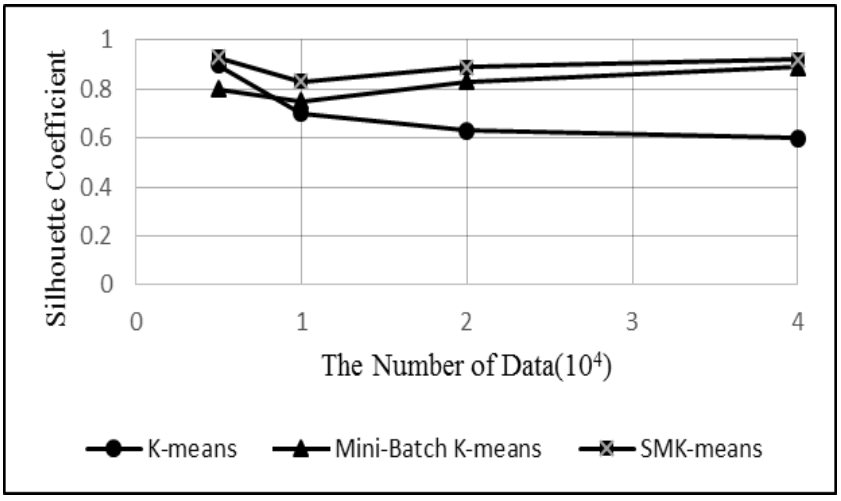

Figure 7: Comparison of the accuracy of the algorithm

- Algorithm runtime comparison

Three related algorithms are run for different amounts of data, statistics of their running time and drawing comparison are shown in Fig. 8.

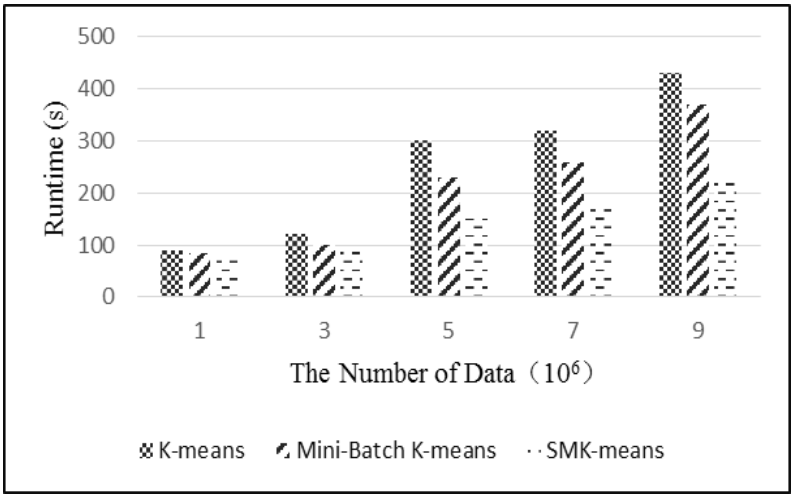

Figure 8: Algorithm runtime comparison

From Fig. 8, it can be seen that when running the same amount of data, the *K*-means algorithm runs more time than the other two algorithms, and as the amount of data increases, the *K*-means algorithm runs The gap between time and the running time of the other two algorithms is also gradually widening; because the Mini Batch *K*-means algorithm is used to process data in batches, the computation time is also due to the *K*-means algorithm, and the SMK-means algorithm is predetermined by the initial The number of cluster centers reduces the number of iterations and also reduces the computing time.

- Algorithm accuracy comparison

The accuracy rate is aimed at the results. How many positive samples are found in the positive samples and two positive samples, one of which is positive sample (TP) and the other is a negative sample calculated as a positive sample (FP) and a formula for calculating the accuracy rate as shown in below:

$$precision = \frac{TP}{TP + FP} \tag{3}$$

By marking the same data set, we mark different outliers to compare the accuracy of the three algorithms.

Table 2: Algorithm precision comparison

The number of outlier	*K*-means	Mini Batch *K*-means	SMK-means
200	0.942	0.864	0.953
400	0.917	0.873	0.932
600	0.876	0.901	0.927
800	0.903	0.881	0.907

From Tab. 2, the precision rate comparison of the algorithm, the precision rate of the *K*-means algorithm is still better, but the whole is not stable enough, but compared to the Mini Batch *K*-means algorithm, the *K*-means algorithm has many advantages. Compared with the other two algorithms, SMK-means is more stable in computing abnormal data, and the overall efficiency is better than that of *K*-means and Mini Batch *K*-means algorithm.

Meanwhile, the complexity of algorithm is a yardstick for computing data and running speed, the proposed algorithm is not only based on the large data platform, but also uses a hierarchical down sampling data, so it supports the calculation of mass data. Therefore, the complexity of SMK-means algorithm is lower than that of the traditional clustering algorithm.

5. Conclusion

Based on the implementation of the parallel implementation of the algorithm SMK-means based on Hadoop implementation, the pretreatment of the data set is first introduced, the processing of missing values, the construction of feature engineering, and the data standardization operations are performed. In the second part, the overall idea of the algorithm and the realization process of parallelization are introduced. The number of cluster centers is initialized based on the simulated annealing algorithm and it is realized by parallelization based on MapReduce. Then the parallelization of the Mini Batch *K*-means algorithm is implemented. The parallelization of the two steps is accomplished by the Map operation and the Reduce operation. The implementation process of the Map function and the Reduce function is introduced in detail, and the corresponding pseudo code is provided. Finally, by comparing the SMK-means algorithm with the *K*-means and Mini Batch *K*-means algorithms for accuracy, precision, and runtime, a number of performance indicators are shown to analyze the SMK-means algorithm from accuracy. Better than other algorithms, SMK-means is more stable in terms of accuracy, and its

runtime is shorter. In summary, the SMK-means algorithm is not only suitable for the processing of massive data, but also can guarantee the accuracy of the algorithm. For the outlier detection of data, it also has a relatively stable precision.

Acknowledgement: This work is supported by Major Program of the National Social Science Fund of China (Grant No. 17ZDA092) and Marie Curie Fellowship (701697-CAR-MSCA-IF-EF-ST), and the PAPD fund.

References

Chen, J.; Li, K.; Tang, Z.; Bilal, K.; Yu, S.; Weng, C. (2017): A parallel random forest algorithm for big data in a spark cloud computing environment. *IEEE Transactions on Parallel & Distributed Systems*, vol. 28, no. 4, pp. 919-933.

Cho, H.; An, M. K. (2014): Co-clustering algorithm: batch, mini-batch, and online. *International Journal of Information & Electronics Engineering*, vol. 4, no. 5, pp. 340-346.

Feizollah, A.; Anuar, N. B.; Salleh, R.; Amalina, F. (2015): Comparative study of k-means and mini batch k-means clustering algorithms in android malware detection using network traffic analysis. *International Symposium on Biometrics and Security Technologies*, pp. 193-197.

Gu, W.; Choi, J.; Gu, M.; Simon, H. (2013): Fast Change Point Detection for electricity market analysis. *IEEE International Conference on Big Data*, pp. 50-57.

Gurusamy, R.; Subramaniam, V. (2015): A machine learning approach for MRI brain tumor classification. *Computers Materials & Continua*, vol. 53, no. 2, pp. 91-109.

Joshi, A.; Sabitha, A. S.; Choudhury, T. (2017): Crime analysis using K-Means clustering. *International Conference on Computational Intelligence and Networks (CINE)*, Odisha, India, pp. 33-39.

Kumari, S.; Kapoor, R. K.; Singh, S. (2016): A survey on different techniques for information resource scheduling in cloud computing. *International Conference on Computational Intelligence and Communication Networks*, pp. 736-740.

Li, Q.; Wang, H. (2013): Analysis on Combinations Type and Interstage Energy Match Research of Electrical Power Generating System SPD. Meteorology Journal of Inner Mongolia, vol. 23, no. 5, pp. 35-37.

Liu, J.; Liu, F.; Ansari, N. (2014): Monitoring and analyzing big traffic data of a large-scale cellular network with hadoop. *Network IEEE*, vol. 28, no. 4, pp. 32-39.

Newling, J.; Fleuret, F. (2016): Nested mini-batch k-means. *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, Barcelona, Spain.

Panigrahy, P. S.; Santra, D.; Chattopadhyay, P. (2017): Feature engineering in fault diagnosis of induction motor. *International Conference on Condition Assessment Techniques in Electrical Systems*, pp. 306-310.

Ramírez-Gallego, S.; Krawczyk, B.; García, S.; Woźniak, M.; Benítez, J. M.; Herrera, F. (2017): Nearest neighbor classification for high-speed big data streams using spark. *IEEE Transactions on Systems Man & Cybernetics Systems*, pp. 1-13.

Shanmugam, M.; Singh, M. (2017): A comparitive study on traditonal healthcare system and present healthcare system using cloud computing and big data. *International Conference on Signal Processing and Communication (ICSPC)*, Coimbatore, pp. 269-273.

Song, G.; Rochas, J.; Beze, L.; Huet, F.; Magoules, F. (2016): K nearest neighbour joins for big data on mapreduce: a theoretical and experimental analysis. *IEEE Transactions on Knowledge & Data Engineering*, vol. 28, no. 9, pp. 2376-2392.

Wang, S.; Liu, Q.; Zhu, E.; Porikli, F.; Yin. J. (2018): Hyperparameter selection of one-class support vector machine by self-adaptive data shifting. *Pattern Recognition*, vol. 74, pp. 198-211.

Yang, J. (2017): Clustering analyzing of undergraduate schools based on k-means algorithm. *International Conference on Advanced Mechatronic Systems (ICAMechS)*, Xiamen, China, pp. 309-311.

Yan, Q. Y.; Zhu, M. L.; Tang, X. F. (2015): Electrical energy alternative research based on the cost utility analysis. *Operations Research & Management Science.*

Yildiz, A. B. (2015): Computer-based modelling of network functions for linear dynamic circuits using modified nodal approach. *Computer Modeling in Engineering & Sciences*, vol. 113, no. 3.