ITC 4/48
Information Technology and Control
Vol. 48 / No. 4 / 2019
pp. 673-681
DOI 10.5755/j01.itc.48.4.23881

Parallel Implementation of Improved K-Means Based on a Cloud Platform

Received 2019/07/23 | Accepted after revision 2019/11/07

http://dx.doi.org/10.5755/j01.itc.48.4.23881

Parallel Implementation of Improved K-Means Based on a Cloud Platform

Shufen Zhang
College of Sciences, North China University of Science and Technology, Tangshan 063210, China
Hebei Key Laboratory of Data Science and Application

Xuebin Chen
College of Sciences, North China University of Science and Technology, Tangshan 063210, China
Hebei Key Laboratory of Data Science and Application
Tangshan Key Laboratory of Data Science

Zhiyu Liu
College of Sciences, North China University of Science and Technology, Tangshan 063210, China
Hebei Key Laboratory of Data Science and Application

Changyin Luo
College of Sciences, North China University of Science and Technology, Tangshan 063210, China
Hebei Key Laboratory of Data Science and Application

Corresponding author: chxb@qq.com

In order to solve the problem of traditional K-Means clustering algorithm in dealing with large-scale data set, a Hadoop K-Means (referred to HKM) clustering algorithm is proposed. Firstly, according to the sample density, the algorithm eliminates the effects of noise points in the data set. Secondly, it optimizes the selection of the initial center point using the thought of the max-min distance. Finally, it uses a MapReduce programming model to realize the parallelization. Experimental results show that the proposed algorithm not only has high accuracy and stability in clustering results, but can also solve the problems of scalability encountered by traditional clustering algorithms in dealing with large scale data.

KEYWORDS: K-Means; MapReduce; Sample Density; Max-Min Distance.

1. Introduction

Clustering analysis is one of the most important methods for data mining and analysis. It aims at partitioning a data set into clusters such that the objects in a cluster have high similarity and the objects in different clusters are low similarity. Namely, the higher the similarity among the objects, the greater the probability that the objects are divided into the same cluster. Due to the simplicity and effectiveness, K-Means clustering algorithm is widely used to solve various problems in real applications, such as text evolutionary analysis, image clustering, community detection, etc. [30] Clustering analysis algorithms can be roughly divided into partition-based methods, hierarchical-based methods, density-based methods, grid-based methods, and model-based methods [9]. K-Means is one of the most widely used clustering algorithms, which are based on partitioning. The traditional K-Means algorithm is efficient and simple, however, it has three problems: clustering results are susceptible to isolated or noise points; clustering results depend on the selection of initial points; and there are algorithm scalability issues when dealing with large data sets.

In order to improve these deficiencies, some scholars have proposed a series of improvements from different perspectives. Junfeng proposed an improved K-Means algorithm based on density, which introduces information entropy and weighted distance, starting from the neighborhood density, removes the influence of isolated points on the algorithm, and then determines the initial clustering center, and therefore clustering center is relatively stable [14]. Yan proposed a PK-Means clustering algorithm based on multi-subgroups particle swarm optimization and pseudo means, which not only avoids empty clustering class, but also has well global convergence and local optimization, and also has a great effect on isolated data [20]. Jieling proposed an efficient clustering algorithm, which uses SDPCA (Space Division Preliminary Clustering Algorithm) to pre-cluster the dataset, and then uses OCANC (Optimized Clustering Algorithm based on Neighboring clusters) to optimize pre-clustering results and obtain the final clustering results [31]. Qian proposed a robust clustering center optimal algorithm based on weighted neighborhood distance, which focuses the clustering centers on data dense areas, overcoming problems of K-Means algorithm including poor stability [15]. Ping proposed a K-Means algorithm based on optimization of attribute weight using Lagrange multiplier method, which uses the weight value for each attribute to determine the importance of that attribute while computing the distance between an instance and the centroid of a cluster. In each iteration of clustering, it computes the optimal weight of attributes according to the change of centroid vector which minimizes the sum of distance between each instance and the centroid [28]. Xiaohui proposed a novel K-Means type method (a weighting K-Means clustering approach by integrating intra-cluster and inter-cluster distances, KICIC), which is able to integrate intra-cluster compactness and inter-cluster separation to solve the clustering problem of high-dimensional data [8]. Jiyong proposed a K-Means clustering algorithm based on distance threshold and weighted sample. It selects the sample mean as the first initial clustering center, and then dynamically determines clustering center and clustering number according to distance threshold, and finally uses weighted sample to reduce the influence of the clustering result [1]. Fei proposed a framework based on data stream, and based this framework, proposes an efficient K-Means algorithm, which uses an improved algorithm based on sampling to confirm clustering center for load balance and reducing iteration [24].

In view of the problems existing in the traditional K-Means algorithm, this paper makes the following improvements.

1 Data cleaning

This paper uses sample density as a criterion for determining outliers or noise points, finds outliers or noise points from the source dataset, and then removes them.

2 Selection of the initial clustering center

This paper uses “max-min distance” algorithm to determine the location of the initial centers, which is more representative.

3 Algorithm scalability

In order to effectively solve the K-Means clustering problem of large-scale data sets, this paper uses MapReduce to realize the parallelization of the algorithm.

2. Traditional K-Means Algorithm

K-Means algorithm is the most classical partition-based clustering method. Its basic ideas are as follows: randomly select k data objects from the source data set as the initial clustering center, and divide the remaining objects in the dataset into the nearest cluster based on their distance from each clustering center, then recalculate the mean of each cluster and update the value of the clustering center. Repeat the process until the standard criterion function converges [10]. Usually, the sum of squared errors is used as the clustering criterion function, which is defined as:

$$\mathrm{E} = \sum_{i=1}^{k} \sum_{d \epsilon D_i} |d - O_i|^2 \quad (1)$$

In Equation (1), *E* is the sum of the squared differences between the dataset object and the clustering center it is located, the larger its value, the greater the distance between the objects in the cluster and the clustering center, and the lower the similarity within the cluster. Otherwise, the higher the similarity within the cluster; k is the number of clusters; d is a data object within a cluster; D_i represents the i-th cluster, O_i is the clustering center of the i-th cluster.

The process of K-Means algorithm is given in literature [1]:

Input: the number of clusters (k), a sample set containing n data objects.

Output: k clusters with the smallest variance.

Process flow:

1. Randomly select k data objects as the initial clustering centers, recorded as O_1,O_2,⋯,O_k.
2. Calculate the distance of each object in the sample set from these clustering centers, and then assign it to the nearest cluster.
3. Recalculate the mean of each cluster and adjust the clustering center, the calculation formula is:

$$O_i = \frac{1}{n} \sum_{i=1}^{n} d_i, i = 1, 2, \cdots, n. \quad (2)$$

In Equation (2), n is the number of samples in the i-th cluster, d_i is a sample in the i-th cluster, $0i$ is the clustering center of the i-th cluster.

4. Repeat steps 2) and 3) until the clustering centers don not change, which indicates that the clustering criterion function converges.

K-Means algorithm is simple and fast, and high in efficiency for large data sets. On the other hand, it has many disadvantages:

1. The value of k needs to be specified by the user, but it is difficult.
2. Algorithm is highly dependent on the initial clustering center, which tends to increase the number of iterations and fall into local optimum.
3. Noise data has a large impact on the algorithm.

3. K-Means Algorithm Optimization

3.1. The Idea of Algorithm Optimization

The data objects in the low-density area are called isolated points or noise points when clustering [29]. In order to avoid the error caused by the noise data, we need to clean the data set to eliminate these noise data. This paper uses the Euclidean distance between samples as as a measure of similarity. First calculate the Euclidean distance between each sample in the dataset, and then delete the noise data. The cleaned data is in the high-density region, which can well reflect the distribution of data samples, thus eliminating the error caused by the noise data on the clustering results and improving the accuracy of the clustering results. When selecting k initial clustering centers, the traditional K-Means algorithm does not examine their effectiveness and representativeness [36]. When the k clustering centers belong to k different clusters, the algorithm can converge quickly and obtain better clustering results. However, when some of the k centers belong to the same cluster, it is easy to forcibly divide those data belonging to the same cluster into other clusters. Therefore, the clustering result of the traditional K-Means algorithm depends on the selection of the initial clustering centers [35]. This paper uses the "max-min distance" algorithm to optimize, that is, select k data samples that are the farthest distance from each other as the initial clustering center. The optimized algorithm still uses Equation (1) as the criterion function.

3.2. Defining Relevant Parameters

1 Sample density: *d* is a sample in data set D, taking it as the center of the sphere, *r* as the radius to form a spherical area. Then the number of samples contained in this spherical region is called the sample density, recorded as *Dens(x)* [6].

$$MDens = \sum_{i=1}^{n} \frac{Dist(d, x_i)}{n-1}. \tag{3}$$
$$Dens(x) = |\{p \| Dist(d, p) < r, p \in D\}|.$$

In Equation (3), *Dist(d,p)* represents the Euclidean distance between sample *d* and *p*.

3 Isolated point (noise point): *d* is a sample in data set D. Calculate the average distance from it to the adjacent points [32], the equation is as follows:

$$MDens = \sum_{i=1}^{n} \frac{Dist(d, x_i)}{n-1}. \tag{4}$$

If the average distance between two adjacent samples is greater than a certain threshold, it will be discriminated as an isolated point and deleted from data set D.

3.3. Algorithm Description

Input: number of clusters (k), source data set.

Output: k clustering results that cause the criterion function to converge.

The algorithm implementation process is divided into the following eleven steps:

1 Calculate the Euclidean distance between two adjacent samples.

2 Calculate the average distance from each sample to the adjacent points according to formula (4), then delete those data objects determined as isolated points, thereby obtaining a sample set in high density area.

3 Randomly select an object from cleaned data set, such as x_1, as the first clustering center Z_1.

4 Find the object furthest away from Z_1 in the data set as the second clustering center Z_2.

5 For each object remaining in the data collection, such as x_1, calculate its distance to Z_1 and Z_2, recorded as d_{i1} and d_{i2}, and the smaller one is represented by (min d_{i1}, d_{i2}).

6 Calculate the maximum of all min(d_{i1}, d_{i2}), recorded as max(min(d_{i1}, d_{i2})), and the corresponding object is marked as x_j.

7 If the following conditions are met:
max(min(d_{i1}, d_{i2}))> $m \times |Z_2 - Z_1|$ (*m* is the test parameter, $\frac{1}{2} \le m < 1$),
then use x_j as the third clustering center.

8 Repeat steps 5), 6) and 7), until finding k clustering centers.

9 Calculate the distances from each sample in the data set to the k cluster centers.

10 Assign each sample to the closest cluster based on distance, then recalculate the mean of all samples in each cluster and update the clustering center.

11 Repeat steps 9) and 10) until the criterion function converges.

4. Parallel Implementation

The optimization algorithm proposed above solves two problems of the traditional K-Means algorithm: the randomly selected initial cluster center is not representative, and the noise point has an adverse effect on the clustering result. However, there are still problems with efficiency and scalability when dealing with large-scale data.

The core of K-Means clustering is to calculate the distance between the sample and the clustering center, and assign the sample to the nearest clustering center. The operations are independent of each other, so they can be executed in parallel. This paper uses MapReduce to parallelize the K-Means algorithm to improve operating efficiency.

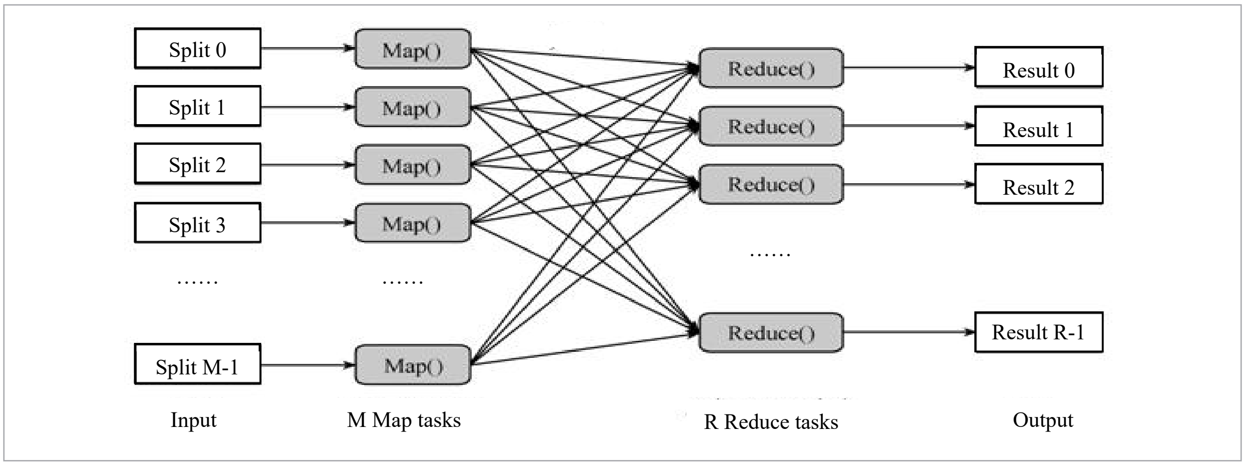
4.1. MapReduce Framework

MapReduce is a parallel programming framework proposed by Google for parallel analysis and operation of massive data sets. It can automatically parallelize programs and provide data segmentation and task scheduling, thus achieving high parallelism and scalability for various tasks [23, 13, 17].

Figure 1 shows the process of MapReduce processing large data sets. A MapReduce operation is divided into two phases: Mapping and Reducing. In the mapping phase, MapReduce framework splits the source

Figure1

The process of MapReduce



data into M splits, corresponding to M Map tasks. The input to each Map task is a set of key-value pairs converted from the split. The Map task calls a function named Map, and output a set of intermediate key-value pairs, and then sorts the output data according to the value of the intermediate keys, finally divides the set of intermediate key-value pairs into R fragments. In the reducing phase, the input to each Reduce task is output of the Map tasks. The Reduce task calls a Reduce function, and outputs the final result.

Hadoop is an open source distributed computing platform provided by AFS (Apache Software Foundation), which can be deployed in a cheap computer cluster [7]. The core of Hadoop is HDFS and MapReduce. HDFS is an open source implementation of GFS (Google File System) with high access speed, fault tolerance and scalability, and supports distributed storage of large-scale data [34, 19, 16]. MapReduce is an open source implementation for Google MapReduce, which allows users to develop parallel applications without knowing the underlying details of the distributed system [33, 5, 26, 25, 18, 22, 21, 10, 11]. This paper uses Hadoop MapReduce to implement the optimized algorithm, called HKM(Hadoop K-Means).

4.2. Algorithm Implementation

The implementation of HKM contains three MapReduce processes: the first MapReduce task is to select k initial clustering centers, the second is to clean the source data set, and the third is clustering.

1 Determining initial clustering centers

The task of the mapping phase is to read the source dataset file, parse it into a set of key-value pairs (which is expressed as <key, value>, key describes the offset of the current sample from file start, value is the string consisting of dimension coordinates of the current sample), and iteratively calculate the Euclidean distance between each sample and the clustering center, and then select the smallest of all the distances as the output of the mapping phase.

The pseudo code of the map function is as follows:

```
Map(<key, value>, <key1, value1>)
    define variables: mindist, index
    index ← hashMap.size()
    define array: dis, its size is the value of index
    for(i=0; i<index; i++)
        dis[i] ← distance between the current sample and i-th clustering center
    for(i=0; i<index; i++)
        if (dis[i] < mindist)
            mindist ← dis[i]
    output<key1,value1>, key1=index, value1=mindist+sample
```

The task of the reduction phase is to to use max-min distance algorithm to select the maximum from all the minimum distances output by the Map function, and output the data sample corresponding to the maximum value as the next clustering center.

The pseudo code of the reduce function is as follows:

```
Reduce(<key2, value2>, <key3, value3>)
    define variables: instance, dist, maxDist
    while (value2.hasNext())
        dist←the minimum distance analyzed from value2
        if (dist > maxDist)
                maxDist ← dist
                instance ← sample
    output<key3,value3>, k3=maxDist, value3 is a string consisted of the coordinates of instance
```

2 Cleaning data

The task of the mapping phase is to iteratively calculate the Euclidean distance between each sample and k initial center points, and selects the minimum distance as the output of the map phase.

The pseudo code of the map function is as follows:

```
Map(<key, value>, <key1, value1>)
    define variables: minDist, distance
    for(i=0; i<k; i++)
        distance ← Euclidean distance between the current sample and i-th clustering center
        if (distance < minDist)
             minDist ← distance
      output<key1,value1>, k1=key, value1=minDist+value
```

The task of the reduction phase is to find out the noise points and delete them. The pseudo code of the reduce function is as follows:

```
Reduce(<key2, value2>, <key3, value3>)
    define variables: maxDist, distance
    while (value2.hasNext())
        distance ←the minimum Euclidean distance analyzed from value2
        if (distance > maxDist/k)
        mark the current sample as a noise point
    output key-value pairs without noise points <key3,value3>
```

3 Clustering

The task at this phase is to aggregate the source data set into k clusters.

The pseudo code of the map function is as follows:

```
Map(<key, value>, <key1, value1>)
    define variables: minDist, dist, index
    for(i=0; i<k; i++)
        dist ← Euclidean distance between the current sample and i-th clustering center
        if (dist < minDist)
             minDist ← dist
               index ← index of the current cluster, which can be obtained from the configuration file
    output<key1,value1>, k1=index, value1=value
```

The pseudo code of the reduce function is as follows:

```
Reduce(<key2, value2>, <key3, value3>)
    define variable: num
    define arrary: temp
    num ← the number of samples parsed from value2
    while (value2.hasNext())
        temp ← dimension coordinates of each sample
        num ← num+1
    calculate the coordinates of each clustering center
    output <key3,value3>, key3=key2, value3 is a string consisted of new clustering center coordinates
```

5. Experimental Results and Analysis

The experiment configures Hadoop cluster on 7 nodes, configures the Name-Node service on the machines 01 and 02 and the Resource -Manager service on the machine 03, configures Data-Node and Node-Manager services on the other four machines, which are used as storage nodes of HDFS and compute nodes of MapReduce. The configuration of each machine is as follows: CPU: Inter(R) Core(TM) i5-3470; Memory: 4GB; Hadoop version: 2.4.1; Linux system version: CentOS-6.6.

The source data set used in the experiment is the data randomly generated by the program. In order to test the performance of the algorithm, four different data sets are generated, which are 200MB, 400MB, 600MB and 800MB.

In order to verify the validity of the initial center selection, the experiment selected 200MB data as the source data set, and performed 5 tests using traditional K-Means and HKM, the results are shown in Table 1.

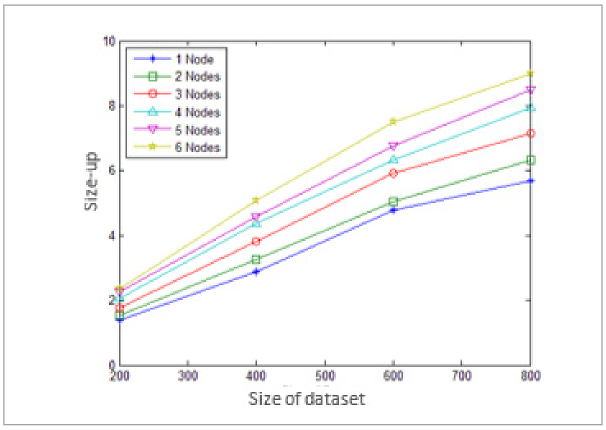
The experimental results show that HKM has a relative advantage in the accuracy and stability of the clustering results.

Table 1
The results of two clustering algorithms

Serial number	Traditional K-Means		HKM	
	initial center	validity / (%)	initial center	validity / (%)
1	106,37	85.31	57,136	84.96
2	128,46	83.39	57,136	84.96
3	34,185	57.72	57,136	84.96
4	167,84	79.58	57,136	84.96
5	62,141	89.75	57,136	84.96
Average		79.15		84.96

In order to test the performance of the algorithm, the experiment selected, this paper uses size-up, speed-up and scale-up as evaluation criteria. The method of testing size-up is to keep the number of computing nodes unchanged and continuously increase the data size. Figure 2 shows the size-up of HKM under the Hadoop platform with different number of computing nodes.

Figure 2
The test results of the size-up performance



The experimental results show that as the data scale continues to increase, the scaling rate is also increasing. Therefore, the HKM algorithm can solve the scalability problem faced by traditional K-Means when dealing with large-scale data.

Speed-up is used to measure the performance of parallel programs or system parallelism, which is the running time ratio of the same task in a single processor system and pa rallel processor system. The method of testing speed-up is to keep the data size unchanged and continuously increase the computing nodes.

In this paper, four data sets (200MB, 400MB, 600MB, 800MB) are used, and the number of computing nodes is gradually increased from 1 to 6. We recorded the time that the HKM algorithm runs on a cluster of different nodes, and finally calculated the speed-up. The result is shown in Figure 3.

Figure 3
The test results of speed-up performance

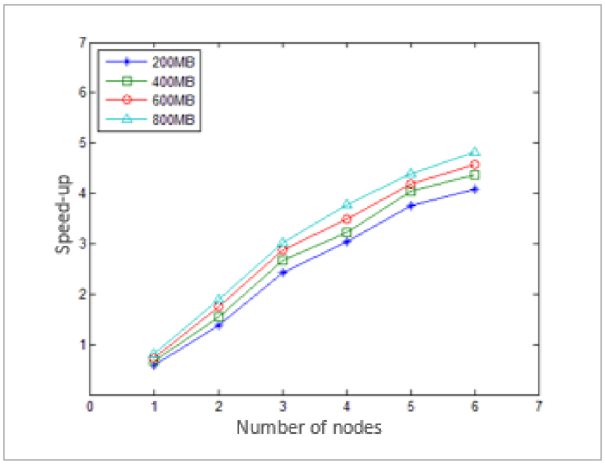
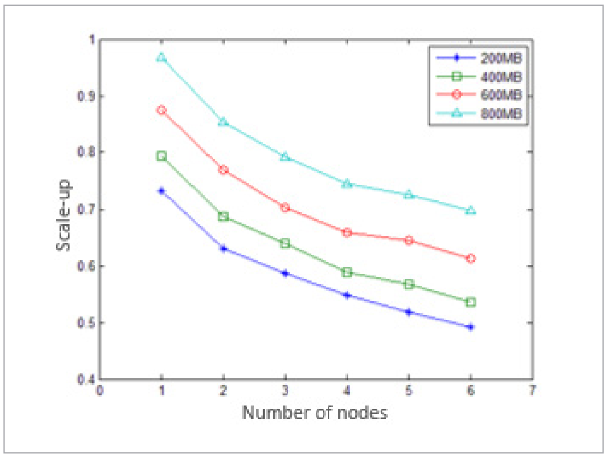


Figure 4
The test results of scale-up performance



The experimental results show that as the data size and computing nodes increase, the speed-up performance of the algorithm continues to increase.

The way of testing the scale-up is to increase computing nodes while increas ing the size of the data. In this paper, four data sets (200MB, 400MB, 600MB, 800MB) are used, and the number of computing nodes is gradually increased from 1 to 6. We recorded the running time under different conditions, and finally calculated the scale-up. The experimental result is shown in Figure 4. The results show that the HKM algorithm has better scalability when dealing with large-scale data sets.

6. Conclusion

This paper analyzes the problems of traditional K-Means, and then proposes the HKM algorithm, which uses the sample density method to eliminate the noise data in the data set, and select the initial clustering center according to "max-min distance" algorithm, and finally uses MapReduce Implement this algorithm in parallel. The experimental results show that the HKM algorithm can not only improve the accuracy of clustering results, but can also be effectively applied to data mining and analysis of large-scale data.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61702091), the Key Research and Development Program of Shaanxi Province (No. 2018ZDXM-GY-036), the Fundamental Research Funds for the Central Universities (No. 2572018BH06) and Scientific Research Program Funded by Shaanxi Provincial Education Department (Program No. 2013JK1139).

References

1. An, J. Y, Yan, Z. J., Zhai, J. X. K-Means Clustering Algorithm Based on Distance Threshold and Sample Weighting. Microelectronics and Computer Science, 2015(8), 135-138.
2. Behera, R., Rath, S., Misra, S., Damaševičius, R., Maskeliūnas, R. Large Scale Community Detection Using a Small World Model. Applied Sciences, 2017, 7(11), 1173. https://doi.org/10.3390/app7111173
3. Capizzi, G., Lo Sciuto, G., Woźniak, M., Damaševicius, R. A Clustering Based System for Automated Oil Spill Detection by Satellite Remote Sensing. In Artificial Intelligence and Soft Computing, 2016, 613-623. https://doi.org/10.1007/978-3-319-39384-1_54
4. Chen, X. S., W, X. S., Wan, W. X. A K-Means Initial Clustering Center Optimization Algorithm Based on Feature Relevance. Journal of Sichuan University (Natural Science Edition), 2015, 47(1), 13-19.
5. Fan, X., Song, H., Wang, H. Video Tamper Detection Based on Multi-Scale Mutual Information. Multimedia Tools & Applications, 2017, 1-18. https://doi.org/10.1007/s11042-017-5083-1
6. Fu, B. L, Zhan, A. K. A K-Means Clustering Text Mining Method Based on Mean Density Center Estimation. Journal of Chongqing University of Posts and Telecommunications (Natural Science Edition), 2014, 26(1), 111-116.
7. Han, W., Zhang, X., Chen, Y. MapReduce Based Image Classification Approach. Journal of Computer Applications, 2014.
8. Huan, X., Wan, C., Xion, L., Zen, H. A Weighting K-Means Clustering Approach by Integrating Intra-cluster and Inter-Cluster Distances. Chinese Journal of Computers, 21(7), 683. https://doi.org/10.3390/e21070683
9. Ji, Z., Pi, H., Wei, W., Xiong, B., Wozniak, M., Damasevicius, R. Recommendation Based on Review Texts and Social Communities: A Hybrid Model. IEEE Access, 2019, 7, 40416-40427. https://doi.org/10.1109/ACCESS.2019.2897586
10. Ke, Q., Zhang, J., Song, H., Wan, Y. Big Data Analytics Enabled by Feature Extraction Based on Partial Independence. Neurocomputing, 2017, 288, 3-10. https://doi.org/10.1016/j.neucom.2017.07.072
11. Lakshmanaprabu, S. K., Shankar, K., Khanna, A., Gupta, D., Rodrigues, J. J. P. C., Pinheiro, P. R., De Albuquerque, V. H. C. Effective Features to Classify Big Data Using Social Internet of Things. IEEE Access, 2018, 6, 24196-24204. https://doi.org/10.1109/ACCESS.2018.2830651
12. Li, J., Gu, H. S. Optimization Method of K-Means Clustering Center in Cloud Computing. Bulletin of Science and Technology, 2015, 31(10), 100-102.
13. Li, X., Men, Z., Xu, C. A Practical Performance Model for Hadoop MapReduce. IEEE International Conference on CLUSTER Computing Workshops. IEEE Computer Society, 2015, 231-239.
14. Lu, J. F., Su, Z. H. A Clustering Algorithm Based on Density for K-Means. Microelectronics and Computer Science, 2014(10), 28-31.

15. Lu, Q. Robust Optimization Algorithm for K-Means Clustering Centers. Computer Engineering and Design, 2015, 36(9), 2395-2400.

16. Ma, H. D., Ha, X. Y, Ma, R. Q. Implementation of Web Log Mining Based on Hadoop's Parallel PSO - K-Means Algorithm. Computer Science, 2015, 42(s1).

17. Odia, T., Misra, S., Adewumi, A. Evaluation of Hadoop/Mapreduce Framework Migration Tools. In IEEE Asia-Pacific World Congress on Computer Science and Engineering, 2014. https://doi.org/10.1109/APWCCSE.2014.7053870

18. Qiang, Y., Zhang, J. A Bijection Between Lattice-Valued Filters and Lattice-Valued Congruences in Residuated Lattices. Mathematical Problems in Engineering, 2013, 36(8), 4218-4229. https://doi.org/10.1155/2013/908623

19. Rao, B. T., Reddy, L. S. S. Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environments. Computer Science, 2015, 34(9), 29-33.

20. Shen, Y., Yu, D., Hua, W. H. L. An Improved K-Means Clustering Algorithm for Particle Swarm Optimization. Computer Engineering and Applications, 2014, 50(21), 125-128.

21. Shua, L., Wenji, L., Dingzh, D. Fractal Intelligent Privacy Protection in Online Social Network Using Attribute-Based Encryption Schemes. IEEE Transactions on Computational Social Systems, 2018, 5(3), 736-747. https://doi.org/10.1109/TCSS.2018.2855047

22. Srivastava, H. M., Zhang, Y., Wang, L., Shen, P., Zhang, J. A Local Fractional Integral Inequality on Fractal Space Analogous to Anderson's Inequality. Abstract and Applied Analysis, 2014, 46(8), 5218-5229.

23. Tia, B. J., Du, X. J., Yang, H. Y., Su, Y. L. Research of Hybrid Collaborative Filtering Optimized Technology in Cloud Computing. Application Research of Computers, 2018, 35(7), 2079-2083.

24. Wan, F., Qin, X. L, Liu, L. Clustering Algorithm Based on Data Flow in Cloud Environment. Computer Science, 2015, 42(11), 235-239.

25. Wei, W., Yang X.-L., Shen, P. Y., Zhou, B. Holes Detection in Anisotropic Sensornets: Topological Methods. International Journal of Distributed Sensor Networks, 2012, 8(10), 135054. https://doi.org/10.1155/2012/135054

26. Wei, W., Yang, X.-L., Zhou, B., Feng, J., Shen, P.-Y. Combined Energy Minimization for Image Reconstruction from Few Views. Mathematical Problems in Engineering, 2012, 154630. https://doi.org/10.1155/2012/154630

27. Xing, C. Z., Gu, H. A K-Means Algorithm for Initial Clustering Center Optimization Based on Average Density. Computer Engineering and Applications, 2014, 50(20), 135-138.

28. Xiong, P., Gu, X. K-Means Clustering Algorithm Based on Attribute Weight Optimization. Microelectronics and Computer Science, 2014(4), 40-43.

29. Yang, Z., Luo, K. An Improved Clustering Algorithm Based on Particle Swarm Optimization. Journal of Computer Applications, 2014, 31(9), 2597-2599.

30. Zhang, J., Zhuo, L., Zhu, Y. X. Improvement and Application of K-Means Clustering Algorithm. Application of Electronic Technique, 2015, 41(1), 125-128.

31. Zhang, J. L, Bai, Q. Y. An Efficient K-Means Clustering Algorithm. Journal of Fuzhou University (Natural Science Edition), 2014, 42(4), 537-542.

32. Zhang, W. J, Jiang, L. H. Parallel Computation Algorithm for Big Data Clustering Based on MapReduce. Application Research of Computers, 2018, 37(1).

33. Zhao, W. Z., Ma, H. F., Fu, Y. X. Research on Parallel K-Means Clustering Algorithm Based on Cloud Computing Platform Hadoop. Computer Science, 2011, 38(10), 166-168.

34. Zhou, R. W., Li, Z. Y., Chen, S. M. Parallel Optimal Sampling Clustering K-Means Algorithm for Large Data. Computer Application, 2016, 36(2), 311-315.

35. Zhou, W. B., Shi, Y. X. An Optimization Algorithm of K-Means Clustering Center Selection Based on Density. Computer Application Research, 2012, 29(5), 1726-1728.

36. Zhu, Y. X., Li, Y. L., Cun, M. T. CARDBK Clustering Algorithm with Improved K-Means Algorithm. Computer Science, 2015, 42(3), 201-205.