Spark-Based Parallel Implementation of an Improved K-means Algorithm (Postprint)

Authors: Du Jiaying, Duan Longzhen, Duan Wenying, Bu Qiujin

Date: 2018-12-13T00:00:00+00:00

Abstract

To address the limitations of the K-means clustering algorithm, we propose an improved K-means to enhance algorithmic performance. The simplified silhouette coefficient is employed as an evaluation criterion to determine the k value in the K-means algorithm, while K-means++ is utilized to complete the selection of initial centroids. After establishing the k value and initial centroids, morphological similarity distance is adopted as the similarity measurement metric to assign data points to clusters formed by the nearest centroids. Finally, the average silhouette coefficient is calculated to determine the appropriate k value, and algorithm parallelization is implemented on Spark. Experimental results on four standard datasets across three dimensions—accuracy, execution time, and speedup ratio—demonstrate that the improved K-means algorithm not only enhances clustering partition quality and reduces computational time compared to the traditional K-means algorithm and SKDK-means algorithm, but also exhibits excellent parallel performance in multi-node cluster environments. Analysis of the experimental results reveals that the proposed improved algorithm can effectively enhance execution efficiency and parallel computing capability.

Full Text

Preamble

Vol. 37 No. 2
Application Research of Computers
ChinaXiv Cooperative Journal

Parallel Implementation of Improved K-means Algorithm Based on Spark

Du Jiaying, Duan Longzhen, Duan Wenying, Bu Qiujin
(College of Information Engineering, Nanchang University, Nanchang 330031,

China)

Abstract: To address the deficiencies of the K-means clustering algorithm, this paper proposes an improved K-means algorithm to enhance performance. The simplified silhouette coefficient is employed as an evaluation metric to determine the optimal k value in K-means, while K-means++ is adopted for selecting initial centroids. After determining the k value and initial centroids, morphology similarity distance (MSD) is used as the similarity measure to assign data points to the nearest centroid' s cluster. Finally, the average silhouette coefficient is calculated to determine the appropriate k value, and the algorithm is parallelized on Spark. Experiments on four standard datasets demonstrate that the improved K-means algorithm not only enhances clustering quality and reduces computation time compared with traditional K-means and SKDK-means algorithms, but also exhibits excellent parallel performance in multi-node cluster environments. The results confirm that the proposed improvements effectively enhance algorithm execution efficiency and parallel computing capability.

Keywords: clustering algorithm; simplified silhouette coefficient; morphology similarity distance; similarity measurement

0 Introduction

Clustering analysis represents a crucial research topic in data mining and serves as a primary task for exploring data patterns, with widespread applications across multiple domains including image processing [**?**], information retrieval [**?**], and text mining [**?**]. The K-means algorithm is widely used in clustering analysis, yet it suffers from two major limitations: the uncertainty of the k value leads to deviations between actual and ideal results after each iteration [**?**], and the arbitrary selection of initial points causes instability, resulting in significant errors in the final clustering outcomes.

With the exponential growth of data, parallel computing frameworks such as MapReduce and Spark have gained prominence. MapReduce is ill-suited for iterative computations because intermediate results are stored on disk, requiring repeated read/write operations on the Hadoop Distributed File System (HDFS) that incur high I/O overhead and time consumption [**?**]. Spark overcomes these limitations by providing an in-memory distributed framework that stores intermediate results in memory and offers fast startup speeds. In this era of big data, numerous scholars have proposed various Spark-based improved algorithms. Xu et al. [**?**] utilized the Canopy algorithm to determine cluster numbers and selected initial points using maximum-minimum distance, though Canopy requires pre-setting thresholds T1 and T2, making results highly sensitive to these parameters. Deng et al. proposed an improved K-means algorithm based on minimum criterion functions, determining k initial centers through i sampling iterations [**?**]. Song et al. [**?**] presented a parallel algorithm employing kd-trees for initial point selection and nearest neighbor searches to reduce distance calculations and

accelerate clustering for massive datasets. While the latter two works focused on improving initial point selection, the k value remained uncertain—yet selecting an appropriate k value critically impacts clustering accuracy. Therefore, this paper adopts the simplified silhouette method as a validity assessment approach to determine the optimal number of clusters k. To address the random initialization issue in K-means, Arthur et al. proposed the K-means++ algorithm [**?**], which automatically obtains k initial points, avoids instability from random selection, and accelerates convergence. This paper incorporates K-means++ to enhance algorithm performance.

Building upon these concepts, this paper proposes an improved K-means algorithm with parallel implementation on Spark to accelerate convergence, improve accuracy, and achieve faster processing times for large-scale datasets while obtaining superior clustering results. Standard UCI datasets are used to demonstrate the enhanced performance of the improved algorithm.

1.1 Spark Framework

Apache Spark is an open-source distributed framework for large-scale data clustering that provides excellent features including data parallelism and high fault tolerance, along with simple programming interfaces for Python, Java, and Scala [**?**]. Spark' s abstraction of Resilient Distributed Datasets (RDDs) enables efficient handling of iterative tasks. RDDs are read-only datasets in a cluster that can be created either from files stored in external systems such as HDFS or derived from other RDDs. RDD operations are divided into two types: transformations, which create new datasets from existing ones, and actions, which return computation results to the Driver program. This programming model is illustrated in [Figure 1: see original paper].

Spark' s cluster architecture follows a master-slave model, consisting of one Master node and multiple Slave nodes. The Master node is responsible for task allocation, scheduling, and monitoring across the cluster, while Slave nodes execute Worker tasks, primarily handling computations and generating status reports. Spark supports various deployment modes including standalone, cluster, YARN, and Mesos.

The mathematical foundations for the proposed algorithm are defined as follows:

Definition 1 (Simplified Silhouette Method) [?]:
The simplified silhouette coefficient measures clustering validity using distances to cluster centers rather than average distances to all points as in the original silhouette method, thereby reducing computation time while yielding better results across different K values. For a data point i, let $a(i)$ denote its Euclidean distance to its cluster center, and $b(i)$ denote the minimum distance from point i to other cluster centers. The simplified silhouette value $s(i)$ is calculated as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

The average simplified silhouette value ss for dataset D containing n points is:

$$ss = \frac{1}{n}\sum_{i=1}^{n} s(i)$$

The value ranges from -1 to 1, where -1 indicates poor clustering and 1 indicates excellent clustering. Therefore, k values producing ss closest to 1 should be selected.

Definition 2 (Minkowski Distance):
For two vectors X_j and Y_k with dimensionality m, the Minkowski distance is:

$$D_{jk} = \left(\sum_{i=1}^{m} |X_{ji} - Y_{ki}|^p\right)^{1/p}$$

where X_{ji} and Y_{ki} represent the i-th attribute values of vectors j and k, respectively. When $p = 1$, this becomes Manhattan distance; when $p = 2$, it becomes Euclidean distance.

Definition 3 (Morphology Similarity Distance, MSD) [**?**]:
MSD combines attribute difference and spatial distance:

$$MSD = \frac{2 \times ED \times ASD}{ED + ASD}$$

where ED represents Euclidean distance and ASD (Absolute Sum of Differences) is defined as:

$$ASD = \sum_{i=1}^{m} |X_{ji} - Y_{ki}|$$

If all attribute differences are simultaneously greater than or smaller than the mean values, ASD reduces to Manhattan distance. Table 1 demonstrates distance calculations between vectors, showing that when $ASD/SAD = 1$ (i.e., ASD equals Manhattan distance), MSD becomes equivalent to Euclidean distance.

1.2 K-means++ Algorithm

K-means++ is a clustering algorithm for selecting initial centroids in K-means, first proposed by David Arthur and Sergei Vassilvitskii in 2007 [**?**]. It addresses the limitations of random initialization in traditional K-means and serves as an improvement that enhances stability and convergence speed. The algorithm flow is shown in [Figure 2: see original paper].

The specific procedure is as follows: 1. Randomly select one point from the dataset as the first centroid. 2. For each sample point x in the dataset, compute its shortest distance $D(x)$ to the currently selected centroids. 3. Select a new point as the next centroid with probability proportional to $D(x)^2$, where points with larger $D(x)$ have higher selection probability. 4. Repeat steps 2-3 until k centroids are selected.

2.1 Improved K-means Algorithm

The proposed improved K-means clustering algorithm aims to enhance accuracy and accelerate convergence. The core idea involves: first presetting a k value and obtaining k initial centroids through K-means++; then iteratively assigning all data points to the nearest cluster using MSD distance; recomputing cluster centroids based on assigned points; repeating until centroids stabilize; and finally calculating simplified silhouette coefficients across different k values to determine the optimal cluster number and corresponding clustering results.

2.2 Parallel Implementation of Improved K-means on Spark

To effectively enhance parallel computing capability, this paper proposes a Spark-based improved K-means algorithm centered around "map" and "reduce" operations, with the workflow illustrated in [Figure 3: see original paper]. During parallelization, each partition corresponds to a pair of map and reduce operations that execute simultaneously without interference. The `mapPartition` operator assigns data points in each partition to their nearest cluster centers, with each map operation completing in parallel. The `reduceByKey` operator merges data points belonging to the same cluster across all partitions and updates cluster centers, with the reduce process also executing in parallel. Each iteration performs multiple concurrent map-reduce operations, and the final clustering result is determined by comparing simplified silhouette coefficients across different k values.

[Figure 4: see original paper] illustrates the three-phase architecture of the Spark-based improved K-means algorithm: data partitioning, aggregation, and validation.

Phase 1: Data Partitioning
a) Prepare the Spark environment for in-memory computation, storing intermediate results in memory without disk output.
b) Read data from HDFS into memory as RDDs.
c) Convert data to dense format, split into partitions, and transform into `<key,value>` pairs. These partitioned data are distributed across compute nodes, with `mapPartition` assigning each node' s data slices to different cluster centers. `cache()` enables data persistence in memory, where each partition constitutes an RDD.

Phase 2: Aggregation

This phase comprises steps 4-8. The k value is manually set and broadcast to all partitions. Using the proposed MSD distance metric, data points are assigned to clusters formed by their nearest centroids. The `reduceByKey` operator then aggregates points with identical `centerId` in `<centerId,pointList>` pairs to form new partitions, counts points per cluster, and updates cluster centers. Iteration continues if the current iteration count is below the maximum threshold and centroids have not converged; otherwise, the aggregation process terminates.

Phase 3: Validation
Step 9 computes the average simplified silhouette value for the entire dataset as the evaluation metric for K-means validity to select the appropriate k value.

3 Experimental Results and Analysis

The improved K-means algorithm was implemented on a Spark distributed cluster consisting of one master node and three slave nodes. Each machine was configured with a dual-core 2.5 GHz CPU, 4 GB RAM, and 500 GB hard disk, running 64-bit Ubuntu 16.04. The test datasets—Iris, Wine, Glass, and Flame —were downloaded from the UCI repository, with details shown in Table 2 .

a) Accuracy
Since each dataset record has a corresponding class label, accuracy is defined as the ratio of correctly clustered points to total sample points, reflecting algorithmic precision. [Figure 5: see original paper] compares the accuracy of the improved K-means algorithm against traditional K-means and the algorithm from reference [8] across four datasets in a standalone environment. The results show significant accuracy improvements for the enhanced algorithm on all datasets, attributable to better k value determination via simplified silhouette coefficient, more stable initialization through K-means++, and effective point assignment using MSD distance.

b) Runtime
Runtime evaluates algorithmic efficiency and determines execution speed. [Figure 6: see original paper] compares the execution times of the three algorithms across different datasets. The parallelized improved algorithm demonstrates reduced time consumption due to fewer iterations and faster convergence, thereby enhancing computational performance. The most time-consuming phase involves computing MSD distances from all points to their cluster centers, with runtime increasing as dataset size grows. [Figure 7: see original paper] shows the improved algorithm' s runtime under varying node counts and dataset types, revealing that execution time decreases and gradually stabilizes as nodes increase, because inter-node communication overhead eventually offsets cluster performance gains.

c) Speedup
Speedup ratio precisely measures parallel algorithm effectiveness by comparing serial versus parallel execution times for the same task, calculated as $R = T_s/T_p$,

where T_s is serial runtime and T_p is parallel runtime on an s-node cluster. Larger R values indicate less parallel computation time and higher parallelism. [Figure 8: see original paper] demonstrates the improved algorithm' s speedup across four datasets, showing that speedup increases with node count. Vertically, larger datasets achieve higher speedup under identical node configurations. When nodes increase, each node processes less data, reducing T_p and increasing speedup. Similarly, with fixed node counts, larger datasets increase serial time T_s, thereby improving parallelism. However, the speedup growth rate declines as nodes increase, because the experimental data size is insufficient to exceed single-node memory capacity, and inter-node communication overhead partially negates performance gains from additional nodes.

In summary, comparative analysis of the improved K-means algorithm against reference [8] and traditional K-means reveals significant performance improvements in both runtime and accuracy. Speedup analysis confirms excellent parallel computing capability. Overall, the proposed algorithm demonstrates clear superiority.

4 Conclusion

This paper proposes an improved K-means algorithm with parallel implementation on the Spark distributed computing framework. The algorithm employs K-means++ to select k initial centroids, uses MSD distance as the similarity metric for cluster assignment, and determines the optimal k value through simplified silhouette coefficient analysis. Experimental results demonstrate that the improved algorithm effectively enhances clustering accuracy, reduces runtime, and exhibits strong parallel performance advantages in Spark cluster environments. The current study employs a small-scale cluster for modest data volumes. Future research will expand cluster scale, further optimize algorithmic parallelism, and validate robustness using larger-scale datasets.

References

[1] Li Bin, Li Rong, Zhou Lei. Research and implementation of distributed k-means clustering algorithm [J]. *Computer Engineering & Software*, 2018, 39(1): 35.

[2] Mahbub U, Imtiaz H, Ahad M A R. Action recognition based on statistical analysis from clustered flow vectors [J]. *Signal, Image and Video Processing*, 2014, 8(2): 243–253.

[3] Lipka N, Stein B, Anderka M. Cluster-based one-class ensemble for classification problems in information retrieval [J]. *SIGIR*, 2012.

[4] Joshi K D, Nalwade P S. Modified K-means for better initial centers [J]. *International Journal of Computer Science and Mobile Computing*, 2013, 2(7): 219-223.

[5] Srirama S N, Batrashev O, Jakovits P, et al. Scalability of parallel scientific applications on the cloud [J]. *Scientific Programming*, 2011, 19(2): 91-105.

[6] Xu Pengcheng, Wang Cheng. Improvement of K-means algorithm and implementation based on Spark computing model [J]. *Journal of Nanjing University of Posts and Telecommunications: Natural Science Edition*, 2017, 37(4): 113-118.

[7] Deng Qing, Yang Ning. Research of improved parallel K-means algorithm based on Spark framework [J]. *Intelligent Computer and Applications*, 2018, 8(1): 76-78.

[8] Song Dongfei, Xu Hua. Parallel implementation of improved K-means algorithm based on Spark [J]. *Computer System & Applications*, 2018, 27(4): 151-156.

[9] Arthur D, Vassilvitskii S. K-means++: the advantages of careful seeding [J]. *SODA*, 2007, 1027-1035.

[10] Zaharia M, Chowdhury M, Franklin M J, et al. Spark: cluster computing with working sets [J]. *HotCloud*, 2010, 1-6.

[11] Wang Fei, Hugo H, Penya F. An analysis of the application of simplified silhouette to the evaluation of k-means clustering validity [J]. *Machine Learning and Data Mining in Pattern Recognition*, 2017, 10358: 291-305.

[12] Li Zhong, Yuan Jinsha, Zhang Weihua. Fuzzy C-mean algorithm with morphology similarity distance [J]. *Fuzzy systems and knowledge discovery*, 2009: 90-94.

Note: Figure translations are in progress. See original paper for figures.