Improvement of Parallelization of KNN Algorithm Based on Sequential MapReduce

Yaoshun Li and Lizhi Liu

***Abstract*—In order to further improve the computing efficiency of the KNN algorithm in a cluster environment, an improved parallelization algorithm based on the sequential MapReduce framework was proposed. Two MapReduce workflows were defined in sequence. The first workflow took training data as input to parallelize the process of calculating Euclidean distance; the second workflow took the output of the calculation process as input to parallelize the process of counting categories. The Experiments have verified that in a cluster environment, compared with a single MapReduce operation process, the sequential improvement accelerated the operation process of counting categories and improved the efficiency of the algorithm.**

***Index Terms*—KNN, MapReduce, parallelization.**

I. Introduction

The KNN algorithm determines the classification of samples by measuring the Euclidean distance between unknown data and known samples, and is often used to deal with problems such as classification and regression [1], [2]. With the advent of the big data era, the complexity and dimensionality of data have gradually increased, and the KNN algorithm running on a single machine has been unable to efficiently process large quantities of data. In response to the above problems, Wenjin Xu and other researchers have proposed some improvements: using clustering algorithms to reduce the dimensionality of the training data set, accelerating the operation process through the CUDA computing platform and GPU, combining the KNN algorithm with other machine learning algorithms, and using filters to filter the data and so on [3]-[6]. Which are based on stand-alone computing. Although the computing efficiency is improved, the amount of data that can be processed is also limited. With the development of distributed clusters, Ying Ma and other researchers proposed to use distributed clusters such as Hadoop and Spark to parallelize the algorithm to solve the problem of overflowing memory of single machine [7]-[12]. Whose improved method does solve the calculation problem in the big data environment, but only parallelizes the calculation process of Euclidean distance, and finally generates a large number of classification labels, which still requires time-consuming statistics. Based on the Hadoop platform, this article discusses using sequential MapReduce framework to improve the KNN algorithm for secondary continuous parallelization. After comparing the training process before and after the improvement, the correctness and advantages of algorithm were verified by experiments.

Manuscript received November 11, 2020; revised February 25, 2021. This paper is supported by the Open-end Fund of Hubei Key Laboratory of Intelligent Robot (Wuhan Institute of Technology) 430073, China (HBIR 201902).

Yaoshun Li and Lizhi Liu are with School of Artificial Intelligence, School of Computer Science and Engineering, Wuhan Institute of Technology, Wuhan, China (e-mail: liyaoshuncn@foxmail.com).

II. Research Background

The principle of the KNN algorithm is that it is assumed that there are m known sample data are distributed in a Euclidean feature space, and the position and data categories of all sample in this space are known. Assuming that each sample has n feature values, now giving a new sample X, we need to determine the category to which the sample belongs. The proceed as follows:

Calculating the Euclidean distance between the new sample and all known samples, the calculation formula is as follows:

$$distance = \sqrt{\sum_{i=1}^{n}(X_i^{(a)} - X_i^{(b)})^2} \quad (1)$$

where $X_i^{(a)}$ represents the *i*-th feature value in the known sample, and $X_i^{(b)}$ represents the i-th feature value in the new sample.

The core idea of the KNN algorithm is to assume that the closer the samples in the space are, the more likely it is to belong to one class. Therefore, after sorting the m Euclidean distances in ascending order, to select the first *k* category labels corresponding to the sample data and count the number of occurrences of each category. The category with the highest frequency is the unknown sample category.

All training data and the test data must be counted again each classification or regression. If the amount of data is large, the computing power required will be amazing. In addition, the KNN algorithm is extremely dependent on training data. If there are some data are wrong in the training data, just right which are placed in the periphery of the data need to classify, which will directly lead to predict data no accurately. With the data dimension increasing, the time and space complexity of the algorithm will also become higher and higher.

III. KNN Algorithm Parallelization

A. Feasibility Analysis

In the case of a relatively small amount of data, the known sample data can be regarded as a two-dimensional matrix, and the new sample data can be regarded as a one-dimensional vector, the Euclidean distance can be easily

DOI: 10.7763/IJCTE.2021.V13.1290

calculated by matrix operation by the Numpy library in Python, and the frequency of samples can also be counted quickly by using the Counter method provided in the collections library.

As can be seen from the prediction process of the KNN algorithm, the two processes of formula (1) and counting categories are the core steps of the algorithm. The number of times to calculate the Euclidean distance is the same as the number of sample data. The sorting statistical process in a big data environment also means a high time and space complexity.

It can be seen that formula (1) is a repetitive operation of all sample data, and the process of counting labels can also be regarded as the merging of small-scale data processing results. The two calculation processes are in line with the idea of dividing and conquering distributed clusters, and the calculation and statistical processes can be parallelized through the MapReduce workflow. From the analysis of the algorithm running process, calculation and statistics have a sequential relationship. Therefore, we consider defining two sequential MapReduce workflows and the output of the calculation process is used as the input of the statistical process, and the final calculation result is the predicted value of the KNN algorithm.

B. KNN Algorithm Based on Sequential MapReduce

MapReduce is a programming framework that can be used for data processing. It adopts the "divide and conquer" idea to distribute operations on large-scale datasets to each child node to complete, and then integrates the intermediate results of each node to obtain the final result. MapReduce highly abstracts the processing process into two functions, map and reduce. Map is responsible for decomposing the task into multiple subtasks, and reduce is responsible for summarizing the processing results of the multiple subtasks.

In the KNN algorithm process, the calculation process is improved to the first MapReduce workflow, and the statistical process is improved to the second MapReduce workflow. The overall workflow is shown in Fig. 1.

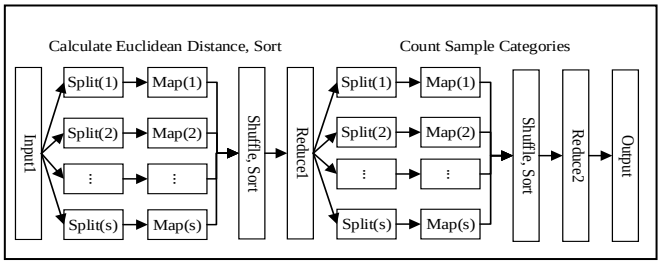

Fig. 1. Sequential MapReduce workflow.

The steps of sequential MapReduce are as follows:

1) MapReduce workflow of calculating Euclidean distance

a) Split

Store the training dataset in the HDFS file system. When the job starts, MapReduce will read the data file and split it according to the HDFS data block size, each slice will be processed by a Map node.

b) Parse key-value pairs

The minimum unit of data processing in MapReduce is key-value pairs. Before the data in the *i*-th is input into the map function, it will be converted into the key-value pairs of "<text start position, text content>" according to the default rule. Each row of sample data "$x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}, y^{(i)}$" will be converted into an input key-value pair "$< \text{key}, \{x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}, y^{(i)}\} >$". Where "$x_n^{(i)}$" represents the nth eigenvalue of the *i*-th row, and "$y^{(i)}$" represents the label value of the *i*-th row, the key value only plays a role in the form of a key, which is specified by the system by default.

c) Map

Each slice is processed by a Map node, and every record in the slice is parsed, which will call a map function. First, the sample feature values "$x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}, y^{(i)}$" will be converted to a string array. After that, the eigenvalue array "$\text{x\_train}[\,] = [x_0^{(i)}, x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}]$" will be extracted by the slicing operation, where "$x_0^{(i)}$" is the sample deviation value, which is designated as an integer 1 by default.

Subtract the eigenvalue array "$\text{X}[\,] = [X_0, X_1, X_2, \dots, X_n]$" and "x_train[ ]" to obtain a new array. Then calculate the Euclidean distance $distance^{(i)}$ of the current sample by formula (1). And then key-value pairs "$< distance^{(i)},\ y^{(i)} >$" will be output to intermediate results.

When all the fragmented map processes have been processed, the entire Job Map process also ends. Finally, all output results will overflow into the HDFS local disk and be saved as a file.

d) Shuffle

All the key-value pairs output will be sorted and merged in the ascending order of distance in the Shuffle process, and then the results will be transmitted to the Reduce stage in the form of a key-value pair "$< distance^{(i)}, < y^{(1)}, y^{(2)} \dots . y^{(i)} > >$" for processing.

e) Reduce

Traverse and output all key-value pairs "$< distance^{(i)},\ y^{(i)} >$" to a file directly.

After the output is completed, the second Job process will be started immediately, and the output file will be used as the input of the next MapReduce process.

2) MapReduce workflow of counting labels

Read the first k items of data in the file.Assume that a record is in the form of "$< \text{key}, \{\text{distance}^{(i)}, \text{y}^{(i)}\} >$" after segmentation and parsing.

a) Map

Adding the number 1 to the entered key is convenient for the Reduce process to accumulate, that is, directly output key-value pairs in the form of "$< \text{y}^{(i)}, 1 >$".

b) Shuffle

After all key-value pairs are sorted and merged, they will be output in the form of key-value pairs "$< y^{(i)}, < 1,1 \dots 1 > >$", and the results will be transmitted to the Reduce stage for processing.

c) Reduce

This process only needs to accumulate the records of the same key value, and then output to the file in the form of "$< y^{(i)},\ count^{(i)} >$".

Finally, read the output file and output the category with the largest sum value, which is the category of new sample.

C. The Specific Implementation of the Algorithm

The algorithm implementation consists of three parts: the first introduces the MapReduce workflow of calculating the

Euclidean distance; the second introduces the MapReduce workflow of counting labels; the third introduces the sequential MapReduce startup process.

1) Algorithm 1: MapReduce of calculating Euclidean distance

Function MapReduce-One(X[], data)

Input: X[] ← new sample feature array, data ← sample dataset

Output: "$< distance^{(i)},\ y^{(i)} >$"

Begin

a) for line in data

b) array ← Numpy.array(line.split(","))

c) array ← data preprocessing

d) x_train ← generate training matrix

e) label ← y_train

*f) distance ← sqrt(Numpy.sum((x_train - X) ** 2))*

g) output "$< distance^{(i)},\ y^{(i)} >$"

h) shuffle, sort

i) output "$< distance^{(i)},\ y\_train^{(i)} >$"

End MapReduce-One

2) Algorithm 2: MapReduce of counting labels

Function MapReduce-Two(data)

Input: data ← MapReduce-One Output

Output: $<y\_train^{(i)}, count^{(i)}>$

Begin

a) for line in data:

b) output "$< \mathrm{y\_train}^{(\mathrm{i})}, 1 >$"

c) shuffle, sort

d) count ← 0

e) for value in values

f) count ← count + value

g) output "$< distance^{(i)},\ count^{(i)} >$"

End MapReduce-Two

3) Algorithm 3: Call sequential MapReduce process

Function Train-Data(data, result, X[])

Input: data ← sample dataset, result ← output path, X[] ← new sample feature array，

Output：New sample label

Begin

a) start MapReduce-One

b) calculates Euclidean distance and sorts

c) output1

d) start MapReduce-Two

e) count labels

f) output2

g) read output2

h) output new sample label

End Train-Data

IV. Experiment

A. Environment Configuration

The experimental server is a Lenovo Y7000P notebook, which is configured with 1 physical CPU (Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, CPU with 6 cores and 12 threads), 16G memory, 1T solid state drive, 1 physical network card. The server is installed with Win10 Home Edition operating system, and VirtualBox software is used to create 3 virtual machines. Each virtual machine is configured with 1 core CPU, 2G memory, 20G hard disk, and 1 virtual network card. Each virtual machine is installed with Ubuntu 16.04 operating system and Hadoop 2.7.3 distributed computing platform to form a distributed cluster with 1 master node and 2 slave nodes, using Anaconda3, Python3.6 and VSCode as development environments.

B. Experimental Data Set

1) Introduction

The experimental dataset comes from the adult dataset in the UCI machine learning repository. The data set is a binary classification data set, which is mainly based on the census data to predict whether a person's income exceeds 50,000 dollars each year, in order to promote some products.

The dataset contains a total of 48,000 pieces of data, of which contains 32,000 pieces of training data and 16,000 pieces of test data. The dataset contains a total of 12 feature attributes, including 7 numerical features and 5 text features. Text feature values need to be converted into corresponding numerical forms during the training process. The description of the adult dataset is shown in Table I.

TABLE I: Adult Data Set Description

Feature	Type	Remark
age	Numeric	
work-class	Text	8 types
education-num	Numeric	
marital-status	Text	7 types
occupation	Text	14 types
relationship	Text	6 types
race	Text	5 types
sex	Text	2 types
capital-gain	Numeric	
capital-loss	Numeric	
hours-per-week	Numeric	
native-country	Text	41 types

2) Data preprocessing

The preprocessing process consists of two parts: converting the text type value into a numeric value type to participate in the calculation; normalizing the eigenvalues with a large slope in the data set. Since all the feature values of the adult dataset are the same dimension (within 100), the normalizing process can be omitted.

Taking race racial characteristic values as an example, race includes White, Asian and Pacific Islanders, American Indian Eskimos, Black, and others five categories in total, which are stored in a constant array. In the mapper function for calculating the Euclidean distance, when the race value in a row of sample data is read, the corresponding index value in the returned array is used as the characteristic value to participate in the operation. And the preprocessing process of other text feature values is the same as that of race.

C. Cluster Experimental Process

1) Verify correctness of algorithm improvement

Take 10,000 pieces of data in the training data set, and select 1000 pieces of data each test to train in a stand-alone machine or cluster environment, and then output the results of calculating the Euclidean distance to a file and compare

the data in two files. After 10 tests, the corresponding calculation results of all samples are the same.

Based on the above 10,000 training datasets, take 100 pieces of data in the test dataset to count labels results in the stand-alone machine and cluster environments, and then compared the output files. After 10 tests, the corresponding counting results of all samples are the same.

Combined with the above test process, the correctness of the improved algorithm parallelization can be verified.

2) Comparison of computing process between sequential and single MapReduce

Sequential is the same as the MapReduce workflow of a single MapReduce calculation of the Euclidean distance. The difference is that the statistical process of the former is also completed through the MapReduce workflow, while the statistical process of the latter is completed by reading the output file on a single machine.

Since the correctness of the algorithm has been verified, in order to test as much data as possible, all 48,000 pieces of data are used as training data to participate in the test. Starting from 6,000 pieces of data, each time adds 6,000 pieces of data. Test separately in two environments, and count the running time of the two. Taking the total amount of training data as the abscissa, the running time ratio λ (times) of the sequential and single as the ordinate. The corresponding scatter plot is drawn as shown in Fig. 2.

Experimental results show that the final time ratio stabilizes at about 1.16. Since the entire calculation process is mainly based on the calculation of Euclidean distance, the improvement in efficiency this time is not significant. In the case of a small training data set at the beginning of the experiment, the time impact of MapReduce's IO operation on the statistical process cannot be ignored, so the efficiency of a single MapReduce operation is slightly higher. With the expansion of the training data set, the operational advantages of sequential MapReduce gradually manifested.

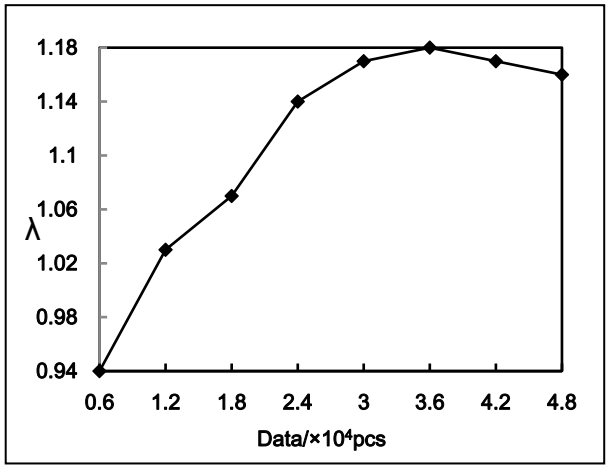

Fig. 2. Time ratio between sequential and single MapReduce training.

V. Conclusion

In order to further improve the computing efficiency of the KNN algorithm in a cluster environment, this paper is based on the Hadoop platform and proposes a sequential MapReduce computing framework to parallelize the KNN algorithm in a cluster environment. The experimental results show that the improved computational efficiency has a certain improvement compared with the previous one. In the actual application process, with the increase of the number of nodes and the improvement of cluster computing performance, the efficiency of the algorithm will be further improved. In the later research and learning process, we will try to combine other big data platforms and optimizing algorithms, and I believe it can further improve the algorithm's ability to process data.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Yaoshun Li conducted the research, analyzed the data and wrote the paper; Lizhi Liu guided the experiment process and participated in the revision of the paper; all authors had approved the final version.

References

[1] Z. Jiang, Z. Bian, and S. Wang, "Multi-view local linear KNN classification: Theoretical and experimental studies on image classification," *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 3, pp. 525–543, 2020.

[2] A. Li, "Exploration of handwritten digit recognition based on KNN algorithm," *Communication World*, vol. 27, no. 2, pp. 37-38, 2020.

[3] W. Xu, K. Guan, Q. Xun, and Y. Xu, "Improved K-means algorithm based on KNN algorithm," *Journal of Qingdao University of Science and Technology (Natural Science Edition)*, vol. 40, no. 5, pp. 107-118, 2019.

[4] Z. Wang, S. Liu, and Q. Luo, "KNN classification algorithm based on improved K-modes clustering," *Computer Engineering and Design*, vol. 40, no. 8, pp. 2228-2234, 2019.

[5] D. Liu, J. Zheng, and Z. Liu, "Research on parallelization of KNN algorithm based on CUDA," *Small Microcomputer System*, vol. 40, no. 6, pp. 1197-1202, 2019.

[6] W. Zhang, T. Liu, and M. Wu, "K-value adaptive KNN algorithm using ring filter," *Computer Engineering and Applications*, vol. 55, no. 23, pp. 45-85, 2019.

[7] Y. Ma, H. Zhao, and Y. Cui, "Improved KNN classification algorithm parallel processing based on Hadoop platform," *Journal of Changchun University of Technology*, vol. 39, no. 5, pp. 484-489, 2018.

[8] Z. Li and G. Huang, "Research on SVM_KNN classification algorithm based on Hadoop platform," *Computer Technology and Development*, vol. 26, no. 3, pp. 75-84, 2016.

[9] J. Zou and F. Li, "Distributed accurate fuzzy KNN classification algorithm under big data," *Computer Application Research*, vol. 36, no. 12, pp. 3701-3704, 2019.

[10] J. Wang, S. Yan, and Y. Gao, "Parallel ML-kNN algorithm based on MPI," *Journal of Zhengzhou University (Science Edition)*, vol. 50, no. 3, pp. 34-38, 2018.

[11] X. B. Zou, J. Wang, and M. Zhan, "KNN-ALS model recommendation algorithm under Spark platform," *Journal of Huaqiao University (Natural Science Edition)*, vol. 40, no. 2, pp. 264-268, 2019.

[12] L. Li, Y. Zhu, and Y. Song, "Spark-KNN parallel pattern recognition method for leakage current data," *Journal of System Simulation*, vol. 30, no. 4, pp. 1473-1481, 2018.

Yaoshun Li was born in Jingzhou, Hubei, China in May 1998. He graduated from Wuhan Institute of Technology with a bachelor's degree in computer science and engineering in 2020. Since 2020, he has continued to complete master's degree in the School of Artificial Intelligence, School of Computer Science and Engineering, Wuhan Institute of Technology, mainly engaged in research work in the direction of deep learning.

Lizhi Liu was born in Wuhan, Hubei, China in February 1973. He graduated from Huazhong University of Science and Technology with a master's degree in computer application technology. He is currently an associate professor at the School of Artificial Intelligence, School of Computer Science and Engineering, Wuhan Institute of Technology. He mainly studies cloud computing, big data, and machine learning.