Copyright ⓒ 2021 KSII

Parallel Algorithm of Improved FunkSVD Based on Spark

Xiaochen Yue[1], and Qicheng Liu[2*]
[1] School of Computer and Control Engineering, Yantai University,
Yantai 264000, Shandong, China
[e-mail: 17854256357@163.com]
[2] School of Computer and Control Engineering, Yantai University,
Yantai 264000, Shandong, China
[e-mail: ytliuqc@163.com]
*Corresponding author: Qicheng Liu

Received January 20, 2021; revised March 15, 2021; accepted April 17, 2021; published May 31, 2021

Abstract

In view of the low accuracy of the traditional FunkSVD algorithm, and in order to improve the computational efficiency of the algorithm, this paper proposes a parallel algorithm of improved FunkSVD based on Spark (SP-FD). Using RMSProp algorithm to improve the traditional FunkSVD algorithm. The improved FunkSVD algorithm can not only solve the problem of decreased accuracy caused by iterative oscillations but also alleviate the impact of data sparseness on the accuracy of the algorithm, thereby achieving the effect of improving the accuracy of the algorithm. And using the Spark big data computing framework to realize the parallelization of the improved algorithm, to use RDD for iterative calculation, and to store calculation data in the iterative process in distributed memory to speed up the iteration. The Cartesian product operation in the improved FunkSVD algorithm is divided into blocks to realize parallel calculation, thereby improving the calculation speed of the algorithm. Experiments on three standard data sets in terms of accuracy, execution time, and speedup show that the SP-FD algorithm not only improves the recommendation accuracy, shortens the calculation interval compared to the traditional FunkSVD and several other algorithms but also shows good parallel performance in a cluster environment with multiple nodes. The analysis of experimental results shows that the SP-FD algorithm improves the accuracy and parallel computing capability of the algorithm, which is better than the traditional FunkSVD algorithm.

Keywords: FunkSVD, RDD, RMSProp, Spark

This research was supported by the Shandong Provincial Natural Science Foundation of China (ZR2016FM42), the Shandong Province Key R&D Program Funded Project (2016GGX109004), the State Oceanic Administration's "13th Five-Year Plan" Marine Economy Innovation and Development Demonstration Key Project (YHC-ZB-P201701) and project supported by the National Natural Science Foundation of China (61702439).

http://doi.org/10.3837/tiis.2021.05.004 **ISSN : 1976-7277**

1. Introduction

Obviously, today is an era of data deluge, a large amount of data has been being continually generated at unprecedented scales. And there is no doubt that big data are now expanding rapidly in all fields of science and engineering. The potential of these massive amounts of data is unquestionably enormous, and makes researches in various fields require new approaches, innovative thinking and novel learning techniques to meet the challenges that big data presents [1]. In terms of recommendation algorithms, the growing amount of data on the platform is one of the challenges. As it is known, big data contains 4 Vs: volume, velocity, variety, and veracity. For volume, 4.4 zettabytes of data existed globally in 2013 and is predicted to increase to 44 zettabytes by 2020, as it more than doubles each year [2]. A large amount of data makes it more difficult for users to filter information and process data [3]. Therefore, collaborative filtering recommendation algorithm, as one of the information filtering technologies, is widely used in recommendation system [4,5,6]. Collaborative filtering recommendation algorithms are mainly divided into user-based collaborative filtering, item-based collaborative filtering, and model-based collaborative filtering [7]. Reference [8] developed a novel approach that incorporates a bipartite network into user-based collaborative filtering for enhancing the recommendation quality of new users. Reference [9] proposed a powerful Item-based Collaborative Memory Network (ICMN) for ICF, which bases on the architecture of Memory Networks. Reference [10] proposed a Bayesian model recommendation algorithm to recommend items by using similar user and item information.

With the increasing amount of data available today, some parallel computing frameworks and platforms for handing big data have gradually become popular, such as Hadoop and Spark [11], which speeds up the calculation of algorithms and reduces the difficulty of processing data. Reference [12] proposed an improved ABC algorithm based on Hadoop, so that the algorithm can efficiently process large data sets. Reference [13] developed a canopy algorithm and KMC parallel design under the Spark platform to improve efficiency.

For massive data, the traditional FunkSVD algorithm faces the problem of low accuracy [14]. In order to improve the precision of the FunkSVD algorithm, [15] proposed an improved fuzzy clustering FunkSVD algorithm, which reduces data sparseness and improves the accuracy of the algorithm. However, there is still an iterative oscillation problem. Reference [16] proposed a matrix decomposition structure and method based on FunkSVD, which provides a reliability value for each prediction in the matrix and improves the accuracy of the algorithm. However, the effect on sparse matrices is not very good. Reference [17] proposed a recommendation algorithm based on FunkSVD matrix decomposition and similarity matrix improves the accuracy of the algorithm, but it may cause iterative oscillation and fall into local optimal solution instead of global optimal. Reference [18] proposed an improved FunkSVD algorithm, which integrated user' s social relation information and project information, and improved the accuracy of the algorithm. However, data sparsity is an urgent problem.

With the continuous growth of data, collaborative filtering recommendation algorithms usually face the important problem of slow computation [19], so many researchers have proposed improved collaborative filtering algorithms based on Hadoop and Spark. Reference [20] proposed a big data analysis method based on Hadoop improved collaborative filtering recommendation algorithm, which improves the efficiency of collaborative filtering algorithm. Reference [21] implemented an improved ALS recommendation algorithm on the Spark distributed platform, which improved the computational efficiency of the algorithm. Reference [22] proposed a parallel CDL-I recommendation algorithm based on Spark to address the problem of low recommendation efficiency in the era of big data.

The traditional FunkSVD algorithm, as one of the collaborative filtering recommendation algorithms, also faces the problem of slow computation when facing big data. However, because Hadoop adopts the MapReduce framework and stores the intermediate results in HDFS, iterative computation is slow [23,24], which is not suitable for FunkSVD algorithms that require iteration. The core of Spark is memory computing, and storing data in distributed systems during computations improves the speed of iterative computation [25], which is ideally suitable for FunkSVD algorithm.

In summary, the goal of this paper is twofold. One is that this paper uses the RMSProp optimization algorithm to improve the traditional algorithm. Aiming at the low precision of the traditional FunkSVD algorithm, the existing method is roughly divided into two aspects. On the one hand, the accuracy of the algorithm is improved by alleviating data sparseness, on the other hand, the accuracy of the algorithm is improved by solving iterative oscillations to avoid local optimal solutions. The difference between SP-FD and the existing improved algorithm is that SP-FD algorithm can not only alleviate the impact of data sparseness on the accuracy of the algorithm but also solve the problem of iterative shock causing the accuracy of the algorithm to decrease, and improve the accuracy of the traditional algorithm from these two aspects. The other is that considering the problem that large data computation speed may be slow, this paper implements the parallelization of the improved FunkSVD algorithm in the framework of Spark to improve the algorithm' s computational speed. The highlights of this paper are made a summary as follows:

- Firstly, the paper introduces the traditional FunkSVD algorithm, RMSProp algorithm and a brief overview of the Spark big data framework.
- Then, propose an improved FunkSVD algorithm, and use the spark big data framework to parallelize the algorithm to get the SP-FD algorithm.
- Next, systematically analyze and compare the performance of SP-FD algorithm with other algorithms in terms of accuracy, execution time, and speedup.
- Finally, summarize the work of this paper.

The remainder of the paper is organized as follows. In Section 2, this paper introduces the traditional FunkSVD algorithm, RMSProp algorithm and Spark big data computing framework. Section 3 proposes a parallel algorithm of improved FunkSVD based on Spark, namely SP-FD. Section 4 carries on the experimental analysis, and compares the performance of SP-FD algorithm with other algorithms in terms of accuracy, execution time and speedup. Section 5 summarizes the work of this paper, and gives research trends.

2. Related Concepts

2.1 Traditional FunkSVD Algorithm

In order to solve the problem of slow computational efficiency of SVD algorithm, the traditional FunkSVD algorithm is proposed. FunkSVD solves the problem that missing values need to be filled in SVD algorithm. The principle is simple and the effect is good. Although the FunkSVD algorithm is simple in principle, it is indeed very extensive in practical applications and pushes the model-based recommendation algorithm to a new level.

As shown in (1), the FunkSVD algorithm decomposes the expectation matrix R into two low-rank matrices, namely the user matrix P and the item matrix Q, and maps them to a k-dimensional space, which corresponds to k hidden factors.

$$R_{m\times n} = P^{T}_{m\times k} Q_{k\times n} \tag{1}$$

When using linear regression to decompose the score matrix, the mean square error is used as the objective function, and the final P and Q optimal values are found by optimizing the objective function. However, in practice, in order to prevent over-fitting, a regularization term is usually added. For the existing scoring samples, the loss function is shown in (2).

$$e_{i,j}^2 = (r_{i,j} - \sum_{k=1}^{K} p_{i,k} q_{k,j})^2 + \lambda \sum_{k=1}^{K} (\| P \|^2 + \| Q \|^2) \tag{2}$$

Where λ is the coefficient of regularization.

The traditional FunkSVD algorithm uses the gradient descent method to optimize optimal value when solving optimal value of (2). The gradient descent method formula is shown in (3).

$$\theta = \theta - \alpha \bullet \nabla_\theta J(\theta) \tag{3}$$

Where θ is the parameter to be updated, α is the step size of the gradient descent iteration.

As shown in (3) and (4), obtain the derivation of P and Q respectively.

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj}) q_{kj} + \lambda p_{ik} = -2 e_{ij} q_{kj} + \lambda p_{ik} \tag{4}$$

$$\frac{\partial}{\partial q_{kj}} e_{ij}^2 = -2(r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj}) p_{ik} + \lambda q_{kj} = -2 e_{ij} p_{ik} + \lambda q_{kj} \tag{5}$$

According to (3), iteratively update P and Q, the formula is shown in (6) and (7).

$$p_{ik}^{'} = p_{ik} - \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + \alpha (2 e_{ij} q_{kj} - \lambda p_{ik}) \tag{6}$$

$$q_{kj}^{'} = q_{kj} - \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + \alpha (2 e_{ij} p_{ik} - \lambda q_{kj}) \tag{7}$$

Bring the optimal values of P and Q into the scoring formula, calculate the updated score, and get the score prediction. The score calculation is shown in (8).

$$T_i = \sum_{i=0, j=0}^{i=n, j=m} p_{ik}^{'} q_{kj}^{'} \tag{8}$$

2.2 RMSProp Algorithm

Deep learning has become the forefront of solving many practical problems. It relies on optimization algorithms to learn the parameters of input data. Therefore, optimization algorithms play a vital role in solving practical problems successfully [26]. The AdaGrad algorithm is a deep learning optimization algorithm, which is a method of selecting the learning rate according to the situation. The learning rate is adaptive, because the actual rate is determined according to the parameters. The parameters with high gradient will reduce the learning rate, while the parameters with small gradient will increase the learning rate. Therefore, it performs well on sparse data [27]. However, the AdaGrad algorithm has the problem of a sharp decline in the learning rate, and the RMSProp algorithm is proposed for this problem [28]. This algorithm changes the way Adagrad accumulates the gradient, and takes the gradient as the element squared exponentially weighted moving average. Specifically, given the hyper parameter $0 \leq \gamma < 1$, the RMSProp algorithm calculates the exponential decay average at time step $t > 0$, as shown in (9).

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2 \tag{9}$$

Where γ is the attenuation index.

As shown in (10), the RMSProp algorithm readjusts the learning rate of each element in the independent variable of the objective function according to the element operation, and then updates the independent variable.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \varepsilon}} g_t \tag{10}$$

Where g is the gradient of parameter θ_i at time t, η is the learning rate, and ε is a constant added to maintain numerical stability, such as 10^{-6}.

Because the state variable $E[g^2]$ of the RMSProp algorithm is an exponentially weighted moving average of the square term g_t^2, it can be regarded as the weighted average of the short-batch stochastic gradient squared term of the last 1/(1-γ) time step. In this way, the learning rate of each element of the independent variable will no longer decrease (or remain unchanged) during the iteration. Then the derivative of a certain dimension is relatively large, the exponential weighted average is large, and the derivative of a certain dimension is relatively small, the exponential weighted average is small. This ensures that the derivatives of each dimension are at the same order of magnitude, thus reducing the iterative oscillation of the gradient descent method [29]. In summary, the RMSProp algorithm improves the learning rate of the AdaGrad algorithm and inherits the characteristics of good effects on sparse data. It also reduces the iterative oscillation phenomenon of the gradient descent method.

2.3 Spark Parallel Computing Framework

Spark is a fault-tolerant memory distributed computing framework that is fast, universal, and very suitable for iterative computation [30]. Compared with Hadoop's MapReduce, Spark's computing speed is 10-100 times faster [31]. Spark supports Java, Python and Scala languages, which is very convenient and easy to use [32]. In order to solve the shortcomings of MapReduce in interactive and iterative computation, Spark introduced RDD (Resilient Distributed Data Set), the core component of Spark [33], which is based on the distributed processing of multiple nodes in the Directed Acyclic Graph (DAG). Make full use of memory resources to improve the efficiency of calculation.

In addition to the core component RDD, Spark also has a tool layer, a storage layer, and a resource scheduling layer. The specific structure of Spark is shown in **Fig. 1**.

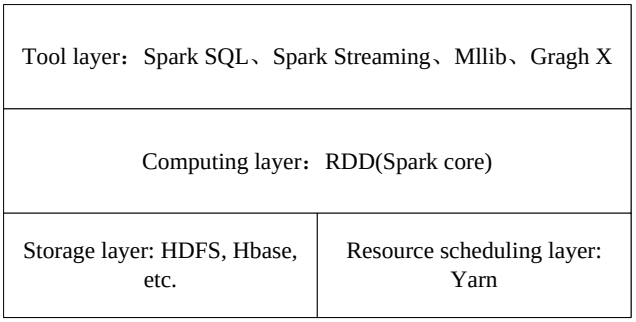

Fig. 1. Schematic diagram of Spark architecture

The tool layer includes four components: Spark SQL, Spark Streaming, MLlib, and Gragh X, which can process structured data, real-time computing, machine learning, and graph computing [34]. The storage layer contains data stored in distributed systems such as HDFS and Hbase. The resource scheduling layer Yarn, which can perform resource management and task scheduling, consists of three parts: ResourceManager, NodeManager, and ApplicationMaster.

3. Improved FunkSVD Algorithm Based on Spark—SP-FD

3.1 Improved FunkSVD Algorithm

The improved FunkSVD algorithm solves the problem of low accuracy of the traditional FunkSVD algorithm. The deep learning optimization algorithm RMSProp is used to improve the traditional algorithm, which not only alleviates the problem of data sparseness but also reduces the vibration phenomenon in the iterative process, thereby improving the accuracy of the algorithm. The steps to the improve FunkSVD algorithm are as follows.

a) Prepare user-item rating matrix R=[A_{11},...,A_{nm}]. Where A is the user's rating of the item.
b) According to (1), the score matrix R is decomposed into a user matrix P and an item matrix Q.
c) According to (2), solving the objective function e.
d) Use RMSProp algorithm to optimize the loss function in (2) to solve the optimal values of P and Q. The iteration steps are as follows:
   (a) Firstly, use (4) and (5) to get the derivation of P and Q respectively, as shown in (11) and (12).

$$\Delta p = -2e_{ij} q_{kj} + \lambda p_{ik} \tag{11}$$

$$\Delta q = -2e_{ij} p_{ik} + \lambda q_{kj} \tag{12}$$

   (b) Secondly, as shown in (13) and (14), calculate the average value of p and q exponential decay.

$$E[g_p^2]_t = \gamma E[g_p^2]_{t-1} + (1-\gamma)(\Delta p)^2 \tag{13}$$

$$E[g_q^2]_t = \gamma E[g_q^2]_{t-1} + (1-\gamma)(\Delta q)^2 \tag{14}$$

   (c) Finally, update the values of the P and Q matrix elements, as shown in (15) and (16).

$$p_t = p_{t-1} - \frac{\eta}{\sqrt{E[g_p^2]_t + \varepsilon}} \Delta p \tag{15}$$

$$q_t = q_{t-1} - \frac{\eta}{\sqrt{E[g_q^2]_t + \varepsilon}} \Delta q \tag{16}$$

e) Repeat steps c) to d) until the stop condition is met and optimal value is reached. The stopping condition is $e_{ij}^2 < \beta$, where β is a minimum value.
f) Bring the optimal values p_t and q_t into (8) to solve the updated scoring matrix.

Improved FunkSVD algorithm pseudo code as shown in Algorithm 1.

Algorithm 1 Improved FunkSVD algorithm

```
Input: Scoring matrix R[][],λ, γ, η, ε, β.
Output: new scoring matrix T[][].
  while R[i][j ]!= null do
    A[i][j] = R[i][j]
    R[i][j] →P[i][k]*Q[k][j]
    for each value do
        if A[i][j]>0 then
            e = A[i][j]
        for each value do
            e -= P[i][k]*Q[k][j]
            e += λ((P[i][k])²+(Q[k][j])²)
        end for
        while e²< β do
          for each value do
            Δp = -2eq+λp
            Δq = -2ep+λq
```

$E_p = \gamma E_p + (1-\gamma)(\Delta p)^2$
$E_q = \gamma E_q + (1-\gamma)(\Delta q)^2$
end
for each value do
$p_t = -(\eta/\sqrt{(E_p+\varepsilon)})(\Delta p)$
$q_t = -(\eta/\sqrt{(E_p+\varepsilon)})(\Delta q)$
end for
while p_t !=0&& q_t!=0 **do**
for each value do
$T_i = \sum p_t * q_t$
end for
for each value do
T[i][j] = T_i
end for
end while
end while

3.2 Parallelization of Improved FunkSVD Algorithm Based on Spark

In order to improve the computational efficiency of the improved FunkSVD algorithm, this paper implements the improved FunkSVD algorithm based on Spark, namely the SP-FD parallel algorithm, which is mainly divided into three parts: Data partition, Update iteration, and Score prediction. The specific workflow in Spark is shown in **Fig. 2**.

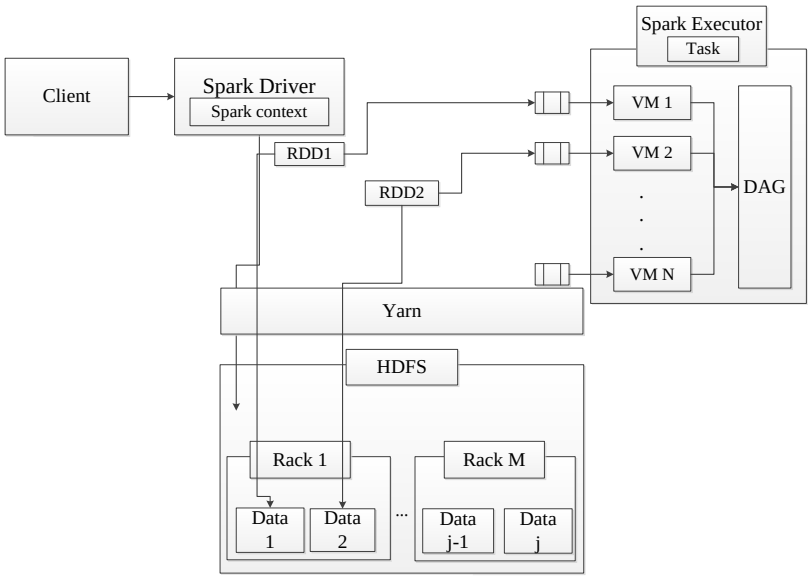

Fig. 2. Work flow chart of SP-FD algorithm in Spark

Among them, Client is a client that accepts tasks and submits tasks to Driver. Driver calls the main method of Spark program and starts SparkContext. SparkContext is the main interaction between the user and the Spark cluster. Yarn is responsible for resource management and scheduling of the cluster, and can read data from HDFS. Executor is the executor, which contains multiple virtual machines. Executor is used to execute multiple Tasks.

3.2.1 Data partition

First, the task is submitted to the Driver through the Client, and the Driver starts the SparkContext. In general, SparkContext is the total entry point for program operations. During

this process, Spark will automatically create job scheduling (DAGScheduler) and task scheduling (TaskScheduler). Then the Driver applies to Yarn for resources. Yarn accepts the application resource command, reads the rating data R from HDFS (step a)), including user (u), item (i) and score (r), loads it into RDD data format, and creates RDD. RDD contains two operations: transformation and action. Whenever a transformation factor is submitted to the RDD, a new RDD will be generated. In the new RDD, the transformation will be recorded as a functional object, which can provide a large number of operation methods, the most commonly used include map, filter, join, etc. Use randomSplit function data is divided into training set and testing set according to the ratio of 8:2. And use the map and mapPartitions functions in the transformation factor in the RDD to partition the P and Q matrix after R decomposition (step b)). The map function divides each row of data in the P and Q matrices, and the mapPartitions function puts the correlate u and the correlate i into different RDD partitions.

The pseudo code of the above algorithm is shown in Algorithm 2.

Algorithm 2. Data partition

Input: Scoring matrix R.
Output: RDD data.
Begin
makeRDD(R)
R1 = R.randomSplit(0.8,0.2)
R1→P∩Q
P_t = map(P)
Q_t = map(Q)
for each P_t, Q_t in P,Q **parallel do**
p(u,r) = mapPartitions(P_t)
q(i,r) = mapPartitions(Q_t)
out (p(u,r), q(i,r))
end for
End

3.2.2 Update iteration

During the iteration, the DAGScheduler module intervenes in calculations to calculate the dependencies between RDD, and the dependencies between RDD will form DAG. Each RDD is an abstraction of a data set, which contains multiple partitions. Another important operation in RDD is action. The action will perform the data manipulation part, usually with count, reduce, foreach and other methods. Calculate the data by calling the function in the action factor. Only the function of the action factor is called, the algorithm will be actually executed.

Update iteration implements the steps of the improved algorithm c) to e). First, use the tuple4Array function to find the optimal iterative value. Then call the collect function of the action factor to start the calculation iteration, and iteratively update each RDD partition in p(u,r) and q(i,r), and finally get the optimal value matrix p, q, the specific update rules are as shown in Algorithm 1. Where E_p, E_q are the initial exponential decay average values, and the initial value is 0.

The pseudo code of the above algorithm is shown in Algorithm 3.

Algorithm 3. Update iteration

Input: p(u,r), q(i,r), λ, γ, η, E_p, E_q.
Output: optimal value p, q
Begin
e= p(u,r)*q(i,r)+ λ(p(u,r)^2+q(i,r)^2)
tuple4Array(iteration)
for each p(u,r), q(i,r) **parallel do**

$\Delta p = -2eq(i,r)+\lambda p(u,r)$
$\Delta q = -2ep(u,r)+\lambda q(i,r)$
$E_p = \gamma E_p+(1-\gamma)(\Delta p)^2$
$E_q = \gamma E_q+(1-\gamma)(\Delta q)^2$
end for
for each t in p(u,r),q(i,r) **parallel do**
$p_t = -(\eta/\sqrt{(E_p+\varepsilon)})(\Delta p)$
$q_t = -(\eta/\sqrt{(E_p+\varepsilon)})(\Delta q)$
end for
$p = p_t$.collect()
$q = q_t$.collect()
out(p,q)
End

3.2.3 Score prediction

Use the p and q finally obtained in the previous step to implement step f). This process uses the Cartesian operation, which is time-consuming, so when calculating formula (8), use the repartionPairRDDBySize function to achieve block processing. This function has two parameters. The first parameter is the RDD that needs to be divided into blocks, that is, the optimal matrix p and q are divided into x and y blocks respectively, and the second parameter is the number of blocks defined by the user. Then the SparkContext will create DAGScheduler during the startup process, build the DAG graph, decompose it into Stage, and send the Taskset to the TaskScheduler, and finally the TaskScheduler will submit the Task to the Excutor, complete the calculation in multiple virtual machines, and finally get the updated score matrix T.

The above algorithm is shown in Algorithm 4.

Algorithm 4. Score prediction

Input: optimal value p,q, block n,score matrix R.
Output: new score matrix T
Begin
x_n = repartionPairRDDBySize(p, n)
y_n = repartionPairRDDBySize(q, n)
for each n **do**
for each w,z **in** x_n,y_n **parallel do**
$T_n = \sum p_{nw}q_{nz}$
end for
end for
i=R.numRows()
j=R.numCols()
for each value do
T[i][j] = T_n
end for
out(T[i][j])
End

4. Analysis of Experimental Results

This paper realizes the improvement of the traditional FunkSVD algorithm and the parallelization of the improved algorithm on the Spark. The Spark distributed experiment environment consists of a master node and two slave nodes. The hardware configuration of each machine is: CPU dual-core 3.2GHz, memory 4GB, and hard disk 500GB. The operating system is 64-bit Ubuntu16.04.

4.1 Experimental Data

The datasets used in this experiment are three publicly recommended data sets downloaded from the Internet: Jester, Movielens, and Wikipedia. The size of the datasets increases in turn, as shown in **Table 1**.

Jester dataset was completed by Ken Goldberg of the University of California, Berkeley. The data spans a period of more than 5 years, including approximately 2.3 million reviews up to Mar 2015. Reviews include jokes, users and ratings. The total number of users is 73,496 and the total number of jokes is 100. However, the total number of ratings available in the dataset is 2.3 million, which makes data sparser than Movielens dataset (31.2% sparsity).

Movielens is a publicly available dataset(ml-20M). The dataset consists of ratings of movies provided by users with corresponding user and movie IDs. There are 138,493 users and 27,278 movies with 20,000,263 ratings in the dataset. Had every user would have rated every movie total rating available should have been 3,777,812,054(i.e. 138,493×27,278); however only 20,000,263 ratings are available which means that not every user has rated every movie and the sparsity of dataset is 0.52%. This dataset resembles an actual scenario in E-commerce, where not every user explicitly or implicitly expresses preferences for every item.

Wikipedia is a collaborative encyclopedia written by its users. Wikipedia provides each user with a data dump for each article edited. The dataset can take the editing actions taken by the user as an implicit score, indicating that they care about the page for some reason, and allow us to use the dataset to make recommendations. The number of users is 5,583,724 and the number of articles is 4,936,761. The actual number of ratings available is 417,996,336. Therefore, the sparsity of the dataset is 0.0015%.

Table 1. Experimental data set

Data set	Users	Items	Rating	Density
Jester	73496	100	2300000	31.2%
Movielens	138493	27278	20000263	0.52%
Wikipedia	5583724	4936761	417996336	0.0015%

4.2 Evaluation Index

4.2.1 Accuracy

This paper uses Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), two evaluation indicators to measure the accuracy of the SP-FD algorithm.

The smaller the MAE value, the smaller the difference between the predicted value and the true value, and the higher the recommended accuracy. As shown in (17).

$$MAE = \frac{\sum_{i=1}^{n} | r_{ui}^{'} - r_{ui} |}{n} \tag{17}$$

Where $r_{ui}^{'}$ represents user u's predicted score for item i, and r_{ui} represents user u's actual score for item i.

The smaller the RMSE value, the higher the recommended accuracy rate, as shown in (18).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(s_{ui}^{'} - s_{ui})}{n}} \tag{18}$$

Where $s_{ui}^{'}$ represents user u's predicted score for item i, and s_{ui} represents user u's actual score for item i.

4.2.2 Execution time

This paper uses execution time (T) to evaluate the effectiveness of the SP-FD algorithm. Execution time determines the running speed of the algorithm. The shorter the execution time, the higher the effectiveness of the algorithm.

4.2.3 Speedup ratio

In order to verify the computational efficiency of the SP-FD parallel algorithm, this paper uses the speedup ratio to measure the performance and effect of the parallel algorithm. Speedup refers to the ratio of time consumed by the same task in a single-processor system and a parallel-processor system. As shown in (19).

$$S_p = \frac{T_l}{T_p} \tag{19}$$

Where p refers to the number of CPUs, T_l refers to the running time of the serial algorithm, and T_p refers to the running time of the parallel algorithm with P processors.

4.3 Experiment Analysis

The SP-FD algorithm has many parameters. Among them, K in (1) is the feature number, usually 10-100 [35], and the value of K in this algorithm is 30; the λ in (2) is the regularization parameter, and the algorithm in this paper is set to 0.2; the attenuation rate γ in (9) is usually set to 0.9, and the learning rate η in (10) is usually set to 0.0001 [36].

4.3.1 Number of iterations

The SP-FD algorithm also has an important parameter that affects the accuracy, that is, the number of iterations. The experiment confirms the suitable number of iterations of the SP-FD algorithm by changing the number of iterations. The larger the number of iterations, the more accurate. The number of experiment iterations is 10, 20, and 30. **Fig. 3** and **Fig. 4** compare the RMSE and MAE values of the SP-FD algorithm under different iteration times on different data sets.

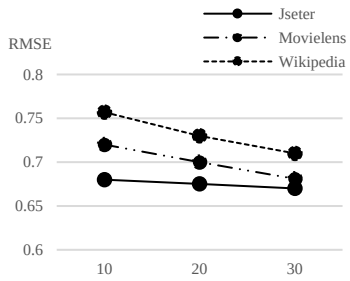

Fig. 3. RMSE value comparison chart

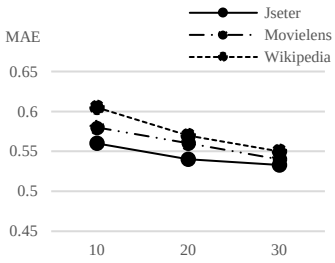

Fig. 4. MAE value comparison chart

It can be seen from **Fig. 3** and **Fig. 4** that as the number of iterations increases, the RMSE and MAE values both decrease, which shows that the SP-FD algorithm can alleviate the vibration phenomenon and improve the accuracy of the algorithm. And on the three data sets, the accuracy of the SP-FD algorithm reaches the highest when the number of iterations is 30. Therefore, it can be roughly considered that the number of iterations that the SP-FD algorithm is more suitable for is 30. In subsequent experiments, the number of iterations is 30.

4.3.2 Accuracy comparison

By comparing the accuracy of SP-FD algorithm in stand-alone environment with traditional FunkSVD algorithm, the improved algorithm proposed in [16] and the improved algorithm proposed in [17] in different data sets, this paper shows that SP-FD algorithm is superior to other algorithms. **Fig. 5** and **Fig. 6** respectively show the comparison of the RMSE and MAE values between the SP-FD algorithm and other algorithms.

Among them, reference [16] proposes a method to assign a reliability value to each prediction and, by extension, to each recommendation. The more suitable a reliability is, the better accuracy results will provide when applied. The specific operating mechanism of the algorithm: first, starting from the original scoring matrix, use the FunkSVD algorithm to perform matrix decomposition to obtain the scoring prediction matrix. Afterwards, the cross-validation method is applied to rank the prediction score matrix, and the error prediction matrix can be obtained. (The article refers to the error of the rating as known reliability, which represents the numerical deviation between prediction and reality. The greater the deviation, the worse the accuracy of the prediction.) Finally, the error prediction matrix is decomposed by the FunkSVD algorithm to obtain the reliability, and realize the recommendation.

The recommendation algorithm based on FunkSVD matrix decomposition and similarity matrix mentioned in [17] combines similarity calculation with FunkSVD algorithm, which effectively alleviates the impact of data sparseness on the accuracy of the algorithm. The specific operating mechanism of the algorithm: first calculate the similarity matrix between users and items. Then, the two similarity matrices are weighted and combined using score data between users to generate a comprehensive similarity matrix. Finally, the FunkSVD algorithm is used to perform matrix decomposition to realize the recommendation.

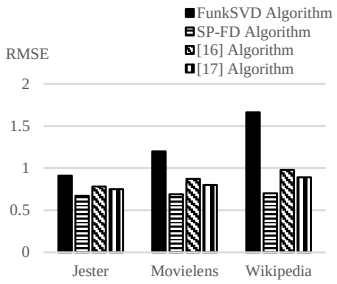

Fig. 5. Comparison of RMSE value between SP-FD algorithm and other algorithms

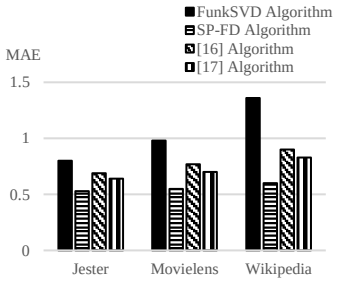

Fig. 6. Comparison of MAE value between SP-FD algorithm and other algorithms

Combined with **Fig. 5** and **Fig. 6**, it can be seen that the accuracy of the SP-FD algorithm on the three data sets is higher than other algorithms, and as the sparsity of the data set increases, compared with the other three algorithms, the accuracy rate of SP-FD algorithm has undergone minor changes, and it can be considered that the SP-FD algorithm performs better than other algorithms on sparse data.

4.3.3 Execution time comparison

In order to compare the effectiveness of the SP-FD algorithm and other algorithms, this article runs the SF-PD algorithm and other algorithms on different datasets, and compares the execution time of different algorithms on different data sets. The results are shown in **Fig. 7**.

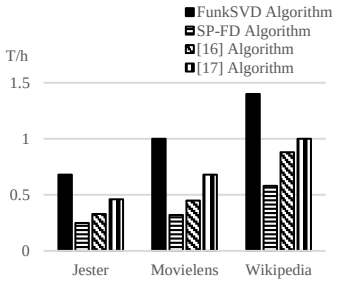

Fig. 7. Comparison of time frequency of different algorithms

As can be seen from **Fig. 7**, with the increase of data sets' size, the execution time of the four algorithms increases, but the execution time of the SP-FD algorithm is significantly less than the other three algorithms. This is because the SF-FD algorithm not only reduces the iteration time, but also reduces the time to calculate the Cartesian, speeds up the running time of the algorithm, and improves the effectiveness of the algorithm. It is significantly better than the other three algorithms in terms of effectiveness.

4.3.4 Speedup comparison

In order to test the parallelism of the SP-FD algorithm, this paper conducts experiments on three data sets on 1, 2 and 3 nodes. The acceleration of different data sets on different nodes is shown in **Fig. 8**.

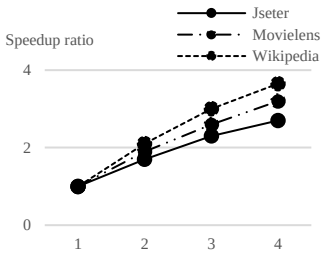

Fig. 8. Speedup of SP-FD algorithm for different nodes

It can be seen from **Fig. 8** that as the number of nodes increases, the speed-up ratio of the SP-FD algorithm gradually increases. This is because the serial time remains unchanged while the parallel calculation time decreases, which increases the speed-up ratio. As the data sets' size increases, the speedup ratio increases, and the parallel effect of the algorithm increases. This is because the increase in serial time is much greater than the increase in parallel time, which increases the speedup.

In summary, combined with the analysis and experimental results, this paper compares the accuracy and time frequency of the SP-FD algorithm with the traditional FunkSVD algorithm, the improved algorithm in [16] and the improved algorithm in [17]. The SP-FD algorithm has improved accuracy and execution time, which is better than the other three algorithms. In terms of parallelism, the SP-FD algorithm also has a good speedup.

5. Conclusion

Aiming at the problem of low accuracy of the traditional FunkSVD algorithm, this paper proposes a parallel algorithm SP-FD based on Spark. Through experimental comparison, it is found that the algorithm improves the accuracy of the algorithm, reduces the execution time, and improves the effectiveness of the algorithm. Through the Spark big data computing framework, the algorithm's parallel computing capability is improved. Compared in many aspects, the IF-BS algorithm proposed in this paper is superior to other algorithms.

However, in actual applications, the data is in units of TB. In future work, we will consider using a larger level of data set to verify the accuracy, effectiveness and robustness of the algorithm in order to apply the algorithm to more practical application areas.

Acknowledgement

Thanks to the National Natural Science Foundation of China (61702439), the State Oceanic Administration's "Thirteenth Five-Year" Marine Economy Innovation and Development Demonstration Key Project (YHC-ZB-P201701), Shandong Provincial Natural Science Foundation of China (ZR2016FM42) and Shandong Province Key R&D Program Funded Project (2016GGX109004) for supporting this project.

References

[1] Qiu J, Wu Q, Ding G, Xu Y and Feng S, "A survey of machine learning for big data processing," *EURASIP Journal on Advances in Signal Processing*, vol. 1, 2016, Art. no 67. Article (CrossRef Link)

[2] Zwolenski M and Weatherill L, "The Digital Universe Rich Data and the Increasing Value of the Internet of Things," *Journal of Telecommunications and the Digital Economy*, vol.2, no. 3, pp.47, 2014. Article (CrossRef Link)

[3] J. Chen, K. Li, Z. Tang, K. Bilal and K. Li, "A Parallel Patient Treatment Time Prediction Algorithm and Its Applications in Hospital Queuing-Recommendation in a Big Data Environment," *IEEE Access*, vol. 4, pp. 1767-1783, 2016. Article (CrossRef Link)

[4] Wang Dawei, Yih Y and Ventresca M, "Improving Neighbor-Based Collaborative Filtering by Using a Hybrid Similarity Measurement," *Expert Systems with Applications*, vol.160, pp. 113651, 2020. Article (CrossRef Link)

[5] G. Liu and X. Wu, "Using Collaborative Filtering Algorithms Combined with Doc2Vec for Movie Recommendation," in *Proc. of 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Chengdu, China, 2019. Article (CrossRef Link).

[6] D. Chae, J. A. Shin and S. Kim, "Collaborative Adversarial Autoencoders: An Effective Collaborative Filtering Model Under the GAN Framework," *IEEE Access*, vol. 7, pp. 37650-37663, 2019. Article (CrossRef Link)

[7] Y. Gao and L. Ran, "Collaborative Filtering Recommendation Algorithm for Heterogeneous Data Mining in the Internet of Things," *IEEE Access*, vol. 7, pp. 123583-123591, 2019. Article (CrossRef Link)

[8] Z. Zhang, M. Dong, K. Ota and Y. Kudo, "Alleviating New User Cold-Start in User-Based Collaborative Filtering via Bipartite Network," *IEEE Transactions on Computational Social Systems*, vol. 7, pp. 672-685, 2020. Article (CrossRef Link)

[9] D. Seng, G. Chen and Q. Zhang, "Item-Based Collaborative Memory Networks for Recommendation," *IEEE Access*, vol. 8, pp. 213027-213037, 2020. Article (CrossRef Link)

[10] P. Valdiviezo-Diaz, F. Ortega, E. Cobos and R. Lara-Cabrera, "A Collaborative Filtering Approach Based on Naïve Bayes Classifier," *IEEE Access*, vol. 7, pp. 108581-108592, 2019. Article (CrossRef Link)

[11] Du Jiaying, Duan Longzhen and Duan Wenying, "Parallel implementation of improved K-means algorithm based on Spark," *Application Research of Computers*, vol. 37, no. 2, pp. 434-436+497, 2020. Article (CrossRef Link)

[12] Wu Ying, Li Xiaoling and Tang Jinglei, "IoT big data feature selection method based on particle filtering under Hadoop platform combined with improved ABC algorithm," *Application Research of Computers*, vol. 36, no. 11, pp. 3297-3301, 2019.

[13] Wang Hui, Zhou Chengdong and Li Leixiao, "Design and Application of a Text Clustering Algorithm Based on Parallelized K-Means Clustering," *Revued' Intelligence Artificielle*, vol. 33, no. 6, pp. 453-460, 2019. Article (CrossRef Link)

[14] Bipul Kumar, "A novel latent factor model for recommender system," *JISTEM: Journal of Information Systems and Technology Management*, vol. 13, no. 3, pp. 497-514, 2016. Article (CrossRef Link)

[15] D. Yao and X. Deng, "Teaching Teacher Recommendation Method Based on Fuzzy Clustering and Latent Factor Model," *IEEE Access*, vol. 8, pp. 210868-210885, 2020. Article (CrossRef Link)

[16] B Zhu, F Ortega and J Bobadilla, "Assigning reliability values to recommendations using matrix factorization," *Journal of Computational Science*, vol. 26, pp. 165-177, 2018. Article (CrossRef Link)

[17] Wang Yun, Ni Jing and Ma Gang, "Recommendation algorithm based on FunkSVD matrix decomposition and similarity matrix," *Computer Applications and Software*, vol. 36, no. 12, pp. 245-250, 2019.

[18] Deng Xiuqin, Liu Taiheng and Li Wenzhou, "A latent factor model of fusing social regularization term and item regularization term," *Physica A: Statistical Mechanics and its Applications*, vol. 525, pp. 1330-1342, 2019. Article (CrossRef Link)
[19] Zhu Li, Li Heng and Feng Yuxuan, "Research on big data mining based on improved parallel collaborative filtering algorithm," *Cluster Computing*, vol. 22, no. 2, pp. 3595-3604, 2019. Article (CrossRef Link)
[20] Yin Nan, "A Big Data Analysis Method Based on Modified Collaborative Filtering Recommendation Algorithms," *Open Physics*, vol. 17, no. 1, pp. 966-974, 2019. Article (CrossRef Link)
[21] Tao Jinhong, Gan Jianhou and Wrn Bin, "Collaborative Filtering Recommendation Algorithm based on Spark," *International Journal of Performability Engineering*, vol. 15, no. 3, pp. 930-938, 2019. Article (CrossRef Link)
[22] Yang Fan, Wang H and Fu J, "Improvement of recommendation algorithm based on Collaborative Deep Learning and its Parallelization on Spark – ScienceDirect," Journal of Parallel and Distributed Computing, to be published.
[23] B. Liu, S. He, D. He, Y. Zhang and M. Guizani, "A Spark-Based Parallel Fuzzy c -Means Segmentation Algorithm for Agricultural Image Big Data," *IEEE Access*, vol. 7, pp. 42169-42180, 2019. Article (CrossRef Link)
[24] K. Binzani and J. S. Yoo, Seattle, "Spark-based Spatial Association Mining," in *Proc. of 2018 IEEE International Conference on Big Data (Big Data)*, WA, USA, pp. 5300-5301, 2018. Article (CrossRef Link)
[25] G. Xian, "Parallel Machine Learning Algorithm Using Fine-Grained-Mode Spark on a Mesos Big Data Cloud Computing Software Framework for Mobile Robotic Intelligent Fault Recognition," *IEEE Access*, vol. 8, pp. 131885-131900, 2020.
[26] Tong Weiguo, Li Minxia and Zhang Yike, "Research on Deep Learning Optimization Algorithm," *Computer Science*, vol. 45, no. S2, pp. 155-159, 2018. Article (CrossRef Link)
[27] John C Duchi, Elad Hazan and Yoram Singer, "Adaptive Subgradient Methods Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121-2159, 2011.
[28] Liu Ran, Liu Yu and Gu Jinguang, "AdaNet improvement based on adaptive learning rate optimization," *Journal of Computer Applications*, pp. 1-7, 2020.
[29] X. Wu, H. Xu, X. Wei, Q. Wu, W. Zhang and X. Han, "Damage Identification of Low Emissivity Coating Based on Convolution Neural Network," *IEEE Access*, vol. 8, pp. 156792-156800, 2020. Article (CrossRef Link)
[30] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, J. M. Benítez and F. Herrera, "Nearest Neighbor Classification for High-Speed Big Data Streams Using Spark," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 10, pp. 2727-2739, Oct. 2017. Article (CrossRef Link).
[31] J. Lee, B. Kim and J. Chung, "Time Estimation and Resource Minimization Scheme for Apache Spark and Hadoop Big Data Systems with Failures," *IEEE Access*, vol. 7, pp. 9658-9666, 2019. Article (CrossRef Link)
[32] Feiran Wang, Yiping Wen and Tianhang Guo, "Collaborative filtering and association rule mining‐based market basket recommendation on spark," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 7, 2020. Article (CrossRef Link)
[33] M Zaharia, M Chowdhury and T Das, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proc. of the 9th USENIX conference on Networked Systems Design and Implementation*, 2012.
[34] T. Hong-Peng and C. En-Jie, "Research on Collaborative Filtering Algorithm Based on Spark Platform," in *Proc. of 2017 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII)*, Wuhan, China, pp. 33-36, 2017. Article (CrossRef Link)

[35] T. Xie, S. Li and B. Sun, "Hyperspectral Images Denoising via Nonconvex Regularized Low-Rank and Sparse Matrix Decomposition," *IEEE Transactions on Image Processing*, vol. 29, pp. 44-56, 2019. Article (CrossRef Link)
[36] F. Zou, L. Shen, Z. Jie, W. Zhang and W. Liu, "A Sufficient Condition for Convergences of Adam and RMSProp," in *Proc. of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, pp. 11119-11127, 2019. Article (CrossRef Link)

Yue Xiaochen, born in 1997. M.S. Her main research interests focus on Data mining and parallel computing.

Liu Qicheng, born in 1970. phD, professor. His research interests include cloud computing, big data, multi-agent systems, data mining, etc.