Distributed Design and Implementation of SVD++ Algorithm for E-commerce Personalized Recommender System

Jian Cao[1(✉)], Hengkui Hu[1], Tianyan Luo[2], Jia Wang[2], May Huang[2], Karl Wang[2], Zhonghai Wu[1], and Xing Zhang[1]

[1] School of Software and Microelectronics, Peking University, Beijing 102600, China
caojian@ss.pku.edu.cn
[2] Department of Electrical and Computer Engineering, International Technological University, San Jose, CA 95113, USA

Abstract. Recommender systems can facilitate people to get effective information from the massive data, and it is the hot research currently in data mining. SVD++ is a kind of effective single model recommendation algorithm, which is based on the matrix decomposition combined with the neighborhood model. On the Spark, using the Stochastic Gradient Descent, this paper realized the distributed SVD++ algorithm through the Scala, deployed and applied the algorithm into an actual recommendation product for testing. The testing results represent that the distributed SVD++ algorithm succeeded in solving problems of terabytes of data processing in the e-commerce recommendation and the sparse data of user-item matrix, enhancing the quality in personalized commodity recommendation.

Keywords: Recommender system · Matrix decomposition · SVD++ · Distributed computation · E-commerce

1 Introduction

The basic task of recommender system is to contact users and items, to solve the problem of information overload. Recommender systems provide users with personalized suggestions about products or services to help customers easily find their favorite commodity in the tens of thousands of items [1, 2]. Now recommender system refers to the personalized recommendation system [3], which depends on the user's behavior data, generally existed in the website as an application. In nowadays e-commerce sites, the amount of data is very large, their user item matrixes has significant data sparseness problem, that is only a few users have history behavior or hit rating on a few commodities [4]. Score matrix's sparse problems will seriously affect the performance of the recommendation algorithm. The matrix decomposition method can be used to do dimension reduction to user items matrixes, for the original matrix, finding the best low dimensional approximation which can solve the problem of data sparsity in a certain sense [5].

X. Zhang et al. (Eds.): ESTC 2015, CCIS 572, pp. 30–44, 2015.
DOI: 10.1007/978-981-10-0421-6_4

This paper has studied using SVD++ algorithm to improve recommendation quality of the personalized items in the recommender system, namely recalling the potential interested items to customers as much as possible, and impressed to the users in a reasonable order. SVD++ algorithm, proposed by Yehuda Koren, a matrix decomposition algorithm combined with neighborhood model [3], is an effective single recommendation algorithm. It can at the same time use the explicit and implicit user feedback data, through machine learning method, convert matrix decomposition problem to the optimization of machine learning problem, thus solving it more effectively. Todays in large e-commerce companies the amount of data has been achieved the level of TB, even PB. In order to apply SVD++ algorithm to recommendation problems of large-scale data sets, this paper tries to realize the distributed computing of the SVD++ algorithm.

2 SVD++ Algorithm and Evaluation Method

2.1 SVD++ Algorithm

SVD++ Algorithm adds history score item to the Latent Factor Model. Bias Latent Factor Model's formula is as follows [6]:

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u^T \cdot q_i \tag{1}$$

In the formula $\hat{r}_{ui}$ represents the prediction score of user u's interest in item i, the greater score means the better effect; μ represents the global average; bu and bi, respectively represent user bias and item bias; pu and qi represent user matrix and commodity matrix respectively.

Item-based Collaborative Filtering can be rewritten as follows:

$$\hat{r}_{ui} = \frac{1}{\sqrt{|N(u)|}} \sum_{j \in N(u)} w_{ij} \tag{2}$$

In the formula, wij is no longer a similarity matrix of the item, but a learning parameters to complete, N(u) represents a collection of items that users like.

So SVD++ Algorithm's formula can be represent as:

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u^T \cdot q_i + \frac{1}{\sqrt{|N(u)|}} x_i^T \sum_{j \in N(u)} y_j \tag{3}$$

xi and yj are two dimensional vectors of F. Here to substitute wij with xiTyj.

Koren puts forward that in order not to increase too much parameter that cause over-fitting, x = q can be made, and eventually SVD++ model is obtained [7]:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i(p_u^T + \frac{1}{\sqrt{|N(u)|}} x_i^T \sum_{j \in N(u)} y_j) \tag{4}$$

The above parameters are with the following meaning:

(1) μ The global average of all the records of score in the training set. This represents the influence of site itself to user ratings.
(2) bu The user bias. This represents the attribute factors part of the coring habit that only has the relationship with user.
(3) bi Item bias. This one represents that part of the scores of items that only has the relationship with items.

By computing the partial derivative of the loss function according to various parameters, the iterative formula can also be deduced. it is needed to optimize parameters as follows [8]:

The optimized parameters used in this article are characteristic dimensions - rank, Number of Iterations - numIter, step length - gamma1, step length - gamma2, regular terms - lambda1, regular terms - lambda2, learning rate - slowRate.

2.2 Algorithm Evaluation Method

There are mainly two kinds of experimental methods to evaluate the recommendation system, respectively are offline evaluating and online evaluation. This paper has used the online experiment method, specifically the A/B testing [7].

A/B testing is a commonly used experimental method of online evaluation algorithm [9]. First of all, by certain rules, users are separated into several groups randomly, and are applied to different algorithms to different groups of users. Through counting different evaluating indicators of different groups of users to compare different algorithms, such as counting user hits of different groups, compare the performance of the algorithm by the degree of the relative size of hits. Figure 1 is a simple diagram of the A/B testing system [7].

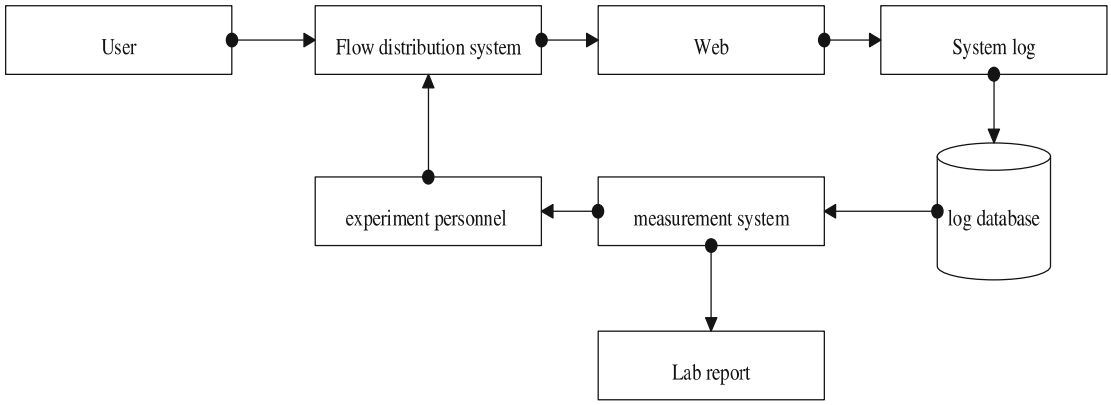

Fig. 1. A/B-test system

3 SVD++'s Distributed Design and Implementation on Spark

3.1 Spark Distributed Computing Platform

The Spark is a scalable data analysis platform based on In - Memory Computing, which has more performance advantages than Hadoop cluster storage method [10]. Spark uses the Scala language and provides a single data processing environment. Spark's features are showed on Resilient Distributed Datasets. It is a fault-tolerant, parallel data structure, which allows users to explicitly store the data into disk and memory, and to control the data partition. Spark GraphX is a distributed figure processing framework. Based on Spark platform, it provides concise, easy-to-use and colorful interfaces for graph calculation and graph mining, bringing great convenience to user demands for distributed figure processing [11].

3.2 Stochastic Gradient Descent Optimization Method

Gradient Descent is a common method of risk-minimizing function and loss function [12]. Stochastic Gradient Descent (SGD) is an iterative solution. Stochastic gradient descent is iteratively updated one time by every sample. In a large sample size such as hundreds of thousands of cases, it may only need tens of thousands or only thousands of samples to iterate the parameter to the optimal solution [13]. For the batch gradient descent, an iteration need to use tens of thousands of training samples, and an iteration may not the best, the 10-time iteration will need to traverse the training sample 10 times [14]. SGD, however, has a problem with it - the noise, which makes the SGD is not in the direction of the overall optimization on each iteration [15].

In this paper, the stochastic gradient descent method is to separate the user and commodity, fix one, update another individually. Because of the large training data set used in this article, which takes up more memory, separately update, can only put half of the data in the memory to do iteration every time, so as to reduce the computational complexity of each iteration. Tests represent that this will cause that the parameter update convergence speed is higher than the convergence speed of putting all the data of user and items into memory at the same time.

3.3 Distributed Design and Implementation Process of the SVD++ Algorithm

In this paper, Spark GraphX graph computing framework of the Spark platform is used to carry out distributed design and implementation of the SVD++ algorithm.

The key of distributed design is how to divide the data into different blocks, so that each block can be independent, and then different data blocks are distributed to different computing nodes. Variables can be divided into two categories, one is related to the user, one is related to the item, and SGD is used to update the two kinds of variables. User ratings according to the user and the item were divided into blocks, each block must have the required scores to updated the corresponding variables, while the users and the items is sent to different pieces of the calculation, and thus effectively complete distributed computing of the SVD++ algorithm.

To sum up, distributed implementation of the SVD++ algorithm is to assign the users and the items of a latent factor model to a cluster of hundreds of machines, and then use user ratings to optimize the model.

The realization of the algorithm steps are as follows:

Step 1: read the log file data, complete the digitalizing of user's and commodity's name, initialize vector at the same time.

Step 2: calculate the mean global miu: through the whole data set, add the score matrix and calculate the number of scoring at the same time; the former value can be divided by the latter to obtain miu.

Step 3: construct the graph model.

Step 4: calculate parameter values.

Calculate the mean score of each user and each commodity, namely each user's bias and commodity bias.

The history of the user behavior (i.e., read or add to cart or buy items), in order to calculate 1/|N|.

Step 5: training model.

Training user node function, is used in fixing commodity factor and training the user factor;

Training commodity node function is used in fixing user factor and training commodity factor.

Step 6: the specific training process.

Fix q, update p, namely fix commodity factor, and update user factor and bias.

First, calculate Pu + |N(u)| ^ (0.5) * sum (y) of each user;

And then update the user factor and bias.

Fix p, update q, namely fix user factor, update commodity factor, neighborhood grade y and commodity bias.

(1) First, calculate Pu + |N(u)| ^ (0.5) * sum (y) of each user;
(2) And then update the commodity factor, the neighborhood grade y and commodity bias.
(3) Remember to update step size after every training session.

4 Experimental Design and Test Analysis

4.1 Experimental Design and Application Deployment

The recommendation strategy this article used is to transfer recommend problems into score predict problems, then using SVD++ algorithm to predict ratings for items, and according to the grading sorting to make recommendation for users. Figure 2 is the flow chart of SVD++ algorithm applied for e-commerce recommender system.

Data Pre-processing. Data sources contain four parts of users' behavior, namely, the user's browsing data, attention or collection data, data added to cart and the purchase order. After logging out the original data from the website, the preliminary filter treatment has been carried on: to filter out enterprise users, and only preserve personal

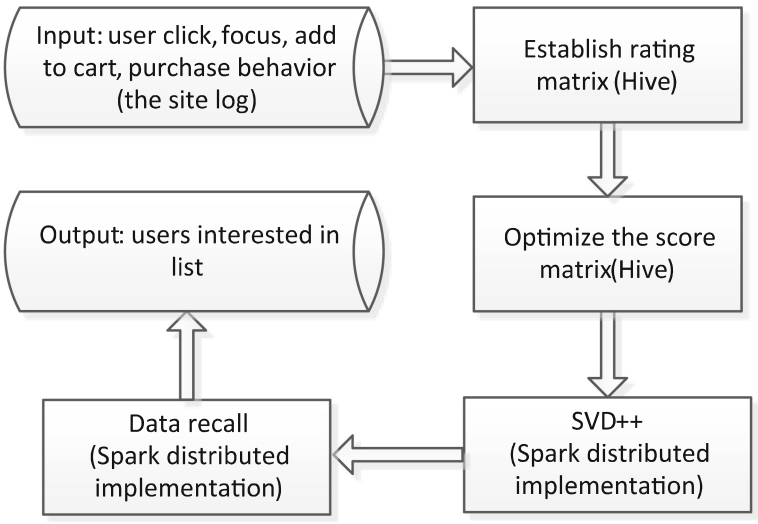

Fig. 2. SVD++ algorithm applied for e-commerce recommender system flow chart

user data; to filter out pates, failure Stock Keeping Unit (SKU) or invalid SKU caused by other reason; Due to the Standard Product Unit (SPU) granularity is more in line with the recommended logical meaning and more relevant to the user's interest model, so commodities SKU has been transferred into the SPU.

Design and Optimization of Score Matrix. The data needed by SVD++ algorithm is the user's commodity score matrix. The data which extracted from web, although has done some business processing, but still need to annotate them in the right way and build up the score matrix, so the algorithm can be used normally. The following is the elaboration to the electronic commerce recommender system's grading design rules of the matrix and its optimization method.

The user feedback data for items usually have five typical types: browse, search, click, attention/collect and add to cart, buy, and explicit feedbacks such as rating or review. These behavior data reflects that the weight of user's interest in the items is different and substantially increased in turn. From SVD++ algorithm is derived from the hidden factor model, it can be known that the main mining target of SVD++ algorithm is the user's potential interest. It mainly uses implicit feedback behavior of users, and the actual users' explicit feedback behavior for items usually is very limited, for example, the majority of users have not the habit of marking comment for the items. So this paper mainly adopts the four kinds of user's recessive feedback behavior to items, including browse, attention, add to cart and buy.

Considering the differences of the weight of user's interest from the different feedback behavior from users for items, this paper takes a 5-point system: the user views within 3 times (containing) is 2.6 points, browses more than five times (not included) or browses the 3 to 4 times at the same time and there is concern or collection behavior is 3.0 points, the add to cart behavior is 3.5 points, an order behavior is 4.5 points, more than twice (containing) order is 5.0 points.

The time factor for further optimization can also be considered.

The Deployment of Testing Environment. For a total of 70 machines, each machine memory is 56 G. The number of CPU cores is 28 and disk total capacity is 2 PB. The version number for the distributed platforms is hadoop-2.2.0, spark-1.2.0.

The recommended products of the experiment choose a column named "guess you like". The column can utmost reflects the characteristics of personalized recommendation, because the scene of this product is the weakest. Therefore experiments will be located on the recommended products, and it can more reflect the effect of SVD++ algorithm in personalized recommended goods.

Experiments involve the items which cover all category of the site, number of users is on tens-of-millions level, commodity number is on hundreds-of-millions level, and the data need to be dealed with in the experiment is on the TB magnitude.

4.2 The Test Results

This section uses A/B testing method for doing a synchronous contrast experiment on line testing to SVD++ algorithm. The online benchmark experiment is based on the Item-based CF recommendation algorithm. Online recommendation report usually adopts evaluation index to measure the effect of the various experimental methods of different recommender system, which mainly include request, display, click, orders and price. Request quantity represents the user traffic to your website, and display quantity is the amount of website commodity exposure under the user request. Hits is the user's clicks on items, quantity of orders is the items' amount users bought, and the price is unit price for the item.

The main parameters in this article are Request Conversion Rate (RCVR), Conversion Rate refers to the order quantity divided by the amount of requests, Gross Merchandise Volume (GMV) is the product of sales price and sales quantity.

Coverage:

$$\text{Coverage} = \text{Display Quantity}/\text{Request Quantity}. \tag{5}$$

Click Through Rate (CTR):

$$\text{CTR} = \text{Hits}/\text{Display Quantity}. \tag{6}$$

Click Conversion Rate (CCVR):

$$\text{CCVR} = \text{Order Quantity}/\text{Hits}. \tag{7}$$

Impression Conversion Rate (ICVR):

$$\text{Display Conversion Rate} = \text{Order Quantity}/\text{Display Quantity}. \tag{8}$$

Request Conversion Rate (RCVR):

$$\text{RCVR} = \text{Order Quantity}/\text{Request Quantity}. \tag{9}$$

Gross Merchandise Volume (GMV):

$$\text{Sales Revenue} = \text{Sale Price} * \text{Sales Quantity}. \tag{10}$$

Statistics Indicators of Test Evaluation. The Following are a statistics chart extracted from online A/B testing experiment. The time interval of experiment statistical data is from February 19, 2015 to March 1, 2015, totals 11 days. In the 11 days, the request, display, click, order and average data of this benchmark model Item-based CF and test model SVD++ are shown in Table 1.

Table 1. Business statistics of benchmark model Item-based CF and test model SVD++

A total of 11 days	Item-based collaborative filtering	SVD++
Request	62233152	24021521
Display	2790711	2136821
Click	32080	44755
Order	45	53
Average price	1647.08	3429.95

Table 2. Coverage

Date	Item-based CF	SVD++
20150219	0.04541036	0.09699144
20150220	0.04577684	0.09653428
20150221	0.04637238	0.09396935
20150222	0.04695301	0.09125606
20150223	0.04749833	0.09165352
20150224	0.04562790	0.08823611
20150225	0.04374205	0.08386233
20150226	0.04204751	0.08046467
20150227	0.04374723	0.08492822
20150228	0.04293864	0.08343252
20150301	0.04287641	0.08304753

Figure 3 is a graph based on the data from Table 2. The horizontal axis represents time, and the vertical axis represents the coverage of the two different methods.

Based on the data of Fig. 3, Coverage's formula is:

Sum(CF) = 0.49299065, Sum(SVD++) = 0.97437603.

Ratio = Sum(SVD++)/Sum(CF) * 100 % = 198 %.

SVD++ is 1.98 times than the benchmark model CF for Coverage.

Figure 4 is a graph based on the data from Table 3. The horizontal axis represents time, and the vertical axis represents the click through rate of the two methods.

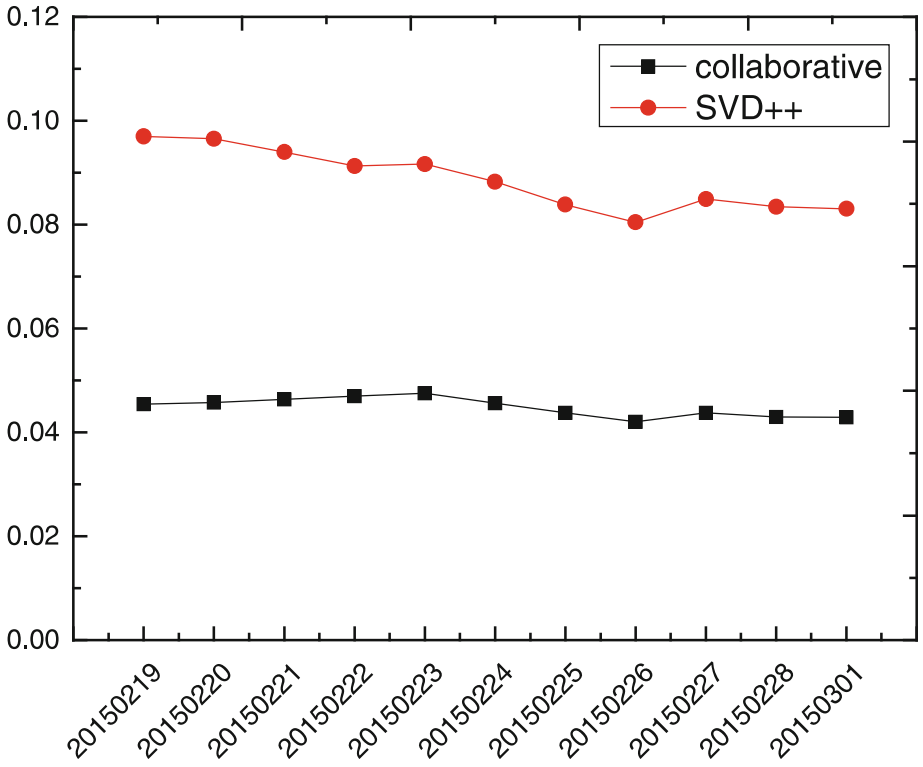

Fig. 3. Coverage

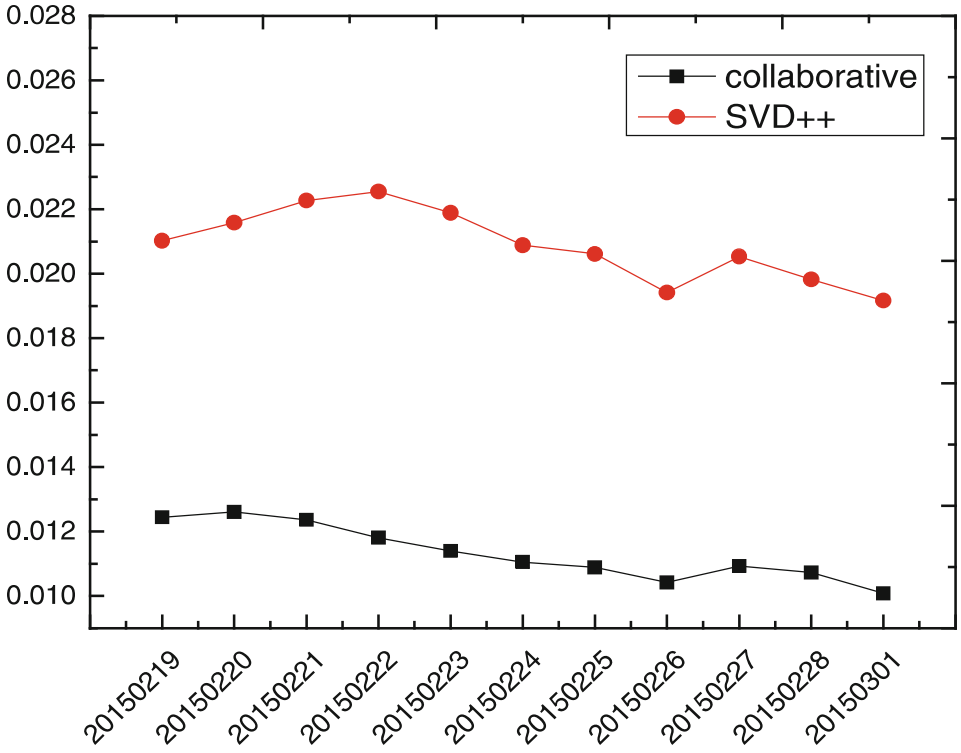

Fig. 4. CTR

Table 3. CTR

Date	Item-based CF	SVD++
20150219	0.01244407	0.02102437
20150220	0.01260828	0.02158315
20150221	0.01236271	0.02227533
20150222	0.01181024	0.02255305
20150223	0.01139838	0.02188653
20150224	0.01105066	0.02088070
20150225	0.01088227	0.02061793
20150226	0.01041635	0.01941242
20150227	0.01092723	0.02053252
20150228	0.01072970	0.01981970
20150301	0.01007968	0.01916598

Based on the data of Figure *******4.33, CTR's formula is:
Sum(CF) = 0.12470957, Sum(SVD++) = 0.22975168.
Ratio = Sum(SVD++)/Sum(CF) * 100 % = 184 %.
SVD++ is 1.84 times than the benchmark model CF for CTR.

Figure 5 is a graph based on the data from Table 4. The horizontal axis represents time, and the vertical axis represents the Click Conversion Rate.

Based on the data of Fig. 4, CCVR's formula is:
Sum(CF) = 0.01519496, Sum(SVD++) = 0.01328705.
Ratio = Sum(SVD++)/Sum(CF) * 100 % = 87 %.
SVD++ is 0.87 times than the benchmark model CF for CCVR.

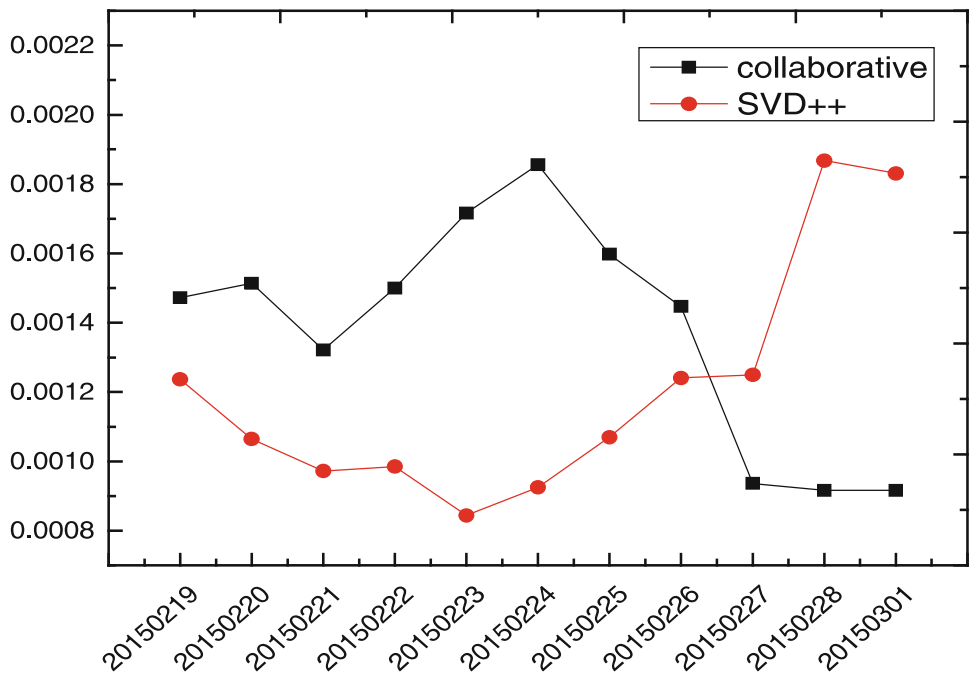

Fig. 5. CCVR

Table 4. CCVR

Date	Item-based CF	SVD++
20150219	0.00147275	0.00123653
20150220	0.00151384	0.00106496
20150221	0.00132205	0.00097182
20150222	0.00149981	0.00098522
20150223	0.00171674	0.00084412
20150224	0.00185615	0.00092535
20150225	0.00159744	0.00106980
20150226	0.00144665	0.00124069
20150227	0.00093633	0.00124961
20150228	0.00091617	0.00186800
20150301	0.00091701	0.00183094

Figure 6 is a graph based on the data from Table 5. The horizontal axis represents time, and the vertical axis represents the Impress Conversion Rate.

Based on the data of Fig. 6, ICVR's formula is:

Sum(CF) = 0.00017331, Sum(SVD++) = 0.00027456.

Ratio = Sum(SVD++)/Sum(CF) * 100 % = 158 %.

SVD++ is 1.58 times than the benchmark model CF for ICVR.

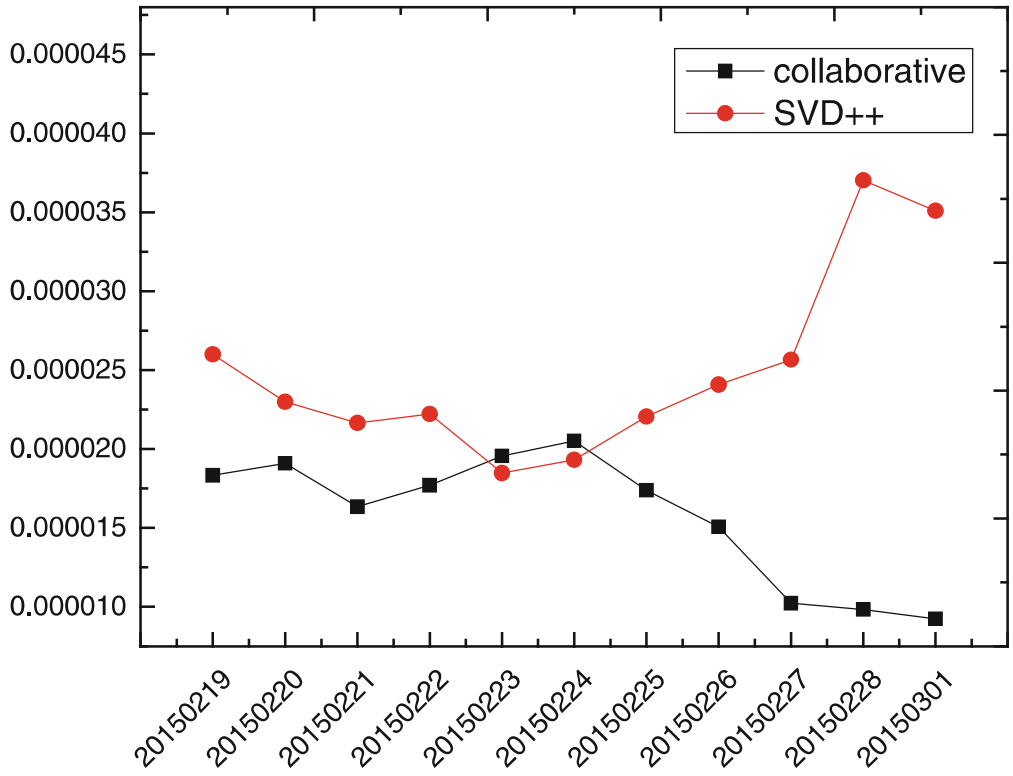

Fig. 6. ICVR

Table 5. ICVR (Order Line/Display).

Date	Item-based CF	SVD++
20150219	0.00001833	0.00002600
20150220	0.00001909	0.00002299
20150221	0.00001634	0.00002165
20150222	0.00001771	0.00002222
20150223	0.00001957	0.00001847
20150224	0.00002051	0.00001932
20150225	0.00001738	0.00002206
20150226	0.00001507	0.00002408
20150227	0.00001023	0.00002566
20150228	0.00000983	0.00003702
20150301	0.00000924	0.00003509

Figure 7 is a graph based on the data from Table 6. The horizontal axis represents time, and the vertical axis represents the Request Conversion Rate.

Based on the data of Fig. 7, RCVR's formula is:

Sum(CF) = 0.00000782, Sum(SVD++) = 0.00002417.

Ratio = Sum(SVD++)/Sum(CF) * 100 % = 309 %.

SVD++ is 3.09 times than the benchmark model CF for RCVR.

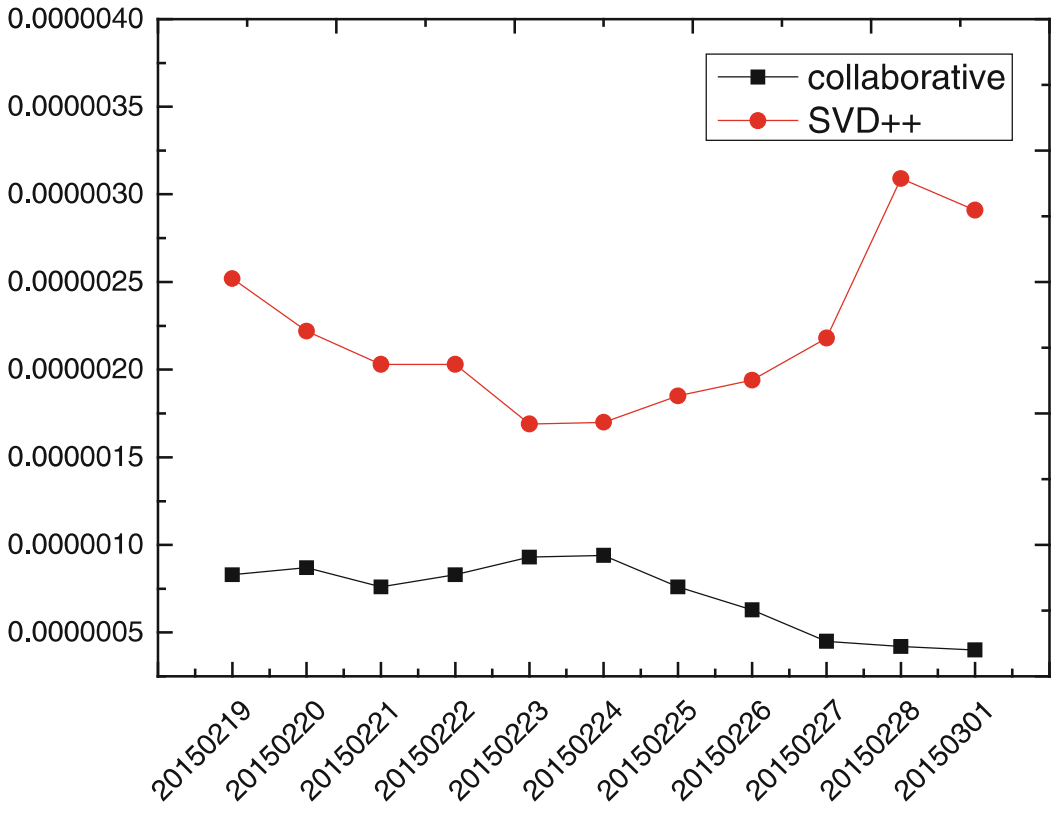

Fig. 7. RCVR

Table 6. RCVR

Date	Item-based CF	SVD++
20150219	0.00000083	0.00000252
20150220	0.00000087	0.00000222
20150221	0.00000076	0.00000203
20150222	0.00000083	0.00000203
20150223	0.00000093	0.00000169
20150224	0.00000094	0.00000170
20150225	0.00000076	0.00000185
20150226	0.00000063	0.00000194
20150227	0.00000045	0.00000218
20150228	0.00000042	0.00000309
20150301	0.00000040	0.00000291

Figure 8 is a graph based on the data from Table 7. The horizontal axis represents time, and the vertical axis represents the Gross Merchandise Volume.

Based on the data of Fig. 8, GMV's formula is:

Sum(CF) = 7245.23, Sum(SVD++) = 17683.91.

Ratio = Sum(SVD++)/Sum(CF) * 100 % = 244 %.

SVD++ is 2.44 times than the benchmark model CF for GMV.

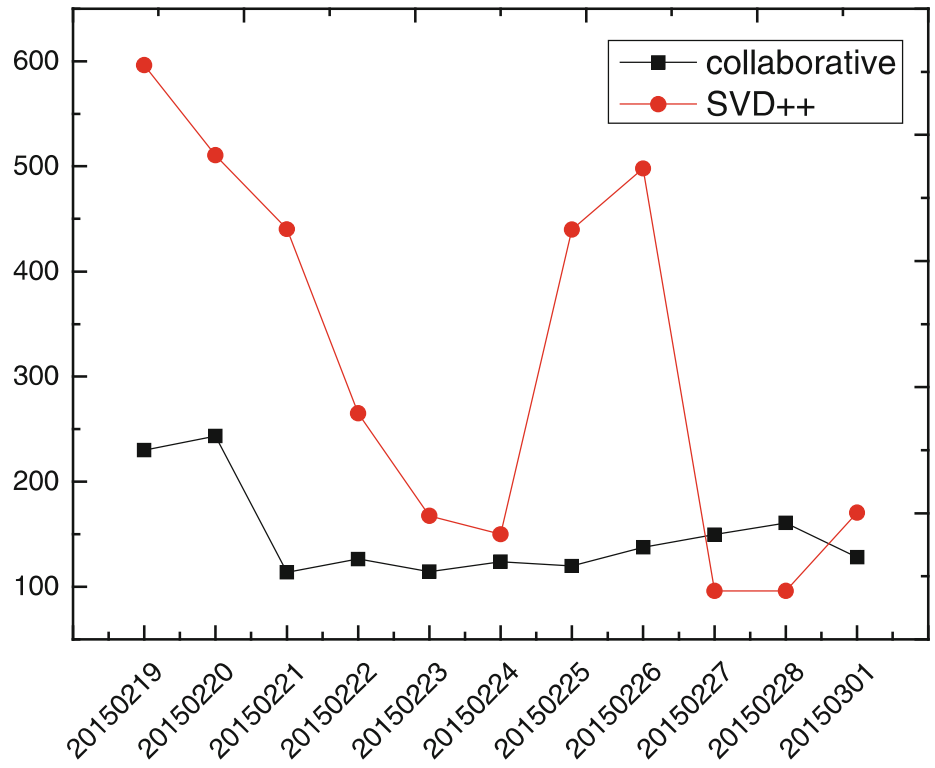

Fig. 8. GMV

Table 7. GMV (Unit: Yuan).

Date	Item-based CF	SVD++
20150219	229.96	596.32
20150220	243.28	510.56
20150221	113.68	440.41
20150222	126.42	264.85
20150223	114.34	167.49
20150224	123.79	149.91
20150225	119.73	439.89
20150226	137.64	497.86
20150227	149.50	96.07
20150228	160.75	96.07
20150301	128.00	170.52

4.3 Analysis of Test Result

The above online statistic results represent that SVD++ algorithm's dimension effects in the above statistics is better than the benchmark index Item-based CF algorithm. Except CCVR, Coverage increased 98 %, CTR increased 84 %, ICVR increased 58 %, RCVR increased 209 %, and GMV increased 144 %. the advantage is obvious, and CCVR of SVD++ was slightly lower than that of the Item-based CF, that is mainly because SVD++ hits is more than the Item-based CF, which leads to a big denominator.

5 Conclusion

In this paper, based on the current e-commerce needs and the current situation of the development of personalized recommender system, how to use SVD++ algorithm to improve the recommendation of personalized items quality problem has been studied. This paper has carried on the design and implementation of a distributed SVD++ algorithm, and applied the algorithm implementation in the actual test in the recommended products. In the process of application, this article is based on the electronic commerce recommendation background to confirm the optimization goal of SVD++, and the original data set also be optimized according to the business needs. The final test results represent that this paper successfully solved the problem of mass data processing in e-commerce personalized recommender systems, the size of its data upped to TB level, and it solved the problem of sparse matrix to a certain extent, and significantly improved the quality of the personalized product recommendation, the evaluation indicator, Coverage increased 98 %, CTR increased 84 %, ICVR increased 58 %, RCVR increased 209 %, and GMV increased 144 %.

References

1. Jia, Y., Zhang, C., Lu, Q., Wang, P.: Users' brands preference based on SVD++ in recommender systems. In: 2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA), pp. 1175–1178. IEEE Press, Ottawa (2014)
2. Hu., Y.F., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: Proceedings of the IEEE International Conference on Data Mining (ICDM 2008), pp. 263–272. IEEE CS Press (2008)
3. Zhang C.Y.: A LBS-based mobile personalized recommendation system. Sci. Technol. Eng. **30**, 7439–7442, 7447 (2011)
4. Zhang, X., Niu, B., Zhao, J.: A novel data mining model based on SOAP in E-commerce. In: Ninth International Conference on Hybrid Intelligent Systems (HIS 2009), pp. 404–407. IEEE, Shenyang (2009)
5. Ren, F., Zhang, C., Liu, L., Xu, W., Owall, V., Markovic, D.: A square-root-free matrix decomposition method for energy-efficient least square computation on embedded systems. Embedded Syst. Lett. **6**, 73–76 (2014). IEEE
6. Funk, S.: Netflix Update: Try This at Home. http://sifter.org/~simon/journal/20061211.html
7. Liang, X.: Recommendation System Practice, pp. 179–193. People's Posts and Telecommunications Press, Beijing (2012)
8. Koren ,Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD Conference, pp. 426–434 (2008)
9. Hynninen, P., Kauppinen, M.: A/B testing: A promising tool for customer value evaluation. In: Requirements Engineering and Testing (RET), pp. 16–17. IEEE, Karlskrona (2014)
10. Lin, X., Wang, P., Wu, B.: Log analysis in cloud computing environment with Hadoop and Spark. In: Broadband Network & Multimedia Technology (IC-BNMT), pp. 273–276. IEEE, Guilin (2013)
11. IBM Corperation: Spark, an alternative for fast data analytics. http://www.ibm.com/developerworks/library/os-spark
12. Hanna, A.I., Yates, I., Mandic, D.P.: Analysis of the class of complex-valued error adaptive normalised nonlinear gradient descent algorithms. In: Proceedings of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2003), vol. 2, pp. II–705–708. IEEE (2003)
13. Malago, L., Matteo. M., Pistone, G.: Stochastic Natural Gradient Descent by estimation of empirical covariances. In: 2011 IEEE Congress Evolutionary Computation (CEC), pp. 949–956. IEEE, New Orleans (2011)
14. Boukis, C.G., Papoulis, E.V.: A normalised adaptive amplitude nonlinear gradient descent algorithm for system identification. In: Proceedings of the 2003 10th IEEE International Conference on Electronics, Circuits and Systems (ICECS 2003), vol. 3, pp. 1042–1045. IEEE (2003)