ISSN 2616-5775 Vol. 5, Issue 14: 82-87, DOI: 10.25236/AJCIS.2022.051413

Design of collaborative filtering recommendation algorithm combining time weight and reward and punishment factors

Panpan Yang, Guangming Li*, Xin Xue

School of Electronic Information and Artificial Intelligence, Shaanxi University of Science and Technology, Xi'an, China
**Corresponding author: ligm1631@163.com*

Abstract: *For distributed recommendation systems built on Spark and Flink big data platforms, when machine learning libraries are used for offline recommendation, Cartesian product calculation will be carried out, which consumes a large amount of memory and causes long algorithm execution time. In real-time recommendation, traditional recommendation algorithm model cannot dynamically sense the interest drift of users, resulting in poor recommendation results. To solve the above problems. This paper introduces the heapsort algorithm into the offline recommendation algorithm to solve the problem that ALS algorithm in MLlib will perform Cartesian product operation in model prediction, which consumes a lot of memory and takes a long time to execute the algorithm. The real-time recommendation algorithm introduces Ebbinghaus forgetting curve, and the fusion of timing weight and reward and punishment factors to dynamically perceive the drift of user interest and generate personalized TOP-N recommendation results. Experimental results show that the execution rate of the offline algorithm adopted in this paper is significantly improved when the RMSE index is basically unchanged. The accuracy rate and recall rate of the real-time recommendation algorithm are significantly improved, and the recommendation results are more in line with users' interests.*

Keywords: *Heap sort, ebbinghaus forgetting curve, time weight, reward and punishment factors*

1. Introduction

This With the advent of the era of big data, recommendation systems have been rapidly developed, such as Jingdong, Taobao, Amazon product recommendation system, Youtube, Netflix video recommendation system, NetEase, Sohu news recommendation system, music, social recommendation system and so on [1,2]. The development of recommendation system application platform cannot be separated from recommendation algorithm. The recommendation algorithm mines users' implicit preferences through historical behavior data and provides users with a personalized product list of length N, namely Top-N recommendation results. How to recommend personalized and accurate recommendation results to users, alleviate the problem of information overload, and solve the problem of sparse data and cold start has become a hot topic in the research of recommendation system.

The impact on the performance of the recommendation system mainly includes two parts, one is the deployment of the recommendation system platform, the other is the recommendation algorithm. Recommendation algorithms are developing rapidly at home and abroad. Goldberg et al. [3] proposed collaborative filtering algorithm and applied it in the Tapestry experimental mail system, which has become an important cornerstone for the development of recommendation system. At present, collaborative filtering algorithm is also the most widely used recommendation algorithm. In order to solve the problems of sparse matrix and cold start, Paatero et al. [4] proposed alternating least square method, and Lee et al. [5] proposed a non-negative matrix decomposition algorithm completed by multiplication iteration. In 2006, Netflix held a recommendation system competition, and matrix decomposition algorithm and constrained Poltz algorithm were born [6,7]. Although the above several recommendation algorithms have achieved good results, Mishra et al. [8] adopted a new combination approach to solve the sparsity problem of the recommendation system, and the hybrid recommendation algorithm can better make up for the shortcomings of the single recommendation algorithm. Yu et al. [9] proposed a recommendation system that integrated users' social status and matrix decomposition. Lu[10] proposed a collaborative filtering recommendation system integrating user interest and rating differences. Chen et al. [11] proposed a scenic spot recommendation system based on fusion graph representation

ISSN 2616-5775 Vol. 5, Issue 14: 82-87, DOI: 10.25236/AJCIS.2022.051413

learning and sequence mining, and the hybrid recommendation algorithm began to rise.

Recently, more and more scholars have paid attention to the problem of perceiving user interest deviation by integrating time information into recommendation algorithm. Short sequence recommendation is a hot related research problem. At present, there are cyclic neural network modeling methods for short sequences to predict users' preference degree for commodities. Singh et al. [12] modeled the commodities that users have browsed and proposed a model recommendation algorithm based on memory priority. Some scholars introduced the Ebbinghaus forgetting curve into the collaborative filtering algorithm to solve the user interest drift. For example, Gao et al. [13] incorporated the collaborative filtering algorithm of forgetting curve into the above information in the context of time, and only recommended from the perspective of similar users without considering the influence of users' historical scoring items. Feng et al. [14], collaborative filtering algorithm based on memory activation theory also incorporates the above information under time and only considers the influence of item groups, without considering the influence of similar users.

Therefore, this paper introduces the Ebbinghaus forgetting curve into the collaborative filtering algorithm, integrates the time weight and reward and punishment factors into the calculation of similar users and similar items, and proposes a collaborative filtering recommendation algorithm that integrates the time weight and reward andpunishment factors.

2. Recommendation system algorithm design

Common web portals recommend videos, news, commodities or music of interest to users by analyzing the log information generated by the recommendation system [15]. In the movie recommendation system, the number of movie Rating.bat scores is very large. How to find useful information from these data is a big challenge for the recommendation algorithm and the operating platform of the algorithm model. In the movie scoring matrix, there is a huge difference in the amount of data between the movie scored by users and the movie, so the sparse matrix problem will occur when the scoring matrix is constructed. The user feature matrix and the movie feature matrix can be obtained by matrix decomposition, and then the optimal solution can be obtained by constructing the loss function and selecting the alternating least square method. Then multiply the trained two matrices to get the final predictive scoring matrix, and get the Top-N items with the highest scores among each user, which is the final recommendation result.

2.1. Optimization of offline recommendation algorithm

The ALS of Spark Mlib uses the alternate least square method to solve collaborative filtering. The ALS of Spark Mlib uses the data partition structure. Related users and scores are stored in the same partition, which reduces the cost of data exchange between partitions. Through analysis, it is found that when the model trained by ALS algorithm of Spark Mlib makes model prediction and recommendation of Top-N movies to users, the machine learning library uses the Cartesian product of the User ID set and Item ID set, which results in a long time spent in model prediction and consumes too much memory resources. In fact, you only need to get the Top N movies recommended by each user with the highest rating, and you do not need to keep the user's ratings for all movies.

In the MapReduce process of Hadoop, Shuffle spans the Map end and Reduce end. Shuffle includes the Spill process on the Map end. The Spill process includes the Collect, Sort, Spill, and Merge steps. During the Merge process, data can be merged and sorted using the heapsort algorithm. Therefore, this paper draws on this process and introduces heap sorting algorithm in model training to optimize alternate least squares (ALS). By maintaining a minimum heap all the time, Top-N score prediction of users can be obtained. The algorithm process is as follows:

Step 1 To get a User's rating for a movie, multiply a user by an Item.

Step 2 If the number of scores is less than the recommended top-N list, add one to Item and repeat step one until a full minimum heap is built, with Min being the Top of the heap.

Step 3 After iterating all items, if the new score is greater than Min at the top of the minimum heap, Min will be replaced, and the minimum heap will be dynamically adjusted to meet the minimum heap condition.

Step 4 Through step 3, Top-N items with the highest rating of a user can be generated.

ISSN 2616-5775 Vol. 5, Issue 14: 82-87, DOI: 10.25236/AJCIS.2022.051413

Step 5 Repeat steps one through four for each user to end up with the Top-N recommended items for each user.

2.2. Real-time recommendation algorithm design

The recommendation result calculated by the offline recommendation algorithm after the user updates a movie score is not much different from the recommendation result generated when the user does not update, so it does not have the real-time recommendation ability. The real-time recommendation should produce new recommendation results after the user scores once or several times. In addition, the calculation amount of the algorithm should be as small as possible. The user feature matrix calculated offline is $U(m\times k)$, and the movie feature matrix is $V(n\times k)$. The features of the user and the movie can be described by k features respectively, so the $V(n\times k)$ vector of each line of $<t_1,t_2,t_3\ldots t_k>$ is used to represent the features of each movie. Therefore, the feature vector of any two movies q and r is the similarity degree $Sim(q,r)$ between V_q and V_r, respectively. It can be expressed using the cosine of V_q and V_r:

$$Sim(q,r)=\frac{\sum_{i=0}^{k}(t_{qi}\times t_{ri})}{\sqrt{\sum_{i=0}^{k}t_{qi}^2}\sqrt{\sum_{i=0}^{k}t_{ri}^2}} \tag{1}$$

In the formula, t_{qi} represents the rating of movie q, and t_{ri} represents the rating of movie r.

The following is the calculation of the user's recommendation priority for the unrated movie q: Firstly, obtain the most recent K rating of the user in chronological order, denoted as RK; obtain the most similar K movie set of movie q, denoted as S; then, for each movie $q\in S$, calculate the recommendation priority of movie q, the formula is as follows:

$$E_{uq}=\frac{\sum_{r\in RK}Sim(q,r)\times R_r}{Sim\_sum}+\lg\max\{incount,1\}-\lg\max\{decount,1\} \tag{2}$$

In the formula, R_r is the user's rating of movie r, $Sim(q,r)$ is the similarity between movie q and movie r, and the threshold of minimum similarity is set as 0.6. Sim_sum represents the number of films whose similarity is greater than the minimum threshold in q and RK; $incount$ represents the number of movies in RK that are similar to movie r and are rated at least 3 points (the maximum score is 5)., and $decount$ represents the number of movies with a score less than 3 points. The meaning of the first term in the formula is that for each candidate movie q, the movie K with high similarity to q is found from the user's recent r scores, and the similarity between the two is multiplied by the average of the user's score to r, which is used as the user's predicted score to movie q. The reward factor of movie q is $\lg\max\{incount,1\}$. The more similar films with high scores, the movie should be recommended to users; the punishment factor of movie recommendation is $\lg\max\{decount,1\}$; the more similar films with low scores, the less should be recommended.

In life, users' interests are not static and will change with time. In order to solve the problem of user interest deviation, Ebbinghaus forgetting curve is introduced. Based on the change law of the forgetting curve, as shown in the formula:

$$f(t)=e^{-t} \tag{3}$$

In the formula, t is the difference between the earliest time record and the current time in the user evaluation film set. When t=0, $f(t)=1$, and when t approaches infinity, $f(t)$ is infinitely close to 0, which conforms to the change trend of Ebbinghaus forgetting curve. The calculation formula of time weight is as follows:

ISSN 2616-5775 Vol. 5, Issue 14: 82-87, DOI: 10.25236/AJCIS.2022.051413

$$g(t) = 1 - \frac{1}{1 + e^{t - t_{\min}}} \tag{4}$$

In the formula, t is the user's current rating time of the movie, and $t_{\min}$ is the earliest rating recording time in the movie set. Generally, it meets $t >= t_{\min}$. The value of $g(t)$ is in the range of $(0,1)$. The longer the user scores, the smaller the value of $g(t)$, and the smaller the weight, the smaller the impact of recommendation. The time weight is incorporated into Formula (2).

$$E_{uq} = \frac{\sum_{r \in RK} Sim(q,r) \times R_r'}{Sim\_sum} + \lg \max\{incount, 1\} - \lg \max\{decount, 1\} \tag{5}$$

In the formula $R_r' = R_r \times \alpha g(t)$, α is used to adjust the change degree of time weight.

When calculating the similarity between user-rated movies, we took into account the influence of time factor on the similarity of movies, and continued to introduce the time weight into the similarity calculation between movie q and movie r.

$$E_{uq} = \frac{\sum_{r \in RK} Sim(q,r) \times R_r'}{Sim\_sum} + \lg \max\{incount, 1\} - \lg \max\{decount, 1\} \tag{6}$$

In the formula, $Sim(q,r)' = \delta(Sim(q,r) \times \beta e^{-|t_q - t_r|})$, $\delta(x) = \frac{1}{1 + e^{-x}}$ is function Sigmod, which is used to keep the result of similarity between $(0,1)$. t_q and t_r respectively represent the scoring time of movie q and movie r. The larger the value of $|t_q - t_r|$ is, the greater the interval between the two movies scored by users, and the smaller the movie similarity is. β is used to adjust the change degree of the time weight function.

3. Experimental data set and evaluation index

Using experimental data set is provided by the university of Minnesota GroupLens MovieLens dataset (https://grouplens.org/datasets/movielens/), the experiment of ml-latest-small experiment with ml-1m data set, The ml-latest-small dataset contains 9742 movies and collects 100836 scores generated by 610 users. The ml-1m dataset contains 3900 movies and collected 10,0209 score data generated by 6040 users. The dataset mainly includes the data sets of movies, ratings and tags. In the experiment, we selected the above data sets, 80% of which were used as the model training set and 20% as the model test set. Root mean square error (RMSE) and mean absolute error (MAE) were selected to evaluate the accuracy of score prediction. Accuracy Precision and Recall were used to evaluate the prediction accuracy recommended by Top-N.

4. Experimental results and discussion

4.1. Offline algorithm performance testing

Experiment selects the ml-1m data set for the improved offline recommendations ALS algorithm model parameter Rank, λ, selection of Iteration. Rank represents hidden features. The more hidden features retained, the better the training effect will be, but it will increase the memory consumption. The regularization parameter λ can avoid overfitting of the algorithm model, and its value determines the quality of the algorithm model. The value of Iteration number iteration is too small, so the model training is not sufficient. The more iterations, the longer the algorithm execution time. In advance experiments, it can be seen that when the value of λ is 0.1 and Iteration is 5, the evaluation index RMSE will get the minimum value, MSE and MAE will reach the optimal value, and the error will also be minimal. Therefore, the value of λ is set to 0.1 and Iterations is 5, and the value of Rank increases from 10 to 200. Table 1 shows the value of experiment Rank.

ISSN 2616-5775 Vol. 5, Issue 14: 82-87, DOI: 10.25236/AJCIS.2022.051413

Table 1: Influence of implicit feature Rank on evaluation index

Rank	The execution time was not improved / ms	RMSE	Improved execution time / ms	RMSE
10	3689	0.922	2916	0.932
20	4437	0.921	3493	0.925
50	5618	0.916	3795	0.916
100	6585	0.915	4710	0.914
150	7943	0.913	5361	0.912
200	9976	0.911	6544	0.910

According to the experimental results, the improved ALS algorithm reduces the calculation of the Cartesian product, improves the algorithm execution efficiency, and shorens the algorithm execution time significantly. When the Rank value is 50, the RMSE index is equal, and the training time will increase significantly as the Rank continues to increase. Therefore, under the condition of combining the algorithm execution time and error, When the Rank is 50, good recommendation results can be obtained.

On the test set, multiple experiments were conducted on the improved ALS algorithm model and the unimproved algorithm model with the help of RMSE and the algorithm model execution time. When the Rank was 50, the optimized ALS algorithm model compared with the unoptimized algorithm, the algorithm execution rate increased by 32.44%.

4.2. Real-time recommendation algorithm performance testing

In the real-time recommendation algorithm, ml-1m data set is selected as the data set, the length of fixed recommendation list K is 10, and the number of Redis message queue user score records is 20. The accuracy rate and recall rate are used to debug the time weight in the real-time recommendation algorithm. When the value of time parameter α in the algorithm model is 0.65 and the value of time parameter β is 0.6, relatively high accuracy and recall rate can be achieved, and the recommendation effect is the best at this time.

To verify the comprehensiveness of the algorithm model, The ml-latest-small dataset was compared with the User-CF collaborative filtering algorithm [16], the Item-CF collaborative filtering algorithm [17], the forgetting curve collaborative filtering algorithm [13], and the memory activation theory collaborative filtering algorithm [14].

Table 2: Performance comparison of real-time recommendation algorithms

Algorithm	Accuracy rate	Rate of recall
User-CF[16]	0.104	0.095
Item-CF[17]	0.105	0.097
FC-CF[13]	0.117	0.122
MA-CF[14]	0.125	0.124
Ours	**0.128**	**0.135**

According to the experiment in Table 2, the collaborative filtering algorithm proposed in this paper, which integrates time weight and reward and punishment factors, has higher recommendation accuracy than traditional recommendation algorithms. Compared with the FC-CF[13] and MA-CF[14] algorithms, the real-time recommended algorithm in this paper is more comprehensive and has more advantages in algorithm performance.

5. Summary and Prospect

The improved ALS algorithm model in this paper reduces the influence of Cartesian product when the model is predicted, and the algorithm execution rate is increased by 32.44% compared with the unimproved algorithm model. Based on the user interest shift, the real-time recommendation algorithm introduces the Ebbinghaus forgetting curve and integrates the time weight and reward and punishment factors. Compared with the traditional recommendation algorithm, the algorithm is more comprehensive. The recommendation method combining time weight and reward and punishment factors is applied to the recommendation system, which can greatly improve the recommendation accuracy and execution efficiency of the recommendation algorithm, and has certain application value.

ISSN 2616-5775 Vol. 5, Issue 14: 82-87, DOI: 10.25236/AJCIS.2022.051413

References

[1] PICHL M, ZANGERLE E, SPECHT G. Improving Con-text-Aware Music Recommender Systems: Beyond the Pre-filtering Approach [Z]. Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval. Bucharest, Romania; Association for Computing Ma-chinery. 2017: 201–208.
[2] SCHEDL M, KNEES P, GOUYON F. New Paths in Music Recommender Systems Research [Z]. Proceedings of the Eleventh ACM Conference on Recommender Systems. Como, Italy; Association for Computing Machinery. 2017: 392–393
[3] Goldberg D, Nichols D A, Oki B M, et al. Using collabo-rative filtering to weave an information TAPESTRY[J]. Communications of the ACM, 1992, 35: 61-70.
[4] Paatero P, Tapper U. Analysis of Different Modes of Fac-tor Analysis as Least Squares Fit Problems[J]. Chemo-metrics and Intelligent Laboratory Systems, 1993, 18: 183-194.
[5] Lee D D, Seung H S. Algorithms for non-negative matrix factorization[J]. Advances in Neural Information Pro-cessing Systems, 2001, 13: 2421-2456.
[6] Ma S, Goldfarb D, Chen L. Fixed Point and Bregman It-erative Methods for Matrix Rank Minimization [J]. Math-ematical Programming, 2009, 128: 321-353.
[7] Hinton G E. Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair[C]. Proceedings of the 27th International Conference on International Confer-ence on Machine Learning. Santa Fe. New Mexico. 2010. 27. 807-814.
[8] Mishra N, Chaturvedi S, Mishra V, et al. Solving Sparsity Problem in Rating-Based Movie Recommendation Sys-tem[M]. Berlin: Springer-Verlag Press, 2017.
[9] Yu Yonghong, Gao Yang, Wang Hao, et al. Recommendation Algorithm Based on User Social Status and Matrix Decomposition. Journal of Computer Research and Development, 2018, 55 (01): 113-124.
[10] Lu Hang, Shi Zhibin, Liu Zhongbao. Collaborative Filtering Recommendation Algorithm based on User Interest and Rating Difference [J]. Computer Engineering and Applications, 2020, 56 (07): 24-29.
[11] Chen Yuanpeng, Gu Tianlong, Bin Chenzhong, et al. A Scenic spot recommendation method based on Fusion Graph Representation Learning and Sequence Mining [J]. Computer Engineering and Design, 2020, 41 (12): 3563-3569.
[12] Singh A, Sharma D. Evaluation Criteria for Measuring the Performance of Recommender Systems[C]. International Conference on Reliability. IEEE, 2015.
[13] Gao Maoting, Xu Binyuan. Recommendation algorithm based on cyclic neural network [J]. Computer Engineering, 2019, 45(08):198-202.
[14] Feng Yong, Zhang Bei, Qiang Baohua, et al. MN-HDRM: Hybrid Dynamic recommendation model of multi-neural networks with Long and short interest [J]. Chinese Journal of Computers, 2019, 42(01):16-28.
[15] GIRSANG A S, EDWIN A W. Song Recommendation System Using Collaborative Filtering Methods [Z]. Pro-ceedings of the 2019 The 3rd International Conference on Digital Technology in Education. Yamanashi, Japan; Association for Computing Machinery. 2019: 160–162
[16] Qiu Ningjia, Xue Lijiao, He Jinbiao, et al. Application Research of Computers, 2020, 37(10):1-6.
[17] Li Xujun, Yin Zi, LV Qiang. Time Context Collaborative Filtering Recommendation based on Counterfactual Reasoning [J]. Computer Engineering and Design, 2019,42(10):2876-2883.