ALS RECOMMENDATION ALGORITHM BASED ON KL DIVERGENCE AND TIME WEIGHTING

Yang Jie Lu[1]*, Dr. Norriza Hussin[2], Assoc. Prof. Dr. Rajamohan[3]

Abstract
People are living in the background of the rapid development of the Internet, and the network data generated has also shown an explosive growth. For this reason, the need for access to personalized data has become more apparent. Therefore, recommendation systems have become one of the hot topics of discussion in recent years. The traditional recommendation algorithms, such as collaborative filtering recommendation algorithms, ignore the influence of time factor on the rating, and also have the problem of low recommendation accuracy on the sparse data. Based on the above two problems, this paper proposes an Alternating Least Squares (ALS) recommendation algorithm based on Kullback-Leibler divergence (KL) and time weighting, which uses KL divergence to calculate item similarity while incorporating a time factor, referred to as KL-TW-ALS algorithm. Three experiments were also conducted on the Spark platform to compare with the item based collaborative filtering algorithm and the ALS-based collaborative filtering algorithm, showing that the optimized ALS algorithm has improved accuracy and performance.

Keywords: Recommendation algorithm, ALS algorithm, KL divergence, Time factor, Collaborative filtering

[1]*MSc Research Scholar, Faculty of Engineering, Built Environment and Information Technology SEGi University Kota Damansara, Email: nhjie@163.com
[2]Parthasarathy Senior Lecturer, Faculty of Engineering, Built Environment and Information Technology SEGi University Malaysia, Email: prajamohan@segi.edu.my
[3]PhD Research Supervisor, Faculty of Engineering, Built Environment and Information Technology Kota Damansara, Email: norriza@segi.edu.my

***Correspondence Author:** Yang Jie Lu
*MSc Research Scholar, Faculty of Engineering, Built Environment and Information Technology SEGi University Kota Damansara, Email: nhjie@163.com

DOI: - 10.48047/ecb/2023.12.si5a.0248

I. INTRODUCTION

With the rapid development of the Internet era, a large amount of web data are collected, stored and analyzed. These big data have also triggered the problem of data overload, and it is very difficult for users to find the content they are interested in, in front of the huge amount of data. For this reason, users get personalized recommendations by predicting their interests with the help of recommendation systems [1].

To make a good recommendation system, the key is still the design of recommendation algorithm. As an important application of artificial intelligence, recommendation algorithm has received wide attention in recent years, mainly applied to many fields such as e-commerce, news, music, video and social. The recommendation algorithms are still mainly analyzing historical interaction data in order to calculate the association patterns between users and different items [2].

Common recommendation algorithms can be classified according to data sources as demographic -based recommendation algorithms, content-based recommendation algorithms, and collaborative filtering recommendation algorithms [3]. Content-based recommendation algorithms usually use text mining techniques, such as Latent Dirichlet Allocation (LDA) a topic model proposed by some researchers. The model can be used to determine the topic to which each article belongs and the topic to which each word belongs by means of a probability distribution, and thus to classify and model the topic of the text [4]. These algorithms can efficiently extract information about the subject matter of the items and serve similar items to users based on this.

The recommendation algorithm based on collaborative filtering is based on the similarity between users to make recommendations and is one of the algorithms that are currently used in the recommendation application field, which has good recommendation effect but still has the problem of data sparsity. Some researchers proposed to use the user nearest neighbor model combined with matrix decomposition to complete the recommendation, which improves the recommendation accuracy, but does not consider the similarity between items, and also converges slowly in a large amount of data [5]. And the mainstream big data framework Spark can help the recommendation algorithm's fast completion of the calculation. There are also other studies that investigate model-based collaborative filtering algorithms, such as the ALS algorithm, which can predict users' ratings of items by iteratively solving the matrix decomposition [6]. Also some researchers have proposed item similarity combined with the ALS algorithm to introduce cosine similarity to alleviate the data sparsity problem [7].

In this paper, we propose an ALS recommendation algorithm based on KL divergence and time weighting, referred to as KL-TW-ALS algorithm, which also calculates item similarity with the help of KL divergence, thus alleviating the problem of sparse data. Meanwhile, the time factor is incorporated to do time weighting on the scoring data. And the experimental validation is performed in parallel on Spark platform.

II. RELATED TECHNOLOGY INTRODUCTION

A. Spark framework and working principle

In 2009, a team at the AMP Lab at the University of California (Berkeley) developed the Spark computing framework, a new type of data analysis application with relatively low latency. Spark is not a single component, but a complete ecosystem. Among the tool layers it provides is the MLlib library, which provides Spark's ability to run machine learning algorithms [8]. Among them is the ALS algorithm studied in this paper.

There are four types of roles in Spark that make up an entire Spark runtime environment, The Master role (managing the resources of the entire cluster), the Worker role (managing the resources of individual servers), the Driver role (managing the work of individual Spark tasks at runtime), and the Executor role (executor of individual tasks at runtime). as shown in Figure 1 below.

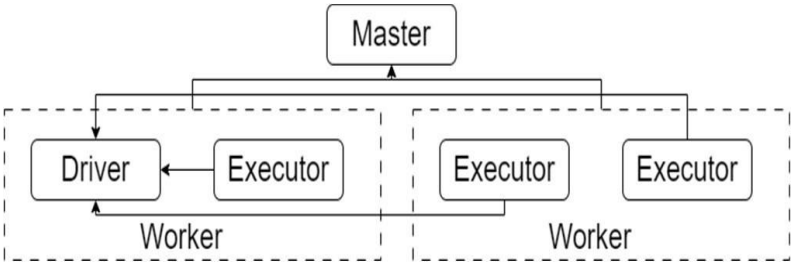

Fig. 1. Spark Role Map

When executing the application, the Driver node in the master node will request CPU/memory resources from the Cluster Manager cluster resource manager. Once these resources are requested, the corresponding Executor process will be started and each job will be executed on that process.

B. Alternating Least Squares (ALS) algorithm

Since 2008 until now, Alternating Least Squares (ALS algorithm) has been the more used recommendation algorithm [9]. To facilitate understanding, the following is an example of a movie recommendation system with a user-movie rating matrix R to demonstrate the ALS algorithm. The dimensionality of the original R data matrix can be reduced to achieve the need for better user characteristics and movie characteristics. At this time, in order to split the original data matrix R into two matrices to reduce the dimensionality. The ALS algorithm can be used to decompose into two matrices, the user feature matrix U and the movie feature matrix D. If the obtained matrix is close to the original matrix, then it has a strong relevance, as in:

$$\hat{R}_{m\times n} = U^T_{m\times k} \cdot D_{k\times n} \approx R \tag{1}$$

In considering the magnitude of the error between predicted and actual values, the mean square error can be used and the corresponding scoring matrix of the iterations can be detected by this function to minimize the total error. In addition, the implicit feature vector of P_u users and the implicit feature vector of Q_i items are added to act as regularization terms, which prevents the corresponding models from overlearning and thus overfitting. And the corresponding λ regularization coefficients can be obtained by always crossiterating the validation [10], as in:

$$L(p,q) = \min \sum_{(u,i\in k)}^{n} (\hat{r}_{u,i} - p_u q_i^T)^2 + \lambda(\| p_u \|^2 + \| q_i \|^2) \tag{2}$$

The above equation is solved using the least squares method by first fixing the eigenmatrix of Q such that the partial derivative of the loss function L(p, q) with respect to p_u is 0, and by alternating the operations until the loss function values converge to obtain, as in:

$$p_u = (Q^T Q + \lambda E)^{-1} + Q^T r_u \tag{3}$$

With the help of the same method, the P identity matrix can be fixed first, as in:

$$q_i = (P^T P + \lambda E)^{-1} + P^T r_i \tag{4}$$

The final matrix is obtained by repeated iterations until the algorithm error value converges or the number of iterations reaches a set value. Finally, the similarity between any two items is obtained by the cosine similarity formula, where $Sim_C(i,j)$ is the similarity between items i and j, and U_{ij} is the set of all users involved in the rating. R_{ui} is the rating of user u on item I, and R_{uj} is the rating of user u on item j, as in:

$$\text{sim}_C(i,j) = \frac{\sum_{u\in U_{ij}} R_{ui} * R_{uj}}{\sqrt{\sum_{u\in U_{ij}} R_{ui}^2} \times \sqrt{\sum_{u\in U_{ij}} R_{uj}^2}} \tag{5}$$

C. Time forgetting curve

The Ebbinghaus forgetting curve is a way to model the decay of human memory for events through temporal weights. This curve fitting function is commonly used in recommender systems to take into account the closeness and timeliness of items. Figure 2 shows the results of fitting the Ebbinghaus forgetting curve to the data (number of days and amount of memory) using the exponential function input curve via the matplotlib package in Python.

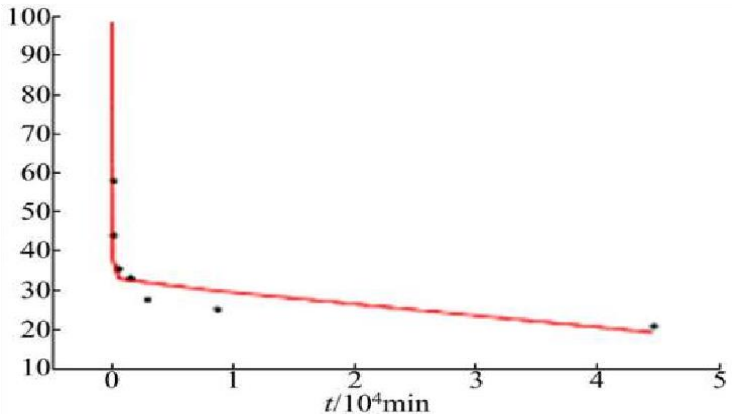

Fig. 2. Python curve fitting

The fitting function of the Ebbinghaus forgetting curve is generally an exponential function that indicates the rate of decay of memory for events over time, as in:

$$T(t) = e^{(-kt)} \tag{6}$$

In the formula (6), T(t) denotes the degree of temporal forgetting, e is the base of the natural logarithm, k is the decay coefficient, and t is the time interval between the occurrence of the event and the current one.

D. KL divergence

Within machine learning, KL divergence is a common measure of similarity between two probability distributions and can measure geometric distance [11]. KL divergence was first evolved from information theory and reflects the average amount of information needed for a probability distribution. Now there are two density functions with different probabilities p and q. The difference between these two probabilities is expressed using $D_{KL}(p||q)$ [12], as in:

$$D_{KL}(p\|q)=\sum_{i=1}^{N} p(x_i)*\log\frac{p(x_i)}{q(x_i)} \tag{7}$$

The KL divergence itself is not a direct measure of similarity and requires the result to be a non-negative distance by opening the root sign, as in:

$$D(P,Q) = \sqrt{D_{KL}(p||q)} \tag{8}$$

It is also possible to normalize the result of the KL divergence so that the result is between 0 and 1 to represent the similarity of two probability distributions, defined in Sim_{KL}, as in:

$$Sim_{KL}(P,Q) = \frac{1}{(1+D(P,Q))} \tag{9}$$

III. KL-TW-ALS ALGORITHM

In the case that the rating matrix is usually sparse, the ALS algorithm can decompose the user-item-rating matrix into a low-order user feature matrix and an item feature matrix. Facing the problem of sparse data, this paper adopts the method of KL divergence and cosine similarity for weight assignment, which can make full use of the item information inside the data set to complete the calculation of item similarity. Facing the problem of recommendation accuracy optimization, this paper takes advantage of the characteristic that people's interest in items decreases with the passage of time and increases the time decay coefficient to improve the accuracy of the recommendation system.

In terms of similarity, KL divergence and cosine similarity are mixed according to certain weights, Sim_{mix} denotes the weighted final similarity, sim_C denotes the item similarity using cosine similarity, and sim_{KL} denotes the item similarity calculated using KL distance, as in:

$$Sim_{mix}(i,j)=\lambda Sim_C(i,j)+(1-\lambda)Sim_{KL}(i,j) \tag{10}$$

In terms of time weighting, a time temporal decay factor is multiplied on each rating item to indicate that this rating becomes progressively less important over time, as in:

$$J=\sum(t*(\hat{r}_{u,i}-p_u q_i^T))^2+\lambda_1(\|p_u\|^2+\|q_i\|^2) \tag{11}$$

In formula (11), t is the time decay coefficient, P_u and Q_i are the user-implicit feature vector and the item-implicit feature vector, respectively, and λ_1 is the regularization parameter.

The algorithm flows roughly as follows:

Input: Data set S

Output: Rating matrix of recommended movies and selection of Top 20

1) Read the data set and determine the user-item rating matrix R.
2) The values of time factors K and t are obtained experimentally, substituted into formula (6), and weighted and updated for R to obtain the new scoring matrix R.
3) The ALS model is trained and the similarity weight parameters are adjusted according to formula (10).
4) The ALS model is obtained by stopping the iteration when the minimum RMSE is obtained or when the maximum number of iterations is reached.

IV. EXPERIMENT AND RESULTS

The processing of huge amount of data is also one of the things that recommendation systems need to face. For this reason, this experiment is based on validation under the Spark framework. The following three types of experiments were designed:

- Experiment to determine the optimal value of the attenuation coefficient k.
- Experiment to determine the optimal value of the similarity weight parameter λ.

- Comparison experiment of three algorithms for each index under the change of the number of recommendations.

A. Experimental environment

For the KL-TW-ALS algorithm proposed in this paper, the experiments are conducted according to the one-master-two slave Spark cluster architecture created based on the Ali Cloud environment, as in:

TABLE I. EXPERIMENTAL ENVIRONMENT CONFIGURATION

Physical Environment		
Configuration	Master Node	Slave Node
Number of hosts	1 set	2 set
Number of CPU cores	2 cores	2 cores
RAM Memory	16 G	8 G
Public Network Bandwidth	100 Mbps	100 Mbps
Software Environment		
System / Software	Configuration	
Operating System	CentOS 7.4	
JDK	Jdk 1.8	
Spark	Spark 2.3.2	
Python	3.8	

B. Experimental data

In this paper, we use the public dataset Movie Lines 100K dataset, which mainly uses data such as user ID (ID of the rated user), movie ID (ID of the rated movie), rating (indicating the user's rating of the movie), and timestamp (timestamp of the rating). The data set is split during the experiment, with 70% of the data being the training set and 30% of the data as the test set.

C. Evaluation indicators

1) Root Mean Square Error (RMSE)

In the ALS model training, a time decay factor k is first fitted, which can only be given a value initialized at a time interval t, and the root mean square error (RMSE) is used to calculate the corresponding evaluation method as a way to examine how large the difference between the evaluation prediction and the actual rating difference can be. S is the total number of scores, r_{ui} is the actual score, and $\hat{r}_{ui}$ is the predicted score, as in:

$$RMSE = \sqrt{\frac{\sum_{u,i \in S} (r_{ui} - \hat{r}_{ui})^2}{|S|}} \quad (12)$$

2) Precision and Recall

In the Top-N recommendation list, Precision accuracy as well as recall are likewise used to measure the performance of the algorithm [13]. For Precision accuracy, it means the percentage of items that should be retrieved out of all retrieved items, as in:

$$Precision(N) = \frac{1}{|s_u|} \sum_{u \in s_u} \frac{R_N(u)}{N} \quad (13)$$

And for recall, it is the percentage of all retrieved items that occupy the items that should have been retrieved, as in:

$$Recall(N) = \frac{1}{|s_u|} \sum_{u \in s_u} \frac{R_N(u)}{R(u)} \quad (14)$$

S_u is the set of all users in the test set, $R_N(U)$ is the items preferred by user u in the TOP-N recommendation results, and R(u) is the rated items in the test set.

3) F1 value

To further validate the performance of the model, the F1 value can be chosen for evaluation using the F1 value, which considers the combined effect of accuracy and recall [14]. Therefore, the F1 value can be chosen as the specific evaluation index in the experiment 2 similarity weight assignment experiment, as in:

$$F1=2*(Precision*Recall)/(Precision+Recall) \quad (15)$$

D. Experimental design and analysis of results

1) Experiment 1: Determining the optimal value of the attenuation coefficient k.

The time interval t was selected as 3 days and the attenuation coefficient k was [0.2,0.8], and the RMSE was used as the evaluation index for different values of k, as in:

TABLE II. RMSE WITH DIFFERENT VALUES OF ATTENUATION COEFFICIENT K

K-value	0.2	0.3	0.4	0.5	0.6	0.7	0.8
RMSE	0.867	0.863	0.859	0.862	0.861	0.864	0.865

Observing Table 2, can find that when the decay sparsity k is taken as 0.4, the ALS algorithm with added time weights can get the optimal RSME.

2) Experiment 2: Determining the optimal value of the similarity weight parameter λ, as in:

TABLE III. COMPARISON OF F1 VALUES UNDER THE CHANGE OF ADJUSTMENT PARAMETER Λ

Parameter λ	F1-value	Parameter λ	F1-value
0	0.049	0.5	0.327
0.1	0.031	0.6	0.368
0.2	0.027	0.7	0.311
0.3	0.185	0.8	0.326
0.4	0.274	0.9	0.298

Before the start of Experiment 2, the best decay coefficient for the time weight of Experiment 1 was selected. The F1 value was used as the evaluation index for Experiment 2.As can be seen from Table 3, the F1 value of $Sim_{mix}(i,j)$ reaches the optimal value when λ is 0.6.

3) Experiment 3: Comparison experiment of three algorithms for each index under the change of the number of recommendations.

To verify the performance of the KL-TW-ALS recommendation algorithm in this paper, the best parameter values obtained from Experiment 1 and Experiment 2 were selected to refine the algorithm model. And the experimental results are derived with the traditional ALS algorithm and the item-based collaborative filtering recommendation algorithm (Item-CF) as the comparison algorithm. The experimental results are shown in the following figures.

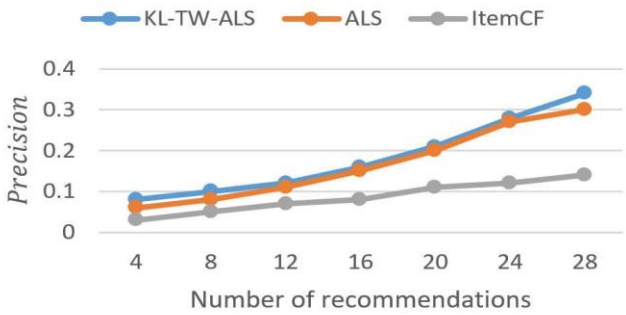

Fig. 3. Comparison chart of precision results under the change of recommended quantity

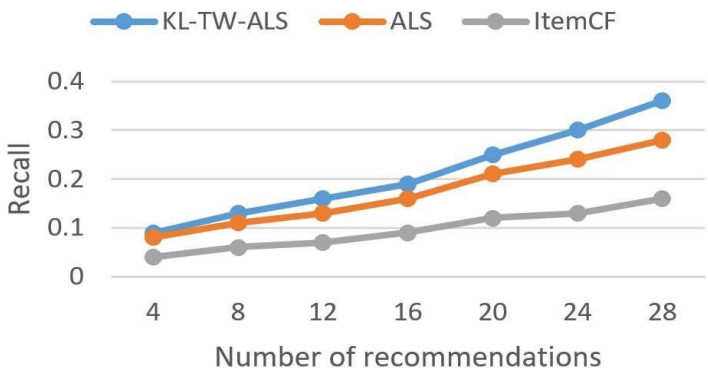

Fig. 4. Comparison chart of recall results under the change of recommended quantity

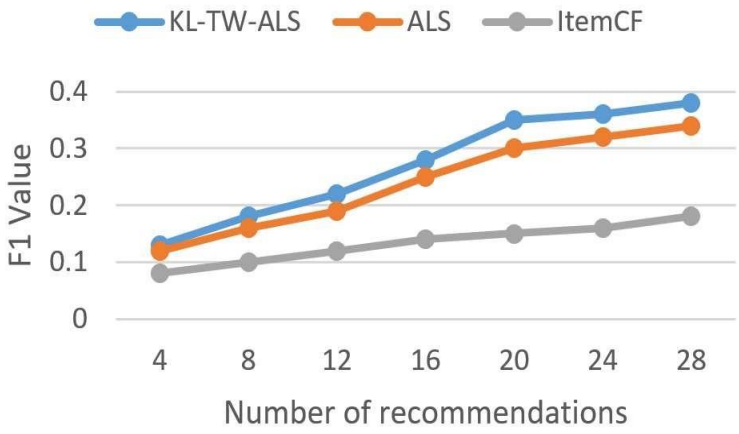

Fig. 5. Comparison chart of the results of F1 value under the change of recommended quantity

From the above three sets of graphs, it can be seen that the item-based collaborative filtering algorithm (Item CF) and the ALS-based collaborative filtering algorithm have significantly lower accuracy, recall, and F1 value than the KLTW-ALS recommendation algorithm, which further indicates that the recommendation algorithm proposed in this paper has better recommendation effect.

V. CONCLUSIONS

In this paper, an ALS recommendation algorithm (KLTW-ALS) based on KL divergence and time weighting is proposed to address the problems of data dispersion and recommendation accuracy. Use KL divergence to mitigate the data sparsity problem. A time decay factor is also introduced to assign time weights to improve the recommend dation accuracy. This paper also use the item-based collaborative filtering algorithm (Item CF), ALS-based collaborative filtering algorithm and KL-TW-ALS to do comparison experiments on Spark platform. The results of the experiments show that the algorithm is optimized for better recommend dation accuracy. However, the algorithm does not consider the implicit behavior of users and text comments, and the subsequent work will use the method of converting the implicit behavior of users into a rating matrix and fusing the text features of comments for recommendation.

REFERENCES

1. Yang-Yong Zhu, Jing Sun. Advances in recommendation system research[J]. Computer Science and Exploration,2015,9(05):513-525.
2. Aaron Kimball, Sierra Michels-Slettvet, Christophe Bisciglia. Cluster computing for web-scale data processing [J]. ACM SIGCSE Bulletin,2008,40(1).
3. Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung. The Google file system[J]. ACM SIGOPS Operating Systems Review,2003,37 (5).
4. David M. Blei, Andrew Y. Ng, Michael I. Jordan. Latent Dirichlet Allocation [J]. Journal of machine learning research,2003,3(4/5).
5. Yang Yang, Xiang Yang, Xiong Lei. Collaborative filtering recommendation algorithm based on matrix decomposition and user nearest neighbor model[J]. Computer Applications, 2012,32(02):395398.
6. HU, YIFAN, KOREN, YEHUDA, VOLINSK Y, CHRIS. Collaborative Filtering for Implicit Feedback Datasets[C]. Data Mining, ICDM, 2008 8th IEEE International Conference on; Pisa, Italy. 2008:263-272.
7. Yuliang Chi. Research on combined recommendation algorithm based on Spark platform[D]. Qufu Normal University,2018.
8. Zaharia M, Chowdhury M, Franklin M J, et al. Spark: Cluster computing with working sets[J]. In: Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing. HotCloud'10. 2010.
9. Zhao NA Yu, Zhong Jie Wang. Research on collaborative filtering algorithm based on Spark[J]. System Simulation Technology,2016, 12(1):40-45.
10. Xuedong Xu, Xiaodong Liu . Collaborative filtering recommendation algorithm based on time-weighted ALS model[J]. Electronic Design Engineering,2022,30(14):39-43.
11. Lee Y, Lee Y. Toward scalable internet traffic measurement and analysis with hadoop[J]. ACM SIGCOMM Computer Communication Review, 2012, 43(1): 5-13.
12. Yuanda Han. ALS recommendation algorithm based on KL scatter[J]. Computer Knowledge and Technology,2022,18(12):4-6.
13. Jesús Bobadilla, Fernando Ortega, Antonio Hernando, Jesús Bernal. A collaborative filtering approach to mitigate the new user cold

start problem [J]. Knowledge-Based Systems, 2011,26:225-238

14. Patra B K, Launonen R, Ollikainen V, et al. Exploiting Bhattacharyya similarity measure to diminish user cold-start problem in sparse data[C] Discovery Science: 17th International Conference, DS 2014, Bled, Slovenia, October 8-10, 2014. Proceedings 17. Springer International Publishing, 2014: 252-263.