UIIM: A User-Item Interest Model Based on Bipartite Network for Top-N Recommender System

Zhixiong Jiang, Chunyang Lu, Siyuan Zheng, and Juan Yang

***Abstract*—Recently, a sparse linear method (SLIM) is developed for top-N recommender systems, which can produce high-quality recommendations for sparse data sets. SLIM provides a better performance than other existing methods. In this paper, we provide a novel user-item interest method (UIIM) based on bipartite network to improve the performance of SLIM. UIIM generates top-N recommendations by building the user-item interest matrix R with the bipartite network of users and items, calculating the item-item similarity matrix $\boldsymbol{W}$ with SLIM and predicting users' ratings on items as a dot product of matrix $\boldsymbol{R}$ and $\boldsymbol{W}$. And we also provide a parallel algorithm based on Spark to learn $\boldsymbol{W}$. Our results indicate that UIIM provides better performance and recommendation quality than other existing methods and parallel algorithm of learning $\boldsymbol{W}$ outperforms serial algorithm on large-scale data sets.**

***Index Terms*—Top-N recommender systems, bipartite network, UIIM, SLIM, parallel.**

I. Introduction

With the fast development of Internet and E-commerce, the frequency of purchasing products online grows rapidly. However, too many products make customers difficult to find favorite products. The problem becomes how to help customers find products that best fit their personal taste effectively. In particular, when given the purchasing history of users, a ranked list of items could be generated for users to choose from. This result in the widely use of top-N recommender systems.

Recent years, various algorithms have been developed for top-N recommender systems. These algorithms can be divided into two categories: neighborhood-based collaborative filtering methods and LFM methods (latent factor model). Neighborhood-based collaborative filtering methods are the most common algorithms in recommender systems. User-based collaborative filtering method, which is proposed in 1992, is one of the most common methods for recommender systems. Sarwar *et al.* proposed an item-based collaborative filtering method in 2001 [1]. Item-based collaborative filtering method is widely used in real-world recommender system like Amazon and Douban. Neighborhood-based collaborative filtering methods, particularly referred to as user/item-based collaborative filtering methods, compute similarities between users/items and recommend items with these similarities. The key idea of LFM methods is to build the user-item matrix based on the purchasing history of users, factorize it into (low-rank) user factors and item factors that represent the user interests and item features, and the users' ratings/tastes can be represent by the dot product of user factors and item factors. In [2], Chi-h Chao Ma introduced a simple Latent Factor Model called SVD, which simply generates recommendations as the dot product of user factors and item factors. A method called BiasSVD improves the performance of SVD by considering the differences of users' tastes and items' characters. LFM methods generate recommendations with high quality, but incur high cost meanwhile. On the other hand, neighborhood-based methods generate results very fast, but the quality is relatively lower.

Recently, a novel method named SLIM has been developed for top-N recommender systems, which combines the advantages of both neighborhood-based methods and LFM. SLIM recommends items for users by learning a sparse aggregation coefficient matrix $\boldsymbol{W}$ from user-item matrix [3], which represents similarities between items. The experiments in [3] show that SLIM achieves better performance and recommendation quality than the state-of-the-art methods. However, there is an inherent limitation of SLIM that during learning matrix $\boldsymbol{W}$, the similarity between two items can be learned only when they have been co-purchased/rated by at least some users. The result is that it cannot find the potential transitive relations between items [4].

In this paper, we propose a method called UIIM (User-Item Interest Model), which solves this problem based on the user-item bipartite network. We build the bipartite network from the user-item matrix, for items that have not been co-purchased by any users, there may be a positive similarity between them that denoted by the paths between them connected by users, UIIM can capture this similarity by traversing these paths, so we can build the transitive relationship between items that have not been co-purchased/rated by at least one user based on this bipartite network. And the experiments show that this method outperforms SLIM.

The rest of this paper is organized as follows: In Section II, a detailed introduction of the previous work is provided. In Section III, our method UIIM and a parallel algorithm for computing matrix $\boldsymbol{W}$ is described. In Section IV and Section V, the data for experiments and the result are presented.

II. Previous Work

A. Definitions and Notation

In this paper, users and items are described by symbols u and t, and for individual users and items, different subscripts

Manuscript received September 22, 2014; revised May 23, 2015.

Zhixiong Jiang and Chunyang Lu are with CNPC Changping Data Center, Beijing 102206, China (e-mail: jiangzhixiong@cnpc.com.cn, luchunyang@cnpc.com.cn).

Siyuan Zheng and Juan Yang are with Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: zsybupt@gmail.com, yangjuan@bupt.edu.cn).

will be used (i.e., u_i, t_j). The user-item purchasing matrix will be represented by $\boldsymbol{A}$, a_{ij} is 1 or 0 represents whether user u_i has purchased/rated item t_j or not, so $\boldsymbol{a}_i^T$ denotes the i-th row of $\boldsymbol{A}$, it represents the purchasing/rating history of user u_i on all items, $\boldsymbol{a}_j$ denotes the j-th column of $\boldsymbol{A}$, it represents the purchasing/rating history of all users on item t_j.

In this paper, all vectors (e.g., $\boldsymbol{a}_i^T$, $\boldsymbol{a}_j$) will be denoted by bold lower-case letters, and all matrices (e.g., $\boldsymbol{A}$) will be denoted by bold upper-case letters. The prediction of user u_i on item t_j will be denoted by having a ~ head, $\tilde{a}_{ij}$ represents the prediction of user u_i on item t_j, and $\widetilde{\boldsymbol{A}}$ denotes the user-item prediction matrix.

B. SLIM

The SLIM method developed by Ning [3] generates recommendations by learning a sparse coefficient matrix $\boldsymbol{W}$ which represents the item-item similarity matrix. Thus, the model utilized by SLIM can be presented as

$$\widetilde{\boldsymbol{A}} = \boldsymbol{AW} \tag{1}$$

SLIM estimates the sparse matrix $\boldsymbol{W}$ as the minimizer for the following regularized optimization problem:

$$\min_{\boldsymbol{W}} \frac{1}{2}\|\boldsymbol{A}-\boldsymbol{AW}\|_F^2 + \frac{\boldsymbol{\beta}}{2}\|\boldsymbol{W}\|_F + \boldsymbol{\lambda}\|\boldsymbol{W}\|_1$$
$$\text{subject to } \boldsymbol{W} \geq 0 \qquad \text{diag}(\boldsymbol{W}) = 0 \tag{2}$$

where $\|\boldsymbol{W}\|_1$ is the entry-wise l_1-norm of $\boldsymbol{W}$, and $\|\boldsymbol{W}\|_F$ is the matrix Frobenius norm. $\boldsymbol{AW}$ is the estimated matrix of recommendation scores (i.e., $\widetilde{\boldsymbol{A}}$), so the first term $\frac{1}{2}\|\boldsymbol{A}-\boldsymbol{AW}\|_F^2$ (i.e., the residual sum of squares) measures how well the sparse linear model fits the training data, $\|\boldsymbol{W}\|_F$ and $\|\boldsymbol{W}\|_1$ are l_F-norm and l_1-norm regularization terms, l_F-norm measures model complexity and prevents over fitting, and l_1-norm introduces sparsity to $\boldsymbol{W}$. The non-negativity constraint is applied so that $\boldsymbol{W}$ represents the positive relations between items. At last, the constraint diag($\boldsymbol{W}$) = 0 is applied to ensure that when calculate $\tilde{a}_{ij}$, a_{ij} doesn't make any contribution. In addition, diag($\boldsymbol{W}$) = 0 can avoid trivial solutions (i.e., the optimal W is an identical matrix).

Various experiments in [3] have showed that SLIM outperforms other existing methods for top-N recommender systems.

III. Algorithm

In this section, we will first introduce the inherent limitation of SLIM, then we will expand the solution of this limitation and the details of our method based on bipartite network. At last we will introduce a method of learning $\boldsymbol{W}$ in parallel based on Spark.

A. Limitation of SLIM

As mentioned in Section I, there is an inherent limitation of SLIM, SLIM can only learn the similarity of two items which at least be co-purchased/rated by one user, similarities between items which have not been co-purchased/rate by any users cannot be learned. Here is a concrete example for clearly understanding this limitation. We build the user-item bipartite network from user-item purchasing matrix. The circular shape represents users, and the triangle represents items, if user u_i has purchased/rated item t_j, there will be an edge between u_i and t_j.

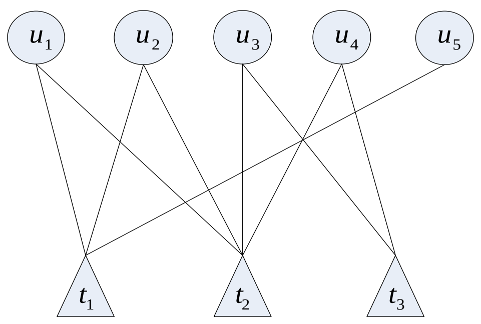

Fig. 1. User-item bipartite network.

As Fig. 1 shows, in this system, there are 5 users (u_1, u_2, u_3, u_4, u_5) and 3 items (t_1, t_2, t_3). The problem is that now we have to make recommendations (top-2) for u_5. Obviously, except u_1 and u_2, other users that have purchased item t_2 all purchased item t_3 meanwhile. That means t_3 has a high similarity to t_2, and except u_5, users (u_1, u_2) that have purchased item t_1 all purchased t_2, there is a high similarity between t_1 and t_2, too, so we can recommend t_2 and t_3 to u_5 based on the high similarities of (t_1, t_2) and (t_2, t_3). However, in this system, there is no user who has co-purchased item t_1 and item t_3, and the SLIM method cannot capture the potential relation between t_1 and t_3. After learning, the entry (1, 3) of the sparse coefficient matrix $\boldsymbol{W}$ will be 0 (w_{13}=0), therefore, if we use the SLIM method to generate recommendations for u_5, only t_2 will be recommended to u_5. But in fact, t_1 indeed has a high similarity to t_3, and t_3 should also be recommended to u_5.

In order to avoid this kind of situation, we propose a novel method based on bipartite network to solve this problem.

B. UIIM

The SLIM method learns a sparse coefficient matrix $\boldsymbol{W}$ as the item-item similarity matrix, and then calculates the user-item prediction matrix $\widetilde{\boldsymbol{A}}$ as a dot product of $\boldsymbol{A}$ and $\boldsymbol{W}$ with Equation (1). In order to solve the limitation mentioned above, UIIM constructs a user-item interest matrix $\boldsymbol{R}$ from $\boldsymbol{A}$ based on the user-item bipartite network, and the matrix $\widetilde{\boldsymbol{A}}$ will be the dot product of matrix $\boldsymbol{R}$ and $\boldsymbol{W}$.

$$\widetilde{\boldsymbol{A}} = \boldsymbol{RW} \tag{3}$$

The user-item interest matrix $\boldsymbol{R}$ represents users' interests in items learned from user-item bipartite network. In the user-item bipartite network, all users and items are regarded as vertexes. If user u_i has purchased item t_j, there will be an edge between u_i and t_j, and the length of each edge is 1. In this bipartite network, paths between users and items indicate the interest of users in items. For a certain user u_i and an item t_j, if u_i has a high interest in t_j, paths between them will have characters below,

1) There will be a lot of paths between them.
2) Lengths of paths between them are short.
3) Paths between them do not contain vertexes of big degrees.

According to this, first of all, we check whether a_{ij} is 1 or not, if a_{ij} is 1, it means that u_i has purchased t_j before, and we don't need to compute r_{ij} again, the value of r_{ij} is 1 in this situation. On the hand, we start at vertex u_i to find paths to vertex t_j based on depth first searching method, if there is a path p_k from u_i to t_j, it indicates that u_i has a positive interest in t_j denoted by I_k which is initialized with 1 before we start to search p_k, when we arrive a vertex v_i in p_k, we multiply I_k by the reciprocal of the degree of v_i to update it, and r_{ij} will be the sum of I_k, k ranges from 1 to the number of paths between u_i and t_j. From the above, we propose formula (4) to compute r_{ij}.

$$r_{ij} = \begin{cases} \sum_{k=1}^{p} \prod_{v \in V_k} \frac{1}{d_v} & a_{ij} = 0; \\ 1 & a_{ij} = 1; \end{cases} \tag{4}$$

where p is the number of paths that start from vertex u_i to vertex t_j, and V_k is the set of vertexes in path p_k, d_v is the degree of vertex v, and in order to terminate the process of searching paths from u_i to t_j, we set a maximum iterations to stop it.

TABLE I: USER-ITEM INTEREST MATRIX

$\boldsymbol{R}$	t_1	t_2	t_3
u_1	1	1	1/16
u_2	1	1	1/16
u_3	1/8	1	1
u_4	1/8	1	1
u_5	1	1/3	0

As the system that Fig. 1 described, Table I shows the user-item interest matrix $\boldsymbol{R}$ after the fourth iteration, the entry (5, 2) r_{52} in $\boldsymbol{R}$ is a positive number, and since t_2 and t_3 have been co-purchased by users, the entry (2, 3) w_{23} in $\boldsymbol{W}$ is a positive number, too. Therefore, the score $\tilde{a}_{53}$ which is the product of r_{52} and w_{23} will be positive and t_3 will be recommended to u_5.

Comparing to the user-item purchasing matrix $\boldsymbol{A}$, besides the existing purchasing records, the user-item interest matrix $\boldsymbol{R}$ enriches the potential interest of users in items, and as links between items, these potential interests can help find the transitive relations between items that have not been co-purchased by users when generating recommendations by compute $\boldsymbol{RW}$.

C. Parallel Algorithm for Computing $\boldsymbol{W}$

Both UIIM and SLIM have to learn $\boldsymbol{W}$ with Equation (1), a SGD (Stochastic Gradient Descent) method is introduced in [5] which has achieved less time of computing $\boldsymbol{W}$ than [3]. But when the scale of the data becomes larger, it is difficult for serial algorithm to learn $\boldsymbol{W}$ with a larger amount of computing, so we provide a parallel algorithm to compute $\boldsymbol{W}$ based on Spark with Batch Gradient Descent method. Serial algorithm learns each column of $\boldsymbol{W}$ with Equation (5):

$$\min_{w_i} \frac{1}{2} \|\boldsymbol{a_i} - \boldsymbol{A}\boldsymbol{w_i}\|_F^2 \tag{5}$$

Assume that the size of $\boldsymbol{A}$ is $m \times n$ and the iterative time is k, for each column of $\boldsymbol{W}$, we have to go through rows of $\boldsymbol{A}$ to compute $\boldsymbol{a_i}(\boldsymbol{a_i} - \boldsymbol{Aw_i})$ of k-time iteration with serial algorithm, the complexity is O(kmn^3). And with Spark, we can compute $\boldsymbol{a_i}(\boldsymbol{a_i} - \boldsymbol{Aw_i})$ in parallel as Fig. 2 and Table II show, the complexity can be reduce to O(kn^3).

TABLE II: PARALLEL ALGORITHM FOR $\boldsymbol{W}$

```
Algorithm of learning $\boldsymbol{W}$
for each  column $\mathbf{w}_i$ of $\boldsymbol{W}$
{
    for q from 1 to k // iterative time
    {
        computing gradient descent
direction  $\boldsymbol{a_i}(\boldsymbol{a_i} - \boldsymbol{Aw_i})$ in parallel
    }
    update $\mathbf{w}_i$
}
```

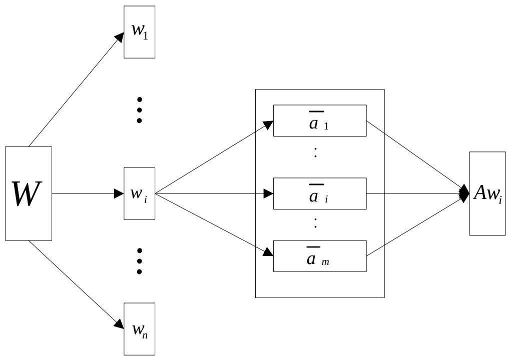

Fig. 2. Compute $\boldsymbol{Aw_i}$ in parallel.

IV. EXPERIMENT

In this section, we evaluated the performance of UIIM and the performance of the parallel algorithm of learning matrix $\boldsymbol{W}$ on different scale data sets.

A. Data Set

We evaluated the performance of UIIM on a benchmark data set in Table III, namely, Movielens [6], and we split these three data sets with program provided by [7].

TABLE III: EXPERIMENTAL DATA SETS

dataset	Users	items	ratings
ML100K	943	1682	100000
ML1M	6040	3952	1000209
ML10M	71567	10677	10000054

UIIM aims to improve the limitation of SLIM that similarities between two items can be learned only when they have been co-purchased by a user, and on ML100K, 81.9% pair of items have not been co-purchased by any user. It indicates that ML100K data set can measure the performance of UIIM effectively.

And we compared the cost of serial algorithm of computing $\boldsymbol{W}$ with parallel algorithm on the three data sets above. To evaluate our algorithm, we conducted performance experiments based on Spark (version 1.0.0), Spark platform is deployed on nine Huawei RH2285 2U servers, including eight worker nodes and one master node. Each node has Intel (R) Xeon (R) CPU E5530 @ 2.40GHz dual-processor, 48GB RAM, 4T hard drive.

B. Evaluation Methodology

In this paper, we evaluate the quality of recommendations with the Precision of the algorithm. Assume that $P(u)$ is the set of items that recommended for user u, $T(u)$ is the set of items user u has purchased in the test data. U is the set of users, and then the Precision can be described as:

$$\text{Precision} = \frac{\sum_{u\in U}|P(u)\cap T(u)|}{\sum_{u\in U}|P(u)|} \quad (6)$$

$p@k$ means that the precision of the recommender system when it recommends a ranked list of items with a size of k to each user in the system. In order to evaluate a recommender system comprehensively, different value of k is proposed to calculate the precision. And we also compare computing time of learning $\boldsymbol{W}$ of the serial algorithm with the parallel algorithm on three different data sets above.

V. Conclusion

We implement the SLIM with SGD (Stochastic Gradient Descent) method mentioned in [5], the UIIM method described above, the parallel algorithm of learning matrix $\boldsymbol{W}$. And the result of itemKNN method is from Table II in [5].

Table IV shows the Precision of three different algorithms on ML100K data set, we can observe that the UIIM method outperforms over SLIM and itemKNN method with different value of k, and UIIM achieves a better precision than SLIM and itemKNN comprehensively.

Fig. 3 shows the time of computing $\boldsymbol{W}$ of serial algorithm and parallel algorithm based on Spark with three data sets above. Fig. 3 indicates that on ML10M data set, serial algorithm learns $\boldsymbol{W}$ with 91.6 minutes, while parallel algorithm based on Spark speeds up the computing time to 49.3 minutes over serial algorithm. Parallel algorithm achieves a better performance than serial algorithm with a large-scale data.

TABLE IV: The Precision of Different Algorithms

ML100K	p@5	p@10	p@15	p@20
itemKNN	0.350	0.296	0.267	0.246
SLIM	0.340	0.306	0.280	0.263
UIIM	0.398	0.350	0.322	0.299

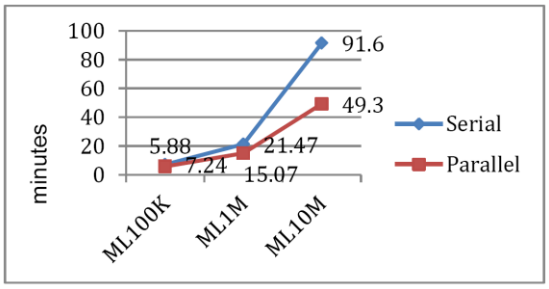

Fig. 3. Time of computing $\boldsymbol{W}$.

References

[1] B. Sarwar, G. Karypis, J. Konstan *et al*., "Item-based collaborative filtering recommendation algorithms," in *Proc. the 10th International Conference on World Wide Web*, ACM, 2001, pp. 285-295.
[2] C. C. Ma, "A guide to singular value decomposition for collaborative filtering," 2008.
[3] X. Ning and G. Karypis, "Slim: Sparse linear methods for top-n recommender systems," in *Proc. 2011 IEEE 11th International Conference on Data Mining*, IEEE, 2011, pp. 497-506.
[4] S. Kabbur, X. Ning, and G. Karypis, "FISM: Factored item similarity models for top-N recommender systems," in *Proc. the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2013, pp. 659-667.
[5] M. Levy and K. Jack, "Efficient top-N recommendation by linear regression," in *Proc. Large Scale Recommender Systems Workshop in RecSys'13*, 2013.
[6] MovieLens. [Online]. Available: http://grouplens.org/datasets/movielens/
[7] GitHub. [Online]. Available: https://github.com/Mendeley/mrec

Zhixiong Jiang is a deputy chief engineer of Changping Data Center and a technical director of cloud computing project of China National Petroleum Corporation. He obtained his M.E degree in computer application from the University of Shanghai for Science and Technology and the PhD degree in computer software from Fudan University, China. His research interests are in the area of cloud computing and system architecture.

Chunyang Lu is the deputy director of Changping Data Center of Bureau of Geophysical Prospecting INC., China National Petroleum Corporation and the deputy general manager of CNPC Beijing Richfit Information Technology Co., Ltd. He obtained his M.E degree in software engineer from Beihang University. His research interests are in the area of construct and operation the cloud computing center.

Siyuan Zheng is a student with the School of Computer in Beijing University of Posts and Telecommunications, his major is computer science. His research area is data mining and recommender system.

Juan Yang is an associated professor of Beijing University of Posts and Telecommunication, China. She obtained her M.E degree in computer application and the PhD degree in communication and information system from Beijing University of Posts and Telecommunication, China. Her research interests are in the area of cloud computing, intelligent information processing and communication software.