ISBN 978-981-14-1684-2
Proceedings of 2019 the 9th International Workshop on Computer Science and Engineering
(WCSE 2019 SUMMER)
Hong Kong, 15-17 June, 2019, pp. 820-828
doi: 10.18178/wcse.2019.06.122

Enhancing Online Collaborative Filtering by Integrating Social Network

Shaobin Lu [1], Guilin Li [2 +]

[1] Xiamen University, Xiamen, China

[2] Research Center on Mobile Internet Technology, Software School of Xiamen University, Xiamen City, 361024, China

Abstract. In recent years, researchers have done a lot of work to enhance online collaborative filtering (OCF) performance. Compared to most of the offline collaborative filtering (offline CF), the online collaborative filtering algorithm has three advantages: the low cost of retraining the model, dynamically tracing the user behavior habits and capturing the change of the item popularity. Many OCF algorithms extract user interests and item popularity features by updating algorithms model in time. But most of OCF ignore the similarity of users or items by updating all users' features or all items' features. In this study, we aim to integrate social network to improve the OCF performance. In order to achieve the goal, we propose two new methods by introducing user similarity which obtains from user social network to online collaborative filtering based on the Probabilistic Matrix Factorization (PMF) frame. One of the methods, which we called OCFUSim_I, is to calculate the similarity of users and find the neighbors of user, then adding the neighbors to the OCF. Another method, called OCFUSim_II, is to add similarity among users to OCF model. We conduct the experiments on three public datasets: MovieLens100K, MovieLens1M and HetRec2011 datasets. The experimental results show that our algorithms achieve better performance than several baseline approaches.

Keywords: Online Collaborative Filtering, Social Network, User Similarity, Recommend System

1. Introduction

With the rapid explosion of big data, all trades and professions have accumulated of sufficient information and data. Now, more and more scientific and technical corporations adopt recommendation technology to help user make quick and reasonable decisions. So, it is getting more and more important for recommend system to precisely predict users' interest and behavior [1].

The most significant algorithms to make more precise predict is collaborative filtering (CF). The CF model mainly predict the rating of user on items by matrix factorization (MF) [2]. The key idea of the MF model is to decompose the rating matrix into user features matrix and item features matrix. Of course. With the development of MF, many improved algorithms based on MF have been proposed, such as using knowledge graph embedding [3] and integrating the social network [4].

However, traditional CF whose updating method are based on batch learning have some disadvantages. Firstly, the excessive expense caused by batch leaning make the CF model retraining slowly [5]. Secondly, traditional CF algorithms are not adept in processing dynamic data [6]. But as we all known, almost the rating system will join new user and new item in every moment. Thirdly, traditional CF are hard to catch the transfer of the user's interest in time [7]. Finally, the traditional CF can not update user features and item features in time when user evaluate some new items [8].

+ Corresponding author.
E-mail address: glli@xmu.edu.cn

To improve these defects above, a lot of OCF methods were proposed. [9] design a new framework for online social recommendation, which incorporated the relationship of user-item into user preference features leaning process. [10] take advantage of online multitask learning to solve the problems of traditional CF. [11] achieve the task by solving the second order confidence to capture the latest change of user interest.

How to update user preference features dynamically in time is becoming more and more important [12]. Therefore, we take the user's neighborhood interest into the OCF from the perspective of Probabilistic Matrix Factorization (PMF). We assume that users with the same preferences will have similar interest changes. By capturing a user's interest changes, the algorithm OCFUSim_I we proposed will update the neighbor interest features of the user. But preferences among users cannot be completely identical. In order to introduce similarity degree, we present OCFUSim_II which added the similarity of users as weight to improve the performance.

We conduct some experiments on three real-world datasets: MovieLens100K, MovieLens1M and HetRec2011 to achieve lower prediction error, i.e. RMSE, and converge faster than other baseline approaches.

2. Background and Related Work

This Section mainly reviews probabilistic matrix factorization as one of the collaborative filtering methods. Next, we mainly elaborate the online collaborative filtering to attempt to explain the relationship of PMF and OCF.

2.1. Probabilistic matrix factorization and problem setting

The commonly used methods for recommendation systems are collaborative filtering (CF). the CF could break down into two main families: model-based CF and memory-based CF [13]. model-based CF recommend items mainly based on build user rating model and predict the user rating on a special item [14]. Memory-based CF predict the user rating based on the rating matrix [15]. Classically, the model-based CF approaches are more precisely than memory-based approaches, because the memory-based CF cannot solve the sparsity problems of data [16].

PMF is an excellent model of collaborative filtering in recommend system [17]. It is a probabilistic linear model with Gaussian observation noise [18]. It decomposed the conditional distribution rating data into latent user features and latent item features. In the features space, the PMF calculate the product of user features and item features to predict the unknown rating.

Let us briefly review the problem setting of PMF. Given a rating matrix $R \in \mathbb{R}^{M\times \mathrm{N}}$, where M is the number of users and N is the number of items. R_{ij} denotes the rating of user i on item j. The latent user matrix is denoted as $U, where\ U \in \mathbb{R}^{K\times \mathrm{M}}$ and K denotes the number of features. The latent item matrix represent with $V, where\ V \in \mathbb{R}^{K\times \mathrm{N}}$ and K denotes the number of features. The PMF model defined the conditional distribution of rating data as:

$$\mathrm{P}(R|U,V,\sigma^2) = \prod_{i=1}^{M}\prod_{j=1}^{N}\left[N\left(R_{ij}\middle|U_i^T V_j,\sigma^2\right)\right]^{I_{ij}} \tag{1}$$

Where $U_i\ and\ V_j$ present the feature of ith user and jth item respectively. And $\mathrm{N}(x|\mu,\sigma^2)$ is the Gaussian distribution with mean μ and variance σ^2. In the formula (1), I_{ij} is the indicator function. If the I_{ij} equals to 1, it denote ith user rated the jth item. If the I_{ij} equals to 0, it denote ith user did not rate the jth item. We assume that $U_i \sim \mathrm{N}(0,\sigma_u^2 I)$, $V_j \sim \mathrm{N}(0,\sigma_v^2 I)$, and the calculation of user factor and item factor are shown as the formula (2) ,(3).

$$\mathrm{P}(U|\sigma_u^2) = \prod_{i=1}^{M} N(U_i|0,\sigma_u^2 I) \tag{2}$$

$$\mathrm{P}(V|\sigma_v^2) = \prod_{i=1}^{N} N(V_j|0,\sigma_v^2 I) \tag{3}$$

The objective function of PMF is defined as follow:

$$L = \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}\left(R_{ij} - U_i^T V_j\right)^2 + \frac{\lambda_u}{2}\sum_{i=1}^{M}\|U_i\|_F^2 + \frac{\lambda_v}{2}\sum_{j=1}^{N}\left\|V_j\right\|_F^2 \tag{4}$$

Where $\lambda_u = \frac{\sigma^2}{\sigma_u^2}, \lambda_v = \frac{\sigma^2}{\sigma_v^2}$, and $\|\cdot\|_F^2$ denotes the Frobenius norm.

The objective of the collaborative filtering is to minimize the loss function. Then we can evaluate the performance of the PMF by Root Mean Square Error (RMSE) which is defined as the formula (5).

$$\text{RMSE} = \sqrt{\frac{1}{|D|}\sum_{(i,j)\in D}\left(R_{ij} - \hat{R}_{ij}\right)^2} \tag{5}$$

Where R_{ij} denotes the observed rating, and $\hat{R}_{ij}$ denotes the predicted rating. D is the set of all observed ratings, and $|D|$ is the number of the set D.

To optimize the RMSE metric, the loss of this model is defined as follows:

$$L_1 = \left(R_{ij} - U_i^T V_j\right)^2 \tag{6}$$

2.2. OCF methods

Online collaborative filtering which update the model in real-time has advantages comparing with traditional collaborative filtering. The advantages include the low cost of retraining model and capture the changes of user interest and item popularity [19]. Because the OCF method training the model by updating the features when the user-item rating changed [20].

Recently, [8] proposed a confidence weighted model based on the OCF. This model combined the confidence weight into the OCF to reduce the cost of re-training and improve the performance of prediction. [7] present a first order sparse collaborative filtering and second order sparse collaborative filtering to solve the problem of data sparsity.

Meanwhile, OCF is used in many fields. For example, [21] proposed a OCF model which is based on probabilistic topic model to push science and technology articles to users. [22] adopted the OCF with the social network information to help the user to find others with the same interests.

The model of online collaborative filtering is working as follows [23]: Given a prediction matrix $R \in U^T V$, where $U \in \mathbb{R}^{K\times \mathrm{M}}$and $V \in \mathbb{R}^{K\times \mathrm{N}}$. Given a single observation $(\mathrm{a}, \mathrm{b}, \mathrm{r})$, then we can confirm the loss on this single observation, which is shown in the formula (7).

$$\mathrm{L}\big(\mathrm{R}, (\mathrm{a}, \mathrm{b}, \mathrm{r})\big) = (r - R_{ab})^2 = (r - \textstyle\sum_{l=1}^{K} U_{al} V_{bl}) \tag{7}$$

Next, we can differentiate with respect to U and V. The differentiating is given in formula (8) and (9).

$$\frac{\partial \mathrm{L}}{\partial U_{al}} = -2V_{bl}(r - \textstyle\sum_{l=1}^{K} U_{al} V_{bl}) \tag{8}$$

$$\frac{\partial \mathrm{L}}{\partial V_{bl}} = -2U_{al}(r - \textstyle\sum_{l=1}^{K} U_{al} V_{bl}) \tag{9}$$

Finally, the update method of online CF would become the form as follows:

$$U_{al} \leftarrow U_{al} + 2\tau V_{bl} \cdot sign(r - r')$$
$$V_{bl} \leftarrow V_{bl} + 2\tau U_{al} \cdot sign(r - r')$$

Where r' is the predict rating and $sign()$ is an indicator function.

3. Enhancing Collaborative Filtering by Integrating Social Network

This section mainly presents our novel algorithms for OCF with social network (OCFUSim_I). The core function of the algorithm we proposed is to add similarity of users of their social network into CF for improving the accuracy of OCF. Also we improve the previous algorithm by introducing the similarity between users as weight to gain better accuracy. And the second novel algorithm is called OCFUSim_II.

3.1. Online collaborative filtering by integrating social network

Use Nearest Neighbor Information has a significant effect on collaborative filtering algorithm [24]. [25] proposed to combine the adaptive neighborhood with the temporal CF, but they only considered the varying size of neighborhood over time. As we all know, users with similar interests are likely to have similar shopping preferences. For example, people who have a preference for the same type of film are likely to be interested in the same actors at the same time. So, the social information is significant important for the matrix factorization.

In order to take the social network into account, we propose a novel method based on OCF and social network, which we called OCFUSim_I. The main steps of the algorithm contain three parts: first, we need to

find the neighbors of each user when we set the similarity less than threshold. Second, we need to confirm the loss function and use stochastic gradient descent (SGD) to optimize the loss function. Third, we should update the user features and item features at each round.

For computing the similarity between users, we adopt the formula (10) to evaluate the similarity of two users u *and* f.

$$sim(u,f) = \frac{|I_u \cap I_f|}{|I_u \cup I_f|} \tag{10}$$

Where I_u and I_f denote the set of user u and f have rated, respectively.

From the probability matrix factorization perspective, the PMF model is defined by the conditional distribution of rating data as formula (1). Then, we add the similarity between users into objection function (4), the objection function defined as follows:

$$L2 = \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}\left(R_{ij} - U_i^T V_j\right)^2 + \frac{\lambda_u}{2}\sum_{i=1}^{M}\|U_i\|_F^2 + \frac{\lambda_v}{2}\sum_{j=1}^{N}\|V_j\|_F^2 + \frac{\gamma}{2}\left\|U_i - \frac{1}{|D|}\sum_{f\in D} U_f\right\|_F^2 \tag{11}$$

Where D denotes the set of neighborhood of user U_i. And U_f is one of U_i neighborhood.

$\lambda_u = \frac{\sigma^2}{\sigma_u^2}, \lambda_v = \frac{\sigma^2}{\sigma_v^2}$, and $\|\cdot\|_F^2$ denotes the Frobenius norm.

We use SGD to optimize Eq. (11):

$$U_i = (1 - \eta\lambda_u)\cdot U_i + \eta e_{ij}\cdot V_j + \gamma\left\|U_i - \frac{1}{|D|}\sum_{f\in D} U_f\right\| \tag{12}$$

$$V_j = (1 - \eta\lambda_v)\cdot V_j + \eta e_{ij}\cdot U_i \tag{13}$$

Where $e_{ij} = R_{ij} - \hat{R}_{ij}$, η is learning rate. When receiving a new rating, the model will make prediction: $\hat{R}_{ij} = U_i^T V_j$. Algorithm 1 shows the detailed algorithmic procedure of the OCFUSim_I.

Algorithm 1 OCFUSim_I for OCF

Parameters: k, λ_u, λ_v, γ, η

Input: a sequence of rating pairs (i, j, r_{ij})

01: Initialization: initialize a random matrix for $\boldsymbol{U} \in \boldsymbol{R^{k\times M}}$, $\boldsymbol{V} \in \boldsymbol{R^{k\times N}}$, respectively

02: Record the neighborhoods of U_i by computing the sim(U_i, U_f) when sim(U_i, U_f) > 0.70,0.75,0.80,0.85,0.90.

03: For $\mathbf{t} = \mathbf{1}, \mathbf{2}\cdots, \mathbf{T}$ do

04: Receive rating prediction request of user $\boldsymbol{i}$ on item $\boldsymbol{j}$

05: Make prediction $\boldsymbol{\hat{R}_{ij} = U_i^T V_j}$

06: The true rating $\boldsymbol{R_{ij}}$ is revealed

07: The algorithm suffers a loss $\boldsymbol{L2(U_i, V_j, \gamma, \eta)}$

08: Update $\boldsymbol{U_i, V_j}$ according to: (2), (3) respectively

09: **End for**

3.2. Online collaborative filtering by introducing similarity weight between users

This subsection mainly presents our second novel algorithm for OCF with similarity as weight to improve the previous algorithm. The second algorithm named OCFUSim_II. In the previous algorithm, we just record the neighborhoods of user and make use of the average value of similarity about all neighborhoods. Maybe we can add the similarity between users as the weight into the objective function. So the new objective function defined as follows:

$$L3 = \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}\left(R_{ij} - U_i^T V_j\right)^2 + \frac{\lambda_u}{2}\sum_{i=1}^{M}\|U_i\|_F^2 + \frac{\lambda_v}{2}\sum_{j=1}^{N}\|V_j\|_F^2 + \frac{\gamma}{2}\sum_{f\in D} sim(U_i, U_f)\left\|U_i - U_f\right\|_F^2 \tag{14}$$

Where D denotes the set of neighborhood of user U_i. And U_f is one of U_i neighborhood.

$\lambda_u = \frac{\sigma^2}{\sigma_u^2}, \lambda_v = \frac{\sigma^2}{\sigma_v^2}$, and $\|\cdot\|_F^2$ denotes the Frobenius norm.

We use SGD to optimize Eq. (14):

$$U_i = (1 - \eta\lambda_u) \cdot U_i + \eta e_{ij} \cdot V_j + \gamma \cdot \sum_{f \in F_u} sim(U_i, U_f) \cdot U_f \quad (15)$$

$$V_j = (1 - \eta\lambda_v) \cdot V_j + \eta e_{ij} \cdot U_i \quad (16)$$

Where $e_{ij} = R_{ij} - \hat{R}_{ij}$, η is learning rate, $sim(U_i, U_f)$ is the similarity between U_i, U_f. When receiving a new rating, the model will make prediction: $\hat{R}_{ij} = U_i^T V_j$. Algorithm 2 shows the detailed algorithmic procedure of the OCFUSim_II.

Algorithm 2 OCFUSim_II for OCF

Parameters: k, λ_u, λ_v, γ, η

Input: a sequence of rating pairs (i, j, r_{ij})

01: Initialization: initialize a random matrix for $\boldsymbol{U} \in \boldsymbol{R^{k \times M}}$, $\boldsymbol{V} \in \boldsymbol{R^{k \times N}}$, respectively

02: Record the neighborhoods of U_i by computing the sim(U_i, U_f) when sim(U_i, U_f) > 0.70,0.75,0.80,0.85,0.90.

03: For $\mathbf{t} = \mathbf{1}, \mathbf{2} \cdots, \mathbf{T}$ do

04: Receive rating prediction request of user $\boldsymbol{i}$ on item $\boldsymbol{j}$

05: Make prediction $\widehat{\boldsymbol{R}}_{\boldsymbol{ij}} = \boldsymbol{U_i^T V_j}$

06: The true rating $\boldsymbol{R_{ij}}$ is revealed

07: The algorithm suffers a loss $\boldsymbol{L3}(\boldsymbol{U_i}, \boldsymbol{V_j}, \boldsymbol{\gamma}, \boldsymbol{\eta})$

08: Update $\boldsymbol{U_i}, \boldsymbol{V_j}$ according to: (3), (4) respectively

09: End for

4. Experiments

This part mainly presents the experiment results of our two novel algorithms. We conduct the experiment on three public datasets and compare with four baseline approaches.

4.1. Datasets

MovieLens100K, MovieLens1M and HetRec2011 are three well-known public datasets which is open for researchers on the MovieLens website1. The table 1 shows the details of the three datasets.

Table 1 The detail information of datasets

Datasets	# Ratings	#Users	#Items	density
MovieLens100K	100,000	943	1,682	6.3%
MovieLens1M	1,000,209	6,040	3,900	4.2%
HetRec2011	855,598	2,113	10,109	4.0%

4.2. Baseline approaches

We make experiments on three public datasets compared with four baseline approaches. These baseline approaches include OLR, SOCF_II and DAPMF. Detailed explanation of these methods are shown as follow:

- OLR: the OLR algorithm adopts online low-rank approximation to learn users features and item features. Its parameter optimization employs the online descent method to gain best resolution [23];
- SOCF_II: second-order sparse OCF, which estimate the user-item distribution and take full account of large latent factors to objective function [12];
- DAPMF: a dual-averaging accelerated online learning framework for CF, which adopts the improved mini-batch accelerated approach to improve convergence rate [11];

4.3. Results and analysis

Firstly, we set the $sim(U_i, U_f)$ > threshold to find neighborhoods of user U_i. the value of threshold is 0.70,0.75,0.80,0.85,0.90. as shown in the table 2, when the threshold is equals to 0.85, the performance of our first algorithm are best in different feature K. Because in the objective function it equals to compute the average value of the similarity to the user features. If the users' neighborhoods are too much, users with different preferences will pull down the performance of the algorithm. In contrast, if the neighborhoods are not enough, the algorithm could not capture the neighbor effect.

Table 2 The average performance of OCFUSim_I on different threshold

	MovieLens100k			MovieLens1M			HetRec2011		
threshold	K=3	K=9	K=15	K=3	K=9	K=15	K=3	K=9	K=15
0.70	1.1025	1.0643	1.0921	1.0532	1.0331	1.0497	1.3232	1.0648	1.2997
0.75	1.0843	1.0432	1.0785	1.0321	1.0135	1.0387	1.1024	1.0201	1.1109
0.80	1.0675	1.0201	1.0696	1.0164	0.9843	1.0101	0.9787	0.9597	0.9879
0.85	**1.0426**	**0.9971**	**1.0602**	**0.9745**	**0.9482**	**0.9992**	**0.8812**	**0.8641**	**0.9154**
0.90	1.0545	1.0322	1.0623	0.9986	0.9645	1.0067	0.9886	0.9493	1.0102

Secondly, we choose the OCFUSim_I which threshold = 0.85 to compare with OLR, DAPMF, SOCF_II and OCFUSim_II. Table 3, table 4 and table 5 present all algorithms' average performance on three public datasets. The bold values in the tables are the best RMSE among all methods in the same K. Compared with other algorithm, we can find that our algorithm perform better in most cases. It shows that the neighbor information is effective to improve the accuracy of the OCF. Also, we can find that the OCFUSim_II is better than OCFUSim_I mostly which indicates that the similarity weight effectively enhanced the ability of the algorithm. This is because the similarity weight enlarged the impact of similar users and reduced the impact of users that are not very similar.

Table 3 The average performance on MovieLens100K

Methods	K=3	K=5	K=7	K=9	K=11	K=13	K=15
OLR	1.23550	1.12384	1.04829	1.01262	1.02432	1.04998	1.08034
DAPMF	1.04966	1.01874	1.00062	1.01002	1.02533	1.04754	1.06717
SOCF_II	1.09795	1.03982	0.99527	0.99973	1.02123	1.04458	1.06646
OCFUSim_I	1.04265	1.00182	0.99024	0.99715	1.02056	1.04323	1.06024
OCFUSim_II	**1.02892**	**0.99172**	**0.98576**	**0.99398**	**1.01228**	**1.03843**	**1.05868**

Table 4 The average performance on MovieLens1M

Methods	K=3	K=5	K=7	K=9	K=11	K=13	K=15
OLR	1.23427	1.11645	1.03841	0.99094	0.97703	0.99178	1.00754
DAPMF	1.02256	1.00132	0.98087	0.96989	0.98997	1.00982	1.02318
SOCF_II	1.06034	1.00587	0.96988	0.95097	0.96032	0.97756	0.99717
OCFUSim_I	0.97456	0.95238	0.94505	0.94824	0.95932	0.97234	0.99927
OCFUSim_II	**0.96905**	**0.94066**	**0.94233**	**0.94688**	**0.95807**	**0.97183**	**0.98958**

Table 5 The average performance on HetRec2011

Methods	K=3	K=5	K=7	K=9	K=11	K=13	K=15
OLR	0.95958	0.9076	0.87647	0.87907	0.89954	0.92143	0.94289
DAPMF	0.92222	0.89001	0.87643	0.87903	0.89961	0.92259	0.94885
SOCF_II	0.90601	0.87061	0.86123	**0.85987**	**0.86733**	**0.87893**	**0.89311**
OCFUSim_I	0.88121	0.86753	0.86243	0.86413	0.87986	0.89982	0.91543
OCFUSim_II	**0.87102**	**0.86035**	**0.85949**	0.86838	0.88242	0.88872	0.89638

Thirdly, we depict the performance of all methods on MovieLens100K and MovieLens1M when the latent factor K varies from 3 to 15. As is shown in the Fig.1, The curve of our algorithms are relatively gentle. the OLR is steep. It is due to the OLR adopted online low-rank approximation to learn users features and item features. When the latent feature is too small, it cannot capture user features. But because our algorithms combined the user neighbor information. They can capture user features.

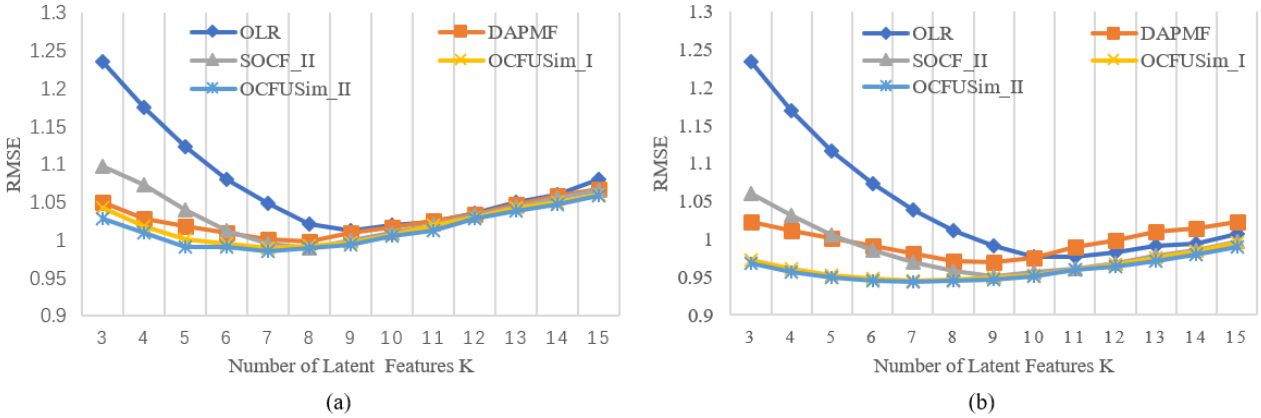

Figure 1: (a) The performance of all methods on MovieLens100K with different latent factor *K*

(b) The performance of all methods on MovieLens1M with different latent factor *K*

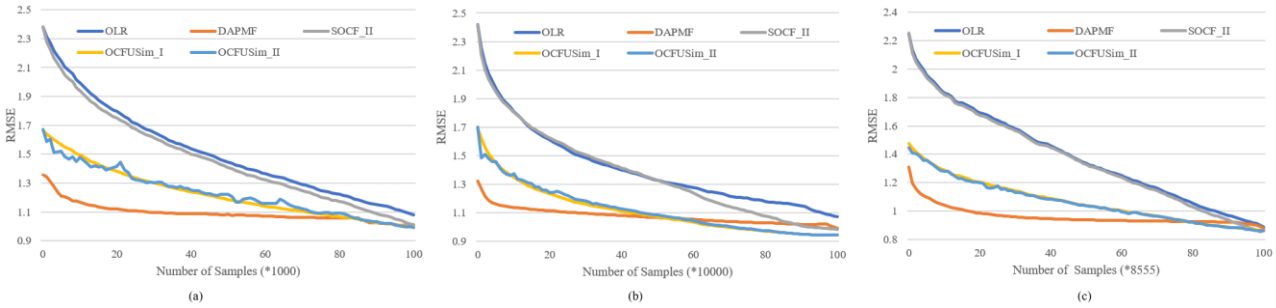

Figure 2: (a) The performance of all methods on Movielens100k and K=6;

(b) The performance of all methods on Movielens1M and K=6

(c) The performance of all methods on HetRec2011 and K=6

Fourthly, the Fig2 present the performance of all algorithm when the number of samples is changing and the latent K=6 on the MovieLens100k and MovieLens1M and HetRec2011. From the these pictures, we can find that the DAPMF convergence rate is fastest. Because the DAPMF is one of the method to speed up the OCF model. But it cannot achieve best performance finally. Conversely, our algorithms' convergence rate is faster than OLR and SOCF_II. Meanwhile, they gain the best accuracy than others. This is also proved that neighbor information is effective to OCF.

Finally, we conduct the experiment to show how the parameters λ_u, λ_v impact the performance of our algorithm. Fig.3(a) shows the OCFUSim_I performance when the number of samples is changing and latent features K = 7 on the MovieLens1M. Fig.3(b) presents the OCFUSim_II performance when the λ_u, λ_v are changing on the MovieLens1M. we found that the RMSE value of our algorithm will firstly fall and then rise.

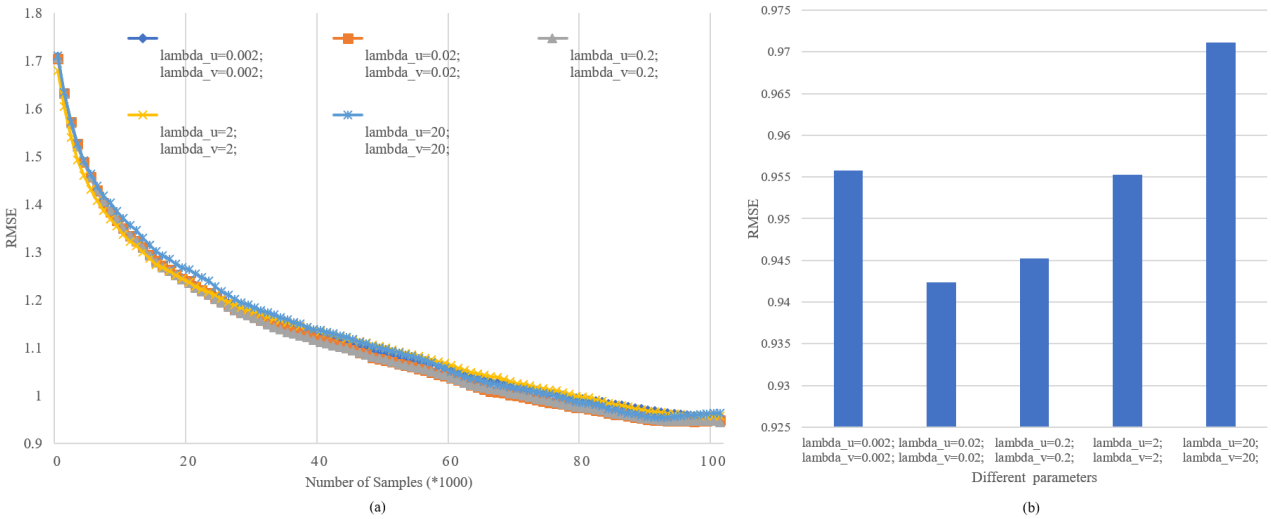

Figure 3: (a) The performance of OCFUSim_I on MovieLens1M and K=7;

(b) The performance of OCFUSim_II on MovieLens1M and K=7;

5. Conclusions and Future Work

In this paper, we incorporate the user neighbor information with the probability matrix factorization into the online collaborative filtering. In our first method, we find the user neighbor and add the average value of similarity to the objective function to predict the users' rating. In the second method, we add the similarity of two users as weight to the objective function to improve the accuracy of the model. Because the more similar people are, the more similar they have shopping preferences.

As is shown in our experiment result, we can find that the first method performs best when the threshold is 0.85 compared with other threshold. Also, when we compared our methods with other baseline methods, our methods usually outperform in most cases. From the perspective of converge rate, the DAPMF is the fastest method than other approaches, but its final RMSE value is not best among the algorithms. We can find our method converge rate is in the second place and have higher prediction accuracy than other algorithms. Thus, the network information is helpful for OCF. In addition, if the magnitude of data is the larger, our algorithms will have better performance. Because the more data we have, the more similar relationships we will find between users.

In future, we will try to incorporate the popularity of item to temporal OCF. Because users' preference and the popularity of item are change over time. Meanwhile, we will adopt the framework of dual-averaging online learning PMF to improve model converge rate.

6. References

[1] Aggarwal C C. Recommender Systems: The Textbook[M]. Springer Publishing Company, Incorporated, 2016.

[2] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. Computer 42, 8 (2009)

[3] Wang H , Zhang F , Wang J , et al. Ripple Network: Propagating User Preferences on the Knowledge Graph for Recommender Systems[J]. 2018

[4] Bahrehmand A , Naseri S , Ding C , et al. Enhancing tag-based collaborative filtering via integrated social networking information[C]// IEEE/ACM International Conference on Advances in Social Networks Analysis & Mining. IEEE, 2013.

[5] G. Ling, H. Yang, I. King, and M. R. Lyu, "Online Learning for Collaborative Filtering," presented at the IEEE World Congress on Computational Intelligence, Brisbane, Australia, 2012, pp. 1–8.

[6] Y.-X. Li, Z.-J. Li, F. Wang, and L. Kuang, "Accelerated Online Learning for Collaborative Filtering and Recommender Systems," 2014, pp. 879–885.

[7] F. Lin, X. Zhou, and W. Zeng, “Sparse Online Learning for Collaborative Filtering,” Int. J. Comput. Commun. CONTROL, vol. 11, no. 2, pp. 248–258, Apr. 2016.

[8] Zhou X , Shu W , Lin F , et al. Confidence-weighted bias model for online collaborative filtering[J]. Applied Soft Computing, 2017.

[9] Z. Zhao, H. Lu, D. Cai, X. He, and Y. Zhuang, “User Preference Learning for Online Social Recommendation,” IEEE Trans. Knowl. Data Eng., vol. 28, no. 9, pp. 2522–2534, Sep. 2016.

[10] J. Wang, S. C. H. Hoi, P. Zhao, and Z.-Y. Liu, “Online multi-task collaborative filtering for on-the-fly recommender systems,” 2013, pp. 237–244.

[11] J. Lu, S. Hoi, J. Wang, and P. Zhao, “Second order online collaborative filtering,” in Asian Conference on Machine Learning, 2013, pp. 325–340.

[12] Nathan N. Liu, Min Zhao, Evan Xiang, and Qiang Yang, “Online evolutionary collaborative filtering,” presented at the Proceeding RecSys ’10 Proceedings of the fourth ACM conference on Recommender systems Pages 95-102, Barcelona, Spain, 2010.

[13] X. Zhou and S. Wu, “Rating LDA model for collaborative filtering,” Knowl.-Based Syst., vol. 110, pp. 135–143, Oct. 2016.

[14] J. Wilson, S. Chaudhury, B. Lall, and P. Kapadia (2014), Improving Collaborative Filtering based Recommenders using Topic Modelling, Web Intelligence, 340–346.

[15] G. Rainer, N. Nrik, H. Peter J., and S. Yannis (2011), Large-scale matrix factorization with distributed stochastic gradient descent, Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, 69–77.

[16] Saric T; Simunovic G; Simunovic K (2013), Use Of Neural Networks In Prediction And Simulation Of Steel Surface Roughness, International Journal Of Simulation Modelling,12(4): 225-236.

[17] Jing L , Wang P , Yang L . Sparse Probabilistic Matrix Factorization by Laplace Distribution for Collaborative Filtering[C]// International Conference on Artificial Intelligence. AAAI Press, 2015.

[18] Josef Bauer and Alexandros Nanopoulos. A framework for matrix factorization based on general distributions. In Proc. of ACM RecSys, pages 249–256,2014.

[19] S. Kant and T. Mahara, “Merging user and item based collaborative filtering to alleviate data sparsity,” Int. J. Syst. Assur. Eng. Manag., vol. 9, no. 1, pp. 173–179, Feb. 2018.

[20] S. Shalev-Shwartz, “Online Learning and Online Convex Optimization,” Found. Trends® Mach. Learn., vol. 4, no. 2, pp. 107–194, 2011.

[21] C. Wang and D. M. Blei, “Collaborative topic modeling for recommending scientific articles,” in Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, 2011, pp. 448–456.

[22] X. Yang, H. Steck, and Y. Liu, “Circle-based recommendation in online social networks,” presented at the Proceeding KDD ’12 Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, Beijing, China, 2012, pp. 1267–1275.

[23] J. Abernethy, K. Canini, J. Langford, and A. Simma, “Online collaborative filtering,” Univ. Calif. Berkeley Tech Rep, 2007.

[24] Y. Koren, “Factor in the neighbors: Scalable and accurate collaborative filtering,” ACM Trans. Knowl. Discov. Data, vol. 4, no. 1, pp. 1–24, Jan. 2010.

[25] Bahrehmand a, Naseri S, Ding C, et al. Enhancing tag-based collaborative filtering via integrated social networking information[C]// Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. ACM, 2013. ACM, 2013.