Information Sciences 505 (2019) 535–548

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Sparse online collaborative filtering with dynamic regularization

Kangkang Li[a], Xiuze Zhou[b], Fan Lin[a,*], Wenhua Zeng[a], Beizhan Wang[a], Gil Alterovitz[c]

[a] *Software School, Xiamen University, Xiamen City, Fujian, China*
[b] *Department of Automation, Xiamen University, Xiamen City, Fujian, China*
[c] *Boston Children's Hospital, Harvard Medical School, Boston, USA*

ARTICLE INFO

Article history:
Received 17 November 2017
Revised 23 July 2019
Accepted 29 July 2019
Available online 29 July 2019

Keywords:
Collaborative filtering
Dynamic regularization
Online collaborative filtering
Neighborhood factor

ABSTRACT

Collaborative filtering (CF) approaches are widely applied in recommender systems. Traditional CF approaches have high costs to train the models and cannot capture changes in user interests and item popularity. Most CF approaches assume that user interests remain unchanged throughout the whole process. However, user preferences are always evolving and the popularity of items is always changing. Additionally, in a sparse matrix, the amount of known rating data is very small. In this paper, we propose a method of online collaborative filtering with dynamic regularization (OCF-DR), that considers dynamic information and uses the neighborhood factor to track the dynamic change in online collaborative filtering (OCF). The results from experiments on the MovieLens100K, MovieLens1M, and HetRec2011 datasets show that the proposed methods are significant improvements over several baseline approaches.

1. Introduction

Recommender systems help consumers find items of interest from a wide range of choices [3]. They provide advice to find products, alleviate information overload [2], and enhance our satisfaction and loyalty to increase sales for e-commerce firms [12].

Collaborative filtering (CF) is one of the most successful techniques applied in recommender systems [3]. CF approaches predict user preferences only on their historical rating data and do not require domain knowledge or additional information. The key idea of CF is that if two users have previously liked similar items, they will continue to do so in the future [34].

Many model-based CF approaches have been proposed to improve the quality of recommender systems. For example, Zhou et al. added rating information to the latent Dirichlet allocation (LDA) model to obtain the distribution of users' interests [39]. Palumbo et al. proposed entity2rec to learn user-item relatedness based on knowledge graphs for top-n recommender systems [26]. Liang et al. used matrix factorization and item embedding to obtain the cooccurrence patterns of rare items [17].

Many deep learning methods have been applied in recommender systems. For example, Wang et al. proposed a hierarchical Bayesian model by integrating a stacked denoising autoencoder into probabilistic matrix factorization [29]. He et al.

* Corresponding author.
E-mail address: iamafan@xmu.edu.cn (F. Lin).

https://doi.org/10.1016/j.ins.2019.07.093
0020-0255/

proposed a general framework, neural network-based CF, which leverages a multi-layer perceptron model to learn the user-item interaction function [8]. Wu et al. proposed recurrent recommender networks combining a long short-term memory model with traditional low-rank factorization [31]. Sedhain et al. used an autoencoder to embed items into latent space [27].

One of the most popular CF approaches is matrix factorization (MF), including probabilistic matrix factorization (PMF) [1] and nonnegative matrix factorization (NMF) [9]. The idea is to obtain user/item latent features from high-dimensional data and use them to predict unknown ratings.

Traditional CF methods that adopt batch learning algorithms [10] have four main drawbacks. First, retraining a model is expensive [4,16,18,19], and the models require all the data to be available before retraining. Second, they do not well handle dynamic ratings data, which are sequential, and new users and products are constantly being added [18]. Third, they fail to capture the drift of user preferences [4,13,37] because user interests are always evolving and the popularity of items is continuously changing. Finally, they are unsuitable and non-scalable for real-world large-scale online applications in which ratings usually arrive sequentially and periodically [23].

A variety of online collaborative filtering (OCF) methods have been proposed recently to alleviate these issues. Abernethy et al. proposed to learn a low-rank MF by optimizing the objective function in stochastic gradient descent [11]. Wang et al. exploited a principle similar to that of online multitask learning for OCF [30]. Lu et al. applied confidence-weighted learning [6] for OCF tasks [23]. Chenghao et al. proposed an online Bayesian inference algorithm incorporating content information into OCF [21].

OCF methods, which apply online learning algorithms, have several attractive advantages: (i) they avoid retraining a model from scratch for new training data because OCF methods update models sequentially [10]; (ii) their scale is linearly related to the number of observed ratings and the size of the latent features [19,38]; and (iii) they have low sensitivity to changes when models add new ratings [38] because they adopt online learning algorithms to update models for new training data.

The parameters of most current CF methods, both offline and online, are predetermined and invariant throughout the entire operation. These methods assume that user preferences and item popularity are static [23], which means that all items are equally rated by every user and all users are equally likely to rate every item. However, this assumption is not based in reality because user preferences continuously evolve [36].

CF methods must be able to track user interests and quickly provide recommendations [22]. Therefore, Koren proposed a CF method with temporal dynamics and built a factorization model of the changing characteristics of users and items [13]. Xiong et al. proposed a factor-based CF method that accounts for time [32]. Plovics et al. exploited temporal influences between users to improve recommendation quality [25].

In this paper, we add dynamic factors, including the dynamic average rating, user dynamic rating habits, item dynamic attributes, and dynamic rating distribution, to OCF to improve its accuracy. The neighborhood factor is also incorporated in our methods to track changes in user preferences. User rating behavior, item popularity, and the distribution of a user's ratings are constantly in flux, especially for online learning. If a user's tastes change, then his/her neighborhood will also change.

Our contributions in this paper include the following:

(i) We compute the dynamic rating behavior, i.e., the rating average of each user, the bias of each user, and the bias of each item in each interaction.
(ii) We update the weight of the user feature vector and item feature vector in each round.
(iii) We add the neighborhood factor to our method to track the drift of user preferences.

The rest of this paper is organized as follows. Section 2 briefly reviews the background and related work. Section 3 presents the details of our proposed model. Section 4 describes the experimental setting and results on three public datasets to demonstrate the performance of our methods. Section 5 provides our conclusions and suggestions for future work.

2. Background and related work

In this section, we introduce the problem settings of CF. Then we review some related work on recommender systems, including matrix-factorization-based CF methods, which perform well to reduce the problem of high-dimensional data, and OCF methods, which we study in this paper.

2.1. Problem setting

CF consists of three basic elements: user, item, and rating. The definitions of CF include the following:

Definition 1. User feature matrix $P \in \boldsymbol{R}^{k \times m}$; and user feature vectors P_u.

Definition 2. Item feature matrix $Q \in \boldsymbol{R}^{k \times n}$; and item feature vectors Q_i.

Definition 3. User-item rating matrix $R \in \boldsymbol{R}^{m \times n}$; $r_{u,i}$ represents the rating of user u on item i.

m is the number of users, n is the number of items, and k is the number of latent features. k is much smaller than n and m. P and Q are used to predict the unknown rating:

$$\hat{r}_{u,i} = P_u^T Q_i.$$

The approximation error is minimized to obtain the optimal matrices P and Q:

$$\underset{P \in \boldsymbol{R}^{k\times m}, Q \in \boldsymbol{R}^{k\times n}}{\operatorname{argmin}} \left\| R - P^T Q \right\|_F^2,$$

where $\|\cdot\|_F$ is the Frobenius norm of the matrix. The above equation can be formulated as

$$\underset{P \in \boldsymbol{R}^{k\times m}, Q \in \boldsymbol{R}^{k\times n}}{\operatorname{argmin}} \sum_{(u,i)\in C} l(P_u, Q_i, r_{u,i}),$$

where $C \subseteq \{(u,\ i) | r_{u,\ i}\, \textit{is known}\}$, and the l function is used to measure the difference between the predicted and real values. The most common measurements are root mean squared error (RMSE) and mean absolute error (MAE), defined as

$$RMSE = \sqrt{\frac{1}{|C|} \sum_{(u,i)\in C} (r_{u,i} - \hat{r}_{u,i})^2}$$

$$MAE = \frac{1}{|C|} \sum_{(u,i)\in C} \left| r_{u,i} - \hat{r}_{u,i} \right|,$$

where $C \subseteq \{(u,\ i) | r_{u,\ i}\, \textit{is known}\}$, $|C|$ is the number of known ratings in C, $\hat{r}_{u,i} = P_u^T Q_i$ is the predicted rating, and $r_{u,i}$ is the known rating. To optimize the RMSE, we define the loss function as

$$l_1(P_u, Q_i, r_{u,i}) = (r_{u,i} - P_u^T Q_i)^2.$$

Similarly, to optimize the MAE, we define the loss function as

$$l_2(P_u, Q_i, r_{u,i}) = \left| r_{u,i} - P_u^T Q_i \right|.$$

2.2. Matrix factorization-based CF methods

CF methods are generally divided into two basic categories of model- and memory-based CF [3]. Model-based CF methods perform better on problems of data sparsity, and are therefore more promising, and once trained, they make predictions more efficiently [19].

MF is one of the most successful model-based algorithms [13]. It performs well at reducing high-dimensional data. Its key idea is that a user's rating behavior on an item is determined by some latent features [24]. First, MF learns the user and item feature vectors based on user-item ratings to reduce the dimensionality, and then uses the feature vector to predict unknown ratings. Matrix-factorization-based CF is defined as

$$\sum_{(u,i)\in C} l(P_u, Q_i, r_{u,i}) + \frac{\lambda}{2} \left(\sum_{u=1}^{m} \|P_u\|^2 + \sum_{i=1}^{n} \|Q_i\|^2 \right),$$

where $C \subseteq \{(u,\ i) | r_{u,\ i}\, \textit{is known}\}$, $l(P_u,\ Q_i,\ r_{u,i})$ is a loss function, and λ is a regularization parameter.

2.3. OCF methods

Online learning algorithms can update the model and avoid the cost of retraining it when a new instance is added [10,28,35]. Many studies on OCF have emerged recently. Abernethy et al. proposed an OCF method for learning low-rank MF by optimizing the objective function directly [11]. However, this method optimized only the loss function and led to overfitting. An effective solution to this problem is to add regularization terms to constrain the objective function. Das et al. proposed a combination of MinHash clustering, probabilistic latent semantic indexing, and covisitation counts to recommend personalized news for online users [5]. Zhou et al. added bias and confidence weights to OCF to improve its stability and accuracy [38]. Ling et al. designed a dual-average method for probabilistic MF by adding previous rating information in an approximate average gradient of the loss [19].

OCF is applied in many other areas. For example, Chenghao et al. combined online adaptive passive-aggressive methods with nonnegative matrix factorization to improve the accuracy [20], and Yang et al. applied OCF to social networking [33].

3. Online collaborative filtering with dynamic regularization

We now introduce the algorithm for online collaborative filtering with dynamic regularization (OCF-DR). Its key function is to add dynamic regularization and a neighborhood factor to OCF to improve its stability and accuracy.

3.1. Online collaborative filtering with dynamic regularization

The parameters of most CF methods are predetermined and invariable, and user preferences and item popularities are assumed to be static [36]. Therefore, the methods assume that all items are equally rated by every user and all users are equally likely to rate every item. This assumption is not realistic [38].

First, the popularity of products constantly changes, especially when new choices emerge [18]. Seasonal factors are among the most important. For example, sweaters are easier to sell in winter than in summer. Second, a user's inclination continuously changes over time, which leads to the repositioning of their tastes. Emotions are another important factor [39]. For example, users who rated a movie at 3 stars may change their rating to 4 stars depending on their mood. So, dynamic factors are essential for OCF methods [18]. However, current work on OCF seldom considers the dynamic changes of users and items, and to consider the rating of only the interaction of the user feature vector and item feature vector is unreasonable.

Therefore, we propose OCF-DR, which adds dynamic regularization and a neighborhood factor to OCF. It has three main parts: (i) computing the dynamic change, i.e., the dynamic rating average of each user, the dynamic bias of each user, and the dynamic bias of each item in each interaction; (ii) updating the weights of the user feature vector and item feature vector in each round; and (iii) taking the neighborhood factor into account to track the change of user preferences.

Our method predicts ratings as follows:

$$\hat{r}_{u,i} = \mu + b_u + b_i + P_u^T Q_i,$$

where μ is the overall rating average; b_u is the user bias, the observed deviation of user u, i.e., the rating behavior of the user; and b_i is the item bias, the observed deviation of item i, i.e., the unique characteristics of the item.

User bias represents the intrinsic trend of the user. For example, some users tend to give higher ratings than others. Item bias represents the intrinsic properties of the item. For example, some items receive higher ratings than others. Both biases are independent of user interactions.

To optimize the RMSE, we define the loss function as

$$l_1(P_u, Q_i, b_u, b_i, r_{u,i}) = (r_{u,i} - \mu - b_u - b_i - P_u^T Q_i)^2. \tag{1}$$

Similarly, to optimize the MAE, we define the loss function as

$$l_2(P_u, Q_i, b_u, b_i, r_{u,i}) = \left| r_{u,i} - \mu - b_u - b_i - P_u^T Q_i \right|. \tag{2}$$

In our methods, we choose l_1 as the evaluation metric.

3.1.1. OCF-DR_I

The objective function of OCF-DR_I is

$$C(P_u, Q_i, b_u, b_i) = \sum_{(u,i)\in C} l(P_u, Q_i, b_u, b_i, r_{u,i}) + \frac{\lambda}{2}\left(\sum_{u=1}^{m} p(u)\|P_u\|^2 + \sum_{i=1}^{n} q(i)\|Q_i\|^2 + b_u^2 + b_i^2\right), \tag{3}$$

where $p(u)$ is the probability-of-observing row of user u, i.e., the number of items rated by the user; $q(i)$ is the probability-of-observing column of item i, i.e., the number of users who have rated item i; and λ is a regularization parameter.

The probability distribution of user rating behavior is not uniform because some items are more popular and more likely to receive ratings, and some users are more likely to provide ratings. Therefore, the penalization of the feature vectors of users or items with more ratings is increased.

Stochastic gradient descent (SGD) is an effective tool for optimization and online learning [10]. We adopt SGD to optimize formula (3) and obtain the update rules P_u, Q_i, b_u, and b_i:

$$P_u = P_u - \eta \frac{\partial C}{\partial P_u} = (1 - \lambda\eta \cdot p(u))P_u + 2\eta e_{u,i} Q_i \tag{4}$$

$$Q_i = Q_i - \eta \frac{\partial C}{\partial Q_i} = (1 - \lambda\eta \cdot q(i))Q_i + 2\eta e_{u,i} P_u \tag{5}$$

$$b_u = b_u - \eta \frac{\partial C}{\partial b_u} = (1 - \lambda\eta)b_u + 2\eta e_{u,i} \tag{6}$$

$$b_i = b_i - \eta \frac{\partial C}{\partial b_i} = (1 - \lambda\eta)b_i + 2\eta e_{u,i}, \tag{7}$$

where $e_{u,i} = r_{u,i} - \hat{r}_{u,i}$. $\hat{r}_{u,i}$ is the predicted rating, and $r_{u,i}$ is the known rating. λ is a regularization parameter. η denotes the learning rate parameter, which controls the change in each step (Algorithm 1).

Algorithm 1 Online collaborative filtering with dynamic regularization (OCF-DR_I).

Parameters: n, m, k, λ, η
Input:a sequence of rating pairs $(u, i, r_{u,i})$
Initialization: initialize a random matrix for user feature matrix $P \in \boldsymbol{R}^{k\times m}$ and item feature matrix $Q \in \boldsymbol{R}^{k\times n}$; initialize a random matrix for $b_u \in \boldsymbol{R}^{1\times m}$ and $b_i \in \boldsymbol{R}^{1\times n}$; initialize zero matrices for $T_u \in \boldsymbol{R}^{1\times m}$, the number of items rated by user u, and $T_i \in \boldsymbol{R}^{1\times n}$, the number of users who have rated item i
For $t = 1, 2, \ldots, T$ **do**

1: Receive rating prediction request of user u on item i
2: Compute μ and make prediction $\hat{r}_{u,i} = \mu + b_u + b_i + P_u^T Q_i$
3: The true rating $r_{u,i}$ is revealed
4: The algorithm suffers a loss $l(P_u, Q_i, b_u, b_i, r_{u,i})$
5: Update T_u and T_i: $T_u = T_u + 1$; $T_i = T_i + 1$
6: Update $p(u)$ and $q(i)$: $p(u) = T_u/t, q(i) = T_i/t$
7: Update P_u, Q_i, b_u, and b_i according to (4), (5), (6), and (7), respectively

End for

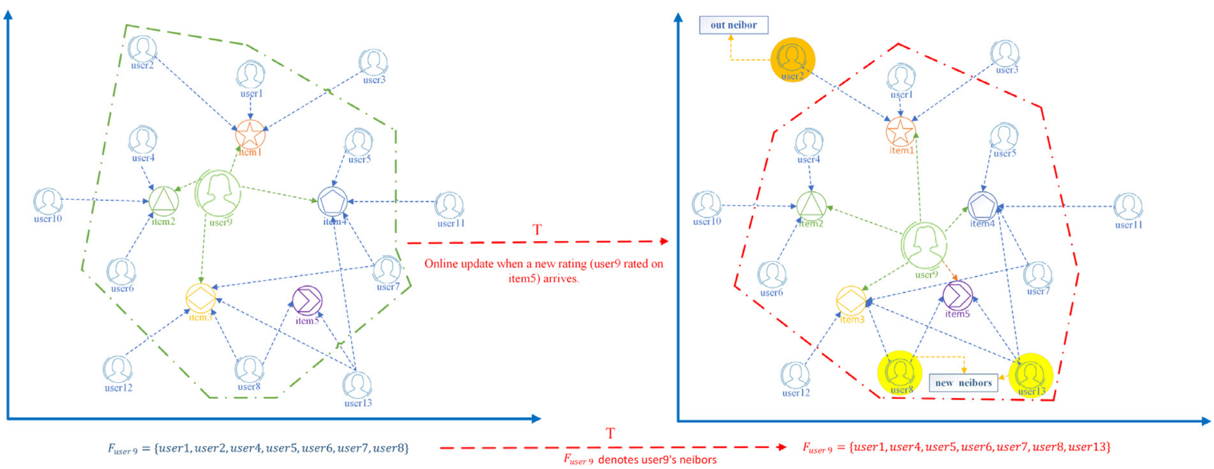

Fig. 1. Diagram of dynamic neighborhoods.

3.1.2. OCF-DR_II

Neighboring temporal information is an important factor in CF that improves the recommendation accuracy [14]. Lathia et al. incorporated the adaptive neighborhood into temporal CF [15], but they considered only the varying size of the neighborhood over time. Liu et al. added temporal information and developed incremental learning similarity to extend the neighborhood-based algorithm [22].

For every interaction, a user feature vector will change after a user rates an item. To prevent overfitting in this dynamic process, we incorporate a regularization term related to the difference between a users feature vectors and his/her neighborhoods feature vectors for a period of time. By online updating, the farthest neighbors of the user may change for a period of time. A diagram of dynamic neighborhoods is shown in Fig. 1.

So, we attempt to take the neighboring temporal information into OCF-DR_I. Then the objective function of OCF-DR_II becomes

$$C(P_u, Q_i, b_u, b_i) = \sum_{(u,i)\in C} l(P_u, Q_i, b_u, b_i, r_{u,i}) + \frac{\lambda}{2}\left(\sum_{u=1}^{m} p(u)\|P_u\|^2 + \sum_{i=1}^{n} q(i)\|Q_i\|^2 + b_u^2 + b_i^2\right) + \frac{\alpha}{2}\left(\sum_{u=i}^{m}\sum_{f\in F_u} sim(u,f)\left\|P_u - P_f\right\|^2\right), \quad (8)$$

where F_u is the set of neighbors of user u; $|F_u|$ is the number of neighbors of user u; α and λ are regularization parameters; $p(u)$ denotes the probability-of-observing row of user u; $q(i)$ is the probability-of-observing column of item i; and $sim(u, f)$ is the similarity between user u and f, defined as

$$sim(u,f) = \frac{\left|I_u \cap I_f\right|}{\left|I_u \cup I_f\right|}, \quad (9)$$

where I_u and I_f are the respective sets that users u and f have rated.

We can similarly obtain the OCF-DR_II update rules:

$$P_u = P_u - \eta \frac{\partial C}{\partial P_u} = (1 - \alpha\eta \sum_{f \in F_u} sim(u,f) - \lambda\eta \cdot p(u))P_u + \alpha\eta \sum_{f \in F_u} sim(u,f)P_f + 2\eta e_{u,i} Q_i \tag{10}$$

$$Q_i = Q_i - \eta \frac{\partial C}{\partial Q_i} = (1 - \lambda\eta \cdot q(i))Q_i + 2\eta e_{u,i} P_u \tag{11}$$

$$b_u = b_u - \eta \frac{\partial C}{\partial b_u} = (1 - \lambda\eta)b_u + 2\eta e_{u,i} \tag{12}$$

$$b_i = b_i - \eta \frac{\partial C}{\partial b_i} = (1 - \lambda\eta)b_i + 2\eta e_{u,i}, \tag{13}$$

where $\hat{r}_{u,i}$ is the predicted rating, $r_{u,i}$ is the known rating, $e_{u,i} = r_{u,i} - \hat{r}_{u,i}$, λ is a regularization parameter, and η is the learning rate parameter, which controls the change in each step.

To improve the algorithm's speed, we regularly update the neighborhood of a user. We define the update cycle of the user neighborhood C and the number of neighborhoods H (Algorithm 2).

Algorithm 2 OCF-DR_II.

Parameters: $n, m, k, \lambda, \alpha, \eta, C, H$
Input:a sequence of rating pairs $(u, i, r_{u,i})$
Initialization: initialize a random matrix for user feature matrix $P \in \boldsymbol{R}^{k\times m}$ and item feature matrix $Q \in \boldsymbol{R}^{k\times n}$; initialize a random matrix for $b_u \in \boldsymbol{R}^{1\times m}$ and $b_i \in \boldsymbol{R}^{1\times n}$; initialize zero matrices for $T_u \in \boldsymbol{R}^{1\times m}$, the number of items rated by user u, and $T_i \in \boldsymbol{R}^{1\times n}$, the number of users who have rated item i; initialize a random matrix for the neighborhood matrix $F \in \boldsymbol{R}^{m\times H}$
For $t = 1, 2, \ldots, T$ **do**

```
Receive rating prediction request of user u on item i
Compute μ and make prediction r̂_{u,i} = μ + b_u + b_i + P_u^T Q_i
The true rating r_{u,i} is revealed
The algorithm suffers a loss l(P_u, Q_i, b_u, b_i, r_{u,i})
Update T_u and T_i: T_u = T_u + 1; T_i = T_i + 1
Update p(u) and q(i): p(u) = T_u/t, q(i) = T_i/t
Update P_u, Q_i, b_u, and b_i according to (10), (11), (12), and (13), respectively
if t = C then
    Update F according to (9)
end if
```

End for

4. Experiments

We performed several experiments to compare the quality of our proposed methods to those of the baseline approaches, and analyzed the results in detail.

4.1. Datasets

We performed experiments on three public datasets: MovieLens100K [7], MovieLens1M [7], and HetRec2011 [7], which are available from the MovieLens website.[1] The MovieLens100K dataset contains 100,000 ratings from 943 users on 1682 movies. The MovieLens1M dataset consists of 1 million ratings from 6000 users on 4000 movies. The HetRec2011 dataset comprises 855,598 ratings from 2113 users on 10,109 movies.

We adopted the most popular metric, RMSE, to evaluate the prediction accuracy of our proposed methods. A smaller RMSE indicates better prediction accuracy.

4.2. Baseline approaches

To demonstrate the advantages and disadvantages of the algorithm, we conducted a comparative experiment with our three baseline approaches.

[1] https://grouplens.org/datasets/movielens/.

Table 1
Performance on MovieLens100K.

Method\\k	5	10	15	20	25	30	35	40
OLR	1.1242	0.9911	**1.0142**	1.1133	1.236	1.3603	1.4833	1.5949
CWOCF_I	1.0405	1.0101	1.0184	1.0536	1.0931	1.1541	1.2340	1.3473
SOCF_II	1.0403	**0.9863**	1.0434	1.1485	1.2432	1.3286	1.3985	1.4861
OCF-DR_I	1.0384	1.0390	1.0401	1.0398	1.0391	1.0393	1.0398	1.0405
OCF-DR_II	**1.0381**	1.0388	1.0384	**1.0385**	**1.0389**	**1.0391**	**1.0387**	**1.0393**

Table 2
Performance on MovieLens1M.

Method\\k	5	10	15	20	25	30	35	40
OLR	1.1166	0.9760	0.9695	1.0236	1.0976	1.1768	1.2500	1.3204
CWOCF_I	0.9732	0.9642	0.9693	0.9851	1.0139	1.0521	1.0806	1.1200
SOCF_II	1.0054	**0.9469**	0.9699	1.0211	1.0874	1.1544	1.2173	1.2765
OCF-DR_I	0.9707	0.9701	0.9691	0.9697	0.9700	0.9705	0.9719	0.9726
OCF-DR_II	**0.9688**	0.9689	**0.9685**	**0.9691**	**0.9683**	**0.9694**	**0.9698**	**0.9710**

Table 3
Performance on HetRec2011.

Method\\k	5	10	15	20	25	30	35	40
OLR	0.9075	0.8869	0.896	0.9099	0.9654	1.0244	1.0862	1.1473
CWOCF_I	**0.8754**	**0.8541**	0.8906	0.8959	0.9076	0.9156	0.9222	0.9392
SOCF_II	0.9131	0.8630	0.8982	0.9147	0.9209	0.9367	0.9419	0.9577
OCF-DR_I	0.8921	0.8906	0.8912	0.8917	0.8920	0.8939	0.8941	0.8957
OCF-DR_II	0.8908	0.8900	**0.8898**	**0.8893**	**0.8888**	**0.8884**	**0.8879**	**0.8870**

- **OLR**: online low-rank approximation, which learns a rank-k MF by using online gradient descent to directly optimize the loss function [11];
- **CWOCF_I**: confidence-weighted OCF, which applies confidence-weighted online learning to address the OCF task [23];
- **SOCF_II**: second-order sparse OCF, which adds an absolute term to the objective function [18].

4.3. Results and analysis

4.3.1. Overall performance

We evaluated the performance of all the methods on the three datasets. The learning rate η of all the methods was set to 0.005 for a fair comparison. The number of latent features varied from 5 to 40. The parameters of these baseline approaches were set according to previous work or by searching from a single experiment on each dataset. Each experiment was run randomly ten times, and we obtained the average values of each method.

Tables 1–3 present the performance of the methods on each dataset. The bold elements in the tables represent the best performance among all methods. The tables show that our proposed methods performed best in most cases. We take Table 1 as an example. As the number of latent features k increases, the RMSE increases slowly in OCF-DR_I. Under the same value of k, the RMSE values in OCF-DR_II are slightly better than those of OCF-DR_I. Because the rating matrix is sparse, the neighbor weight method improves the prediction accuracy of OCF. Compared with those of the baselines, the RMSE values in our methods are stable with increasing k because once our model has been fitted, users with very few ratings will have feature vectors close to the mean, so the predicted ratings for those users will be close to the average movie ratings.

The RMSE values of all approaches on the three datasets with the number of latent features ranging from 5 to 40 are shown in Figs. 2–6. Our methods outperformed the other OCF methods, and converged faster than all the baseline approaches, especially when the algorithm started running.

First, compared with other baseline approaches, i.e., OLR, CWOCF_I, and SOCF_II, we found that our methods performed better in terms of RMSE. In most cases, our method resulted in smaller RMSE values, indicating that methods that consider dynamic regularization are effective for online learning CF.

Second, we observed that our methods were more stable than the baseline approaches in most cases. OLR was relatively sensitive because it loses some temporal information in online learning. Temporal dynamic factors are important for OCF to capture the users dynamic habits and interests throughout the online learning process.

Third, our methods were able to obtain relatively stable RMSE values under different conditions with varying numbers of latent features. Therefore, our methods are robust to the number of latent features.

Finally, when we compared OCF-DR_I and OCF-DR_II, we found that OCF-DR_II achieved lower RMSE values in most cases, indicating that the neighborhood factor is essential for OCF tasks to improve the prediction accuracy.

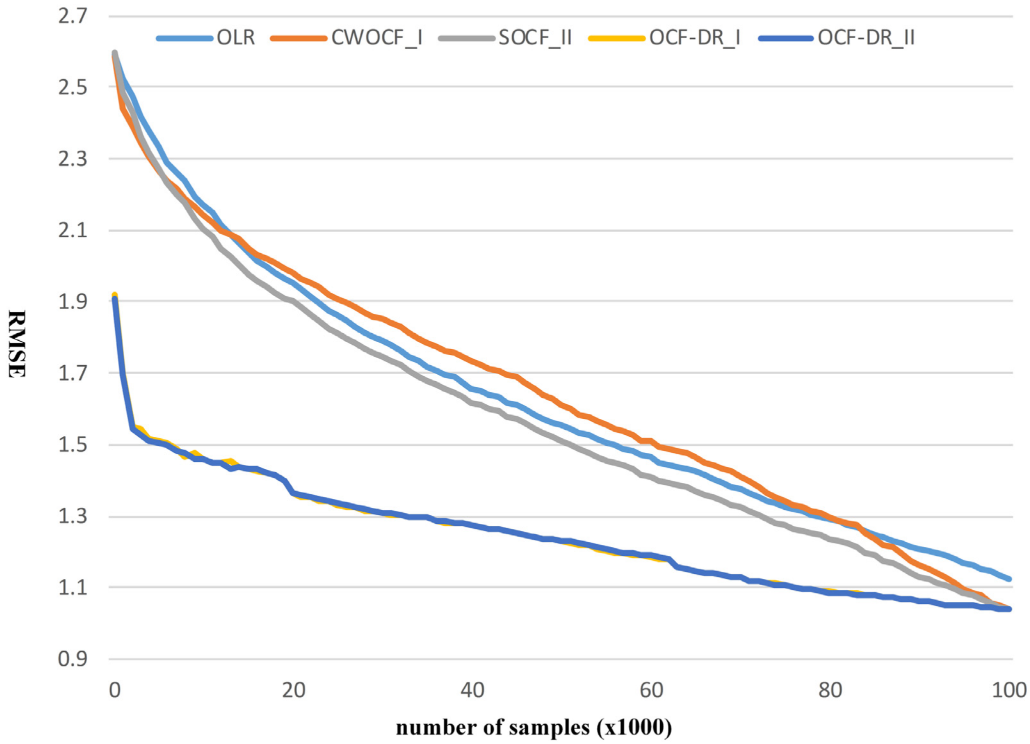

Fig. 2. Performance of all methods on MovieLens100K with $k = 5$.

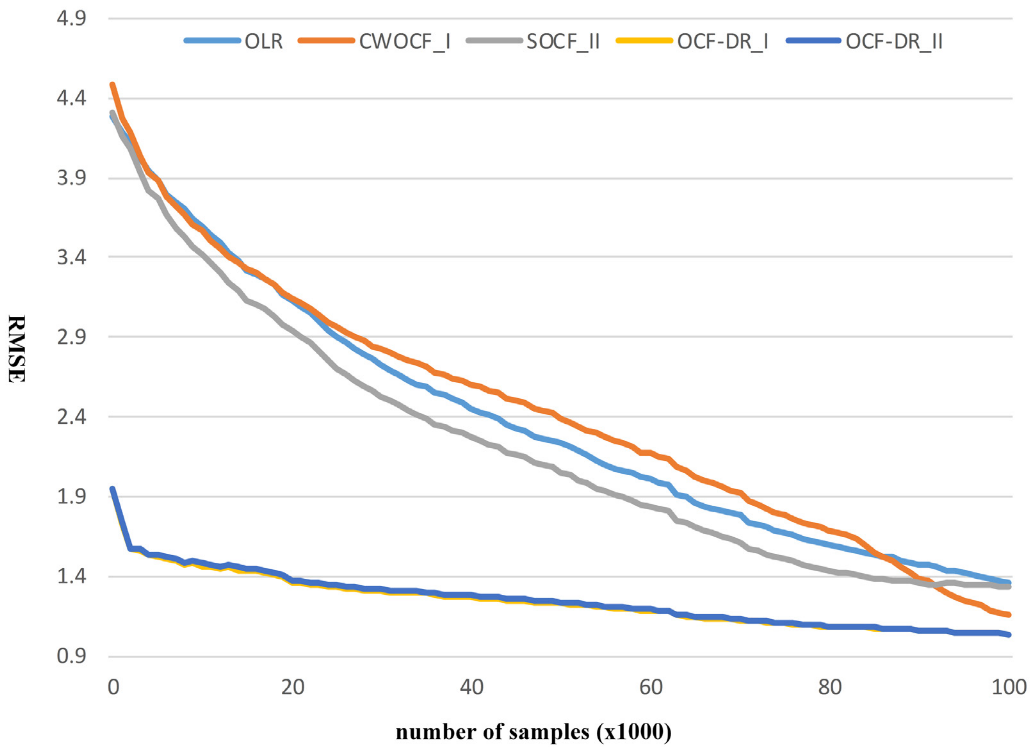

Fig. 3. Performance of all methods on MovieLens100K with $k = 30$.

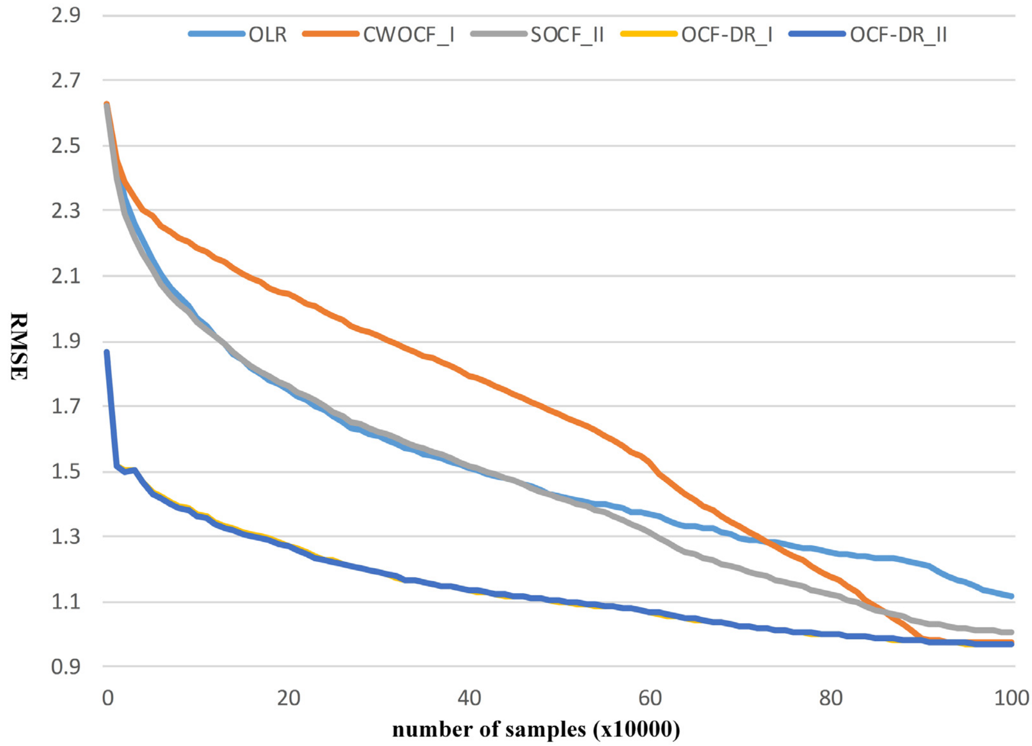

Fig. 4. Performance of all methods on MovieLens1M with $k = 5$.

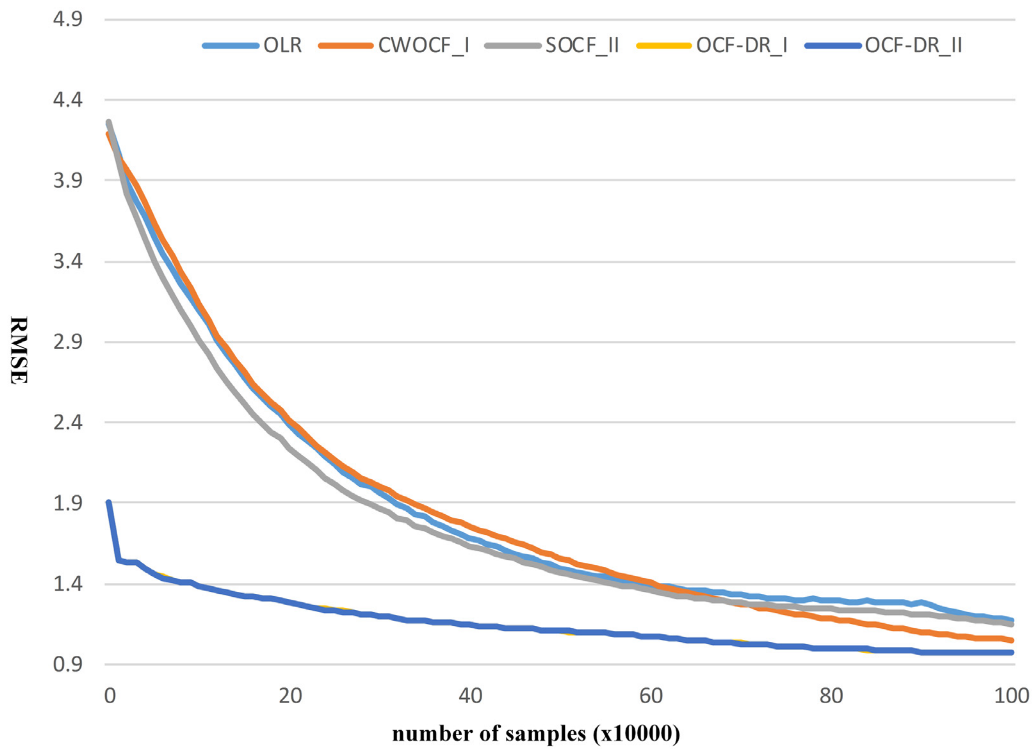

Fig. 5. Performance of all methods on MovieLens1M with $k = 30$.

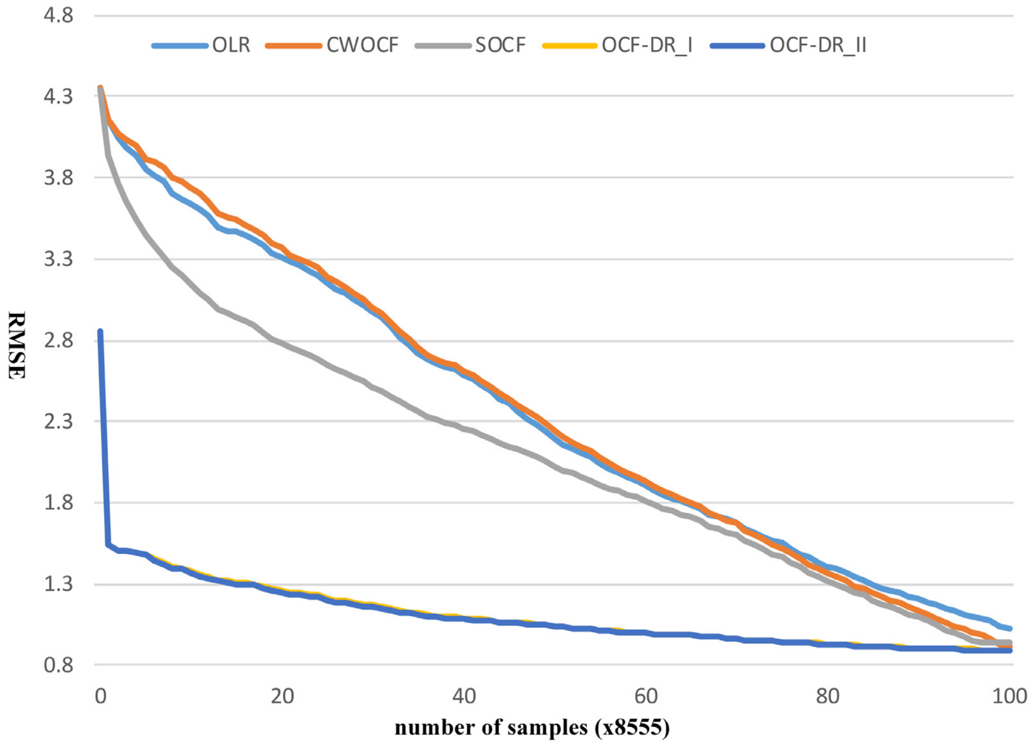

Fig. 6. Performance of all methods on HetRec2011 with $k = 30$.

Table 4
Results of OCF-DR_II with different α on MovieLens1M when $k = 25$.

$\alpha \backslash \lambda$	0.01	0.1	1	10
0.01	0.9870	0.9818	0.9695	1.0142
0.1	0.9866	0.9822	0.9683	1.0155
1	0.9854	0.9807	0.9730	1.0354
10	0.9849	0.9806	0.9745	1.0522

4.3.2. Impact of the number of latent features

Figs. 7 and 8 present the impact of the number of latent features on all methods. Fig. 9 shows an enlarged drawing of our methods with different numbers of latent features k on MovieLens1M. We observe that our methods were stable under different numbers of latent features, while OLR, CWOCF_I, and SOCF_II were sensitive to changes in the number of latent features. On one hand, too small a value of k (less than 15) causes these approaches to underfit. However, they all perform well as the value of k increases. On the other hand, because a large value of k causes these approaches to overfit, when the value of k is large (larger than 15) and continues to increase, the performance of all the baseline approaches worsens. Our methods are stable, and k has little effect on our algorithms.

When the value of k is small ($k < 15$), the latent features are not sufficient to contain the user's interest information, and our algorithms perform no better than other methods. This is because CWOCF_I and SOCF_II are second-order methods which use the covariance information of the latent vector to compensate for the lack of user interest information. When the value of k is large ($k > 15$), the latent features contain sufficient user interest information. Covariance information of the latent vector is redundant, which results in no obvious increase in precision. However, our methods are stable because we track dynamic changes: the average of each user, the bias of each user, and the bias of each item in each round. Moreover, Fig. 9 shows that RMSE values of our methods fluctuate in a very small range as k increases. This suggests that a large latent factor has little effect on our methods.

4.3.3. The impact of parameters

Fig. 10 and Table 4 present the impact of two parameters on our methods. First, we set $k = 25$ for OCF-DR_I to show its performance under different values of λ, which controls the impact of the regularization penalty for OCF-DR_I. We observe that the curve of OCF-DR_I declines more sharply and finally reaches a larger RMSE value with an increasing number of samples. Second, we set $k = 25$ for OCF-DR_II to show its performance under different values of α and λ. From Table 4, we

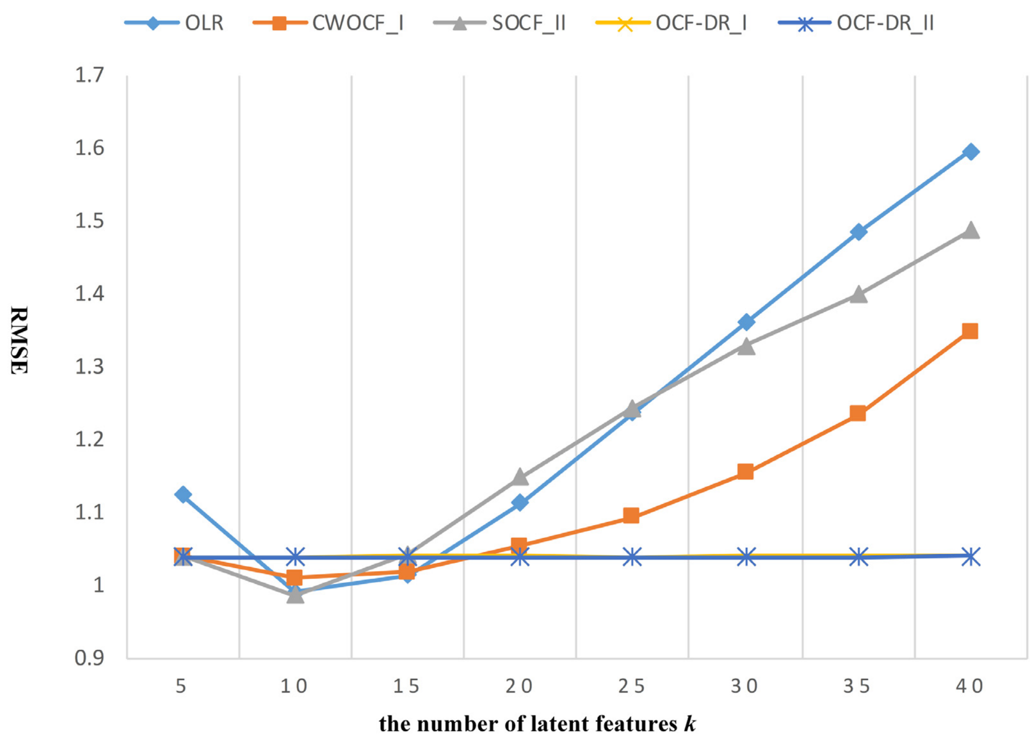

Fig. 7. Performance of all methods on MovieLens100K with different numbers of latent features *k*.

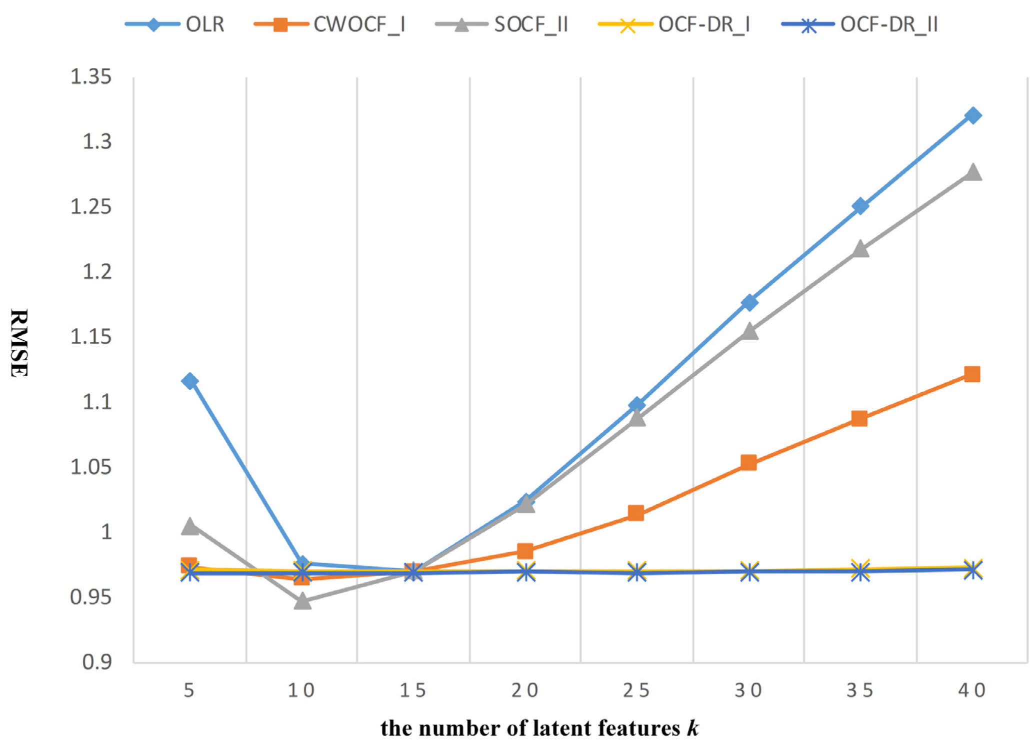

Fig. 8. Performance of all methods on MovieLens1M with different number of latent features *k*.

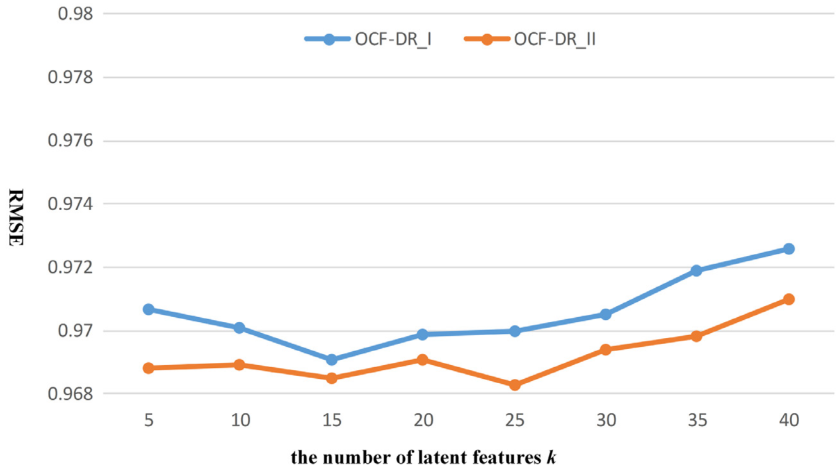

Fig. 9. Enlarged drawing of OCF-DR_I and OCF-DR_II on MovieLens1M with different numbers of latent features k.

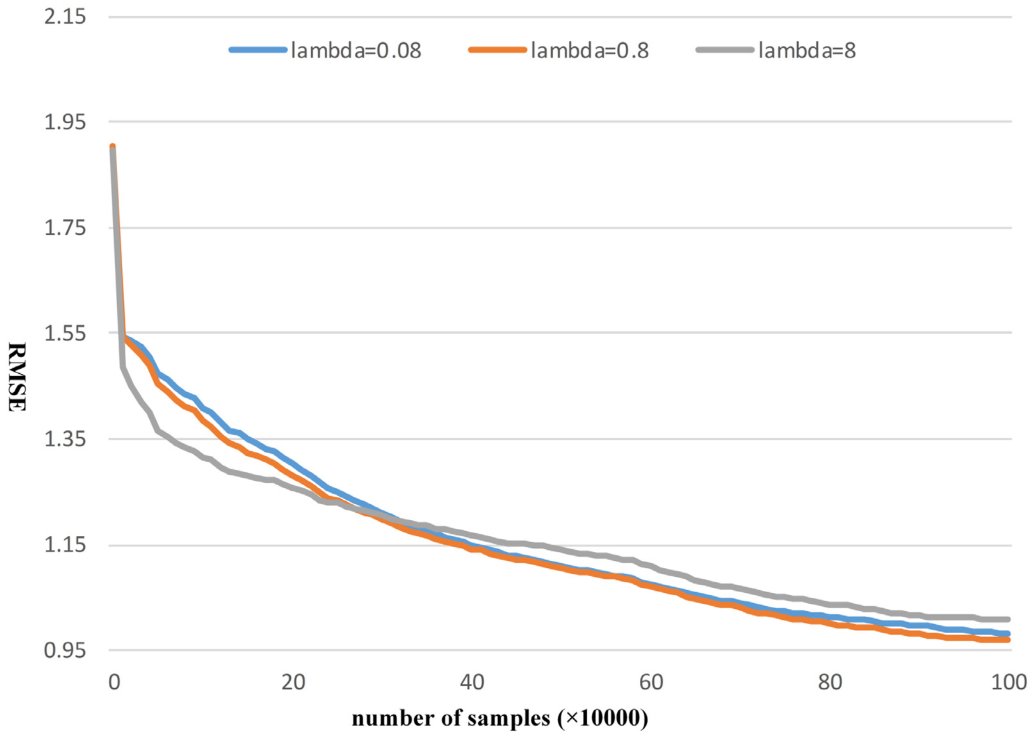

Fig. 10. Convergence effects of OCF-DR_I with different λ on MovieLens1M when $k = 25$.

observe that for a given parameter, the RMSE value decreases first and then increases as another parameter increases because the algorithm is underfit for extremely small λ or α and overfit for extremely large λ or α. Therefore, both parameters are essential to constrain the stability and convergence rate of the algorithms.

5. Conclusions

In this paper, we considered the dynamic rating behavior and updated the weight of the user feature vector and item feature vector in each round to improve prediction accuracy. We then added the neighborhood factor to our method to track the drift of user preferences. Experimental results show that our proposed methods outperformed the baseline approaches. Compared to the baseline approaches, our methods converge rapidly and obtain high prediction accuracy, i.e., lower RMSE values. Therefore, to consider rational dynamic factors is crucial to improve the prediction accuracy of OCF methods. In the future, we will focus on two directions. First, we will try to extract user and item features from user/item side information by deep learning approaches offline as input for OCF. Second, we will attempt to apply our methods to education recommender systems.

Declaration of Competing Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

References

[1] Y. Cao, W. Li, D. Zheng, An improved neighborhood-aware unified probabilistic matrix factorization recommendation, Wirel. Pers. Commun. 102 (4) (2018) 3121–3140.
[2] Z.D. Champiri, S.R. Shahamiri, S.S.B. Salim, A systematic review of scholar context-aware recommender systems, Expert Syst. Appl. 42 (3) (2015) 1743–1758.
[3] R. Chen, Q. Hua, Y. Chang, B. Wang, L. Zhang, X. Kong, A survey of collaborative filtering-based recommender systems: from traditional methods to hybrid methods based on social networks, IEEE Access 6 (2018) 64301–64320.
[4] K. Christakopoulou, F. Radlinski, K. Hofmann, Towards conversational recommender systems, in: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '16, ACM, New York, NY, USA, 2016, pp. 815–824.
[5] A.S. Das, M. Datar, A. Garg, S. Rajaram, Google news personalization: scalable online collaborative filtering, in: Proceedings of the 16th International Conference on World Wide Web, in: WWW '07, ACM, New York, NY, USA, 2007, pp. 271–280.
[6] M. Dredze, K. Crammer, F. Pereira, Confidence-weighted linear classification, in: Proceedings of the 25th International Conference on Machine Learning, in: ICML '08, ACM, New York, NY, USA, 2008, pp. 264–271.
[7] F.M. Harper, J.A. Konstan, The movielens datasets: history and context, ACM Trans. Interact. Intell. Syst. 5 (4) (2015) 19:1–19:19.
[8] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proceedings of the 26th International Conference on World Wide Web, in: WWW '17, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 2017, pp. 173–182.
[9] A. Hernando, J. Bobadilla, F. Ortega, A non negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model, Knowl.-Based Syst. 97 (C) (2016) 188–202.
[10] S.C.H. Hoi, D. Sahoo, J. Lu, P. Zhao, Online learning: a comprehensive survey, CoRR, 2018 abs/1802.02871.
[11] J. Langford, J. Abernethy, K. Canini, A. Simma, Online Collaborative Filtering, Tech. Rep., University of California at Berkeley, 2007.
[12] M. Jugovac, D. Jannach, Interacting with recommenders overview and research directions, ACM Trans. Interact. Intell. Syst. 7 (3) (2017) 10:1–10:46.
[13] Y. Koren, Collaborative filtering with temporal dynamics, Commun. ACM 53 (4) (2010) 89–97.
[14] Y. Koren, Factor in the neighbors: scalable and accurate collaborative filtering, ACM Trans. Knowl. Discov. Data 4 (1) (2010) 1:1–1:24.
[15] N. Lathia, S. Hailes, L. Capra, Temporal collaborative filtering with adaptive neighbourhoods, in: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, in: SIGIR '09, ACM, New York, NY, USA, 2009, pp. 796–797.
[16] Y. Li, Z. Li, F. Wang, L. Kuang, Accelerated online learning for collaborative filtering and recommender systems, in: 2014 IEEE International Conference on Data Mining Workshop, 2014, pp. 879–885.
[17] D. Liang, J. Altosaar, L. Charlin, D.M. Blei, Factorization meets the item embedding: regularizing matrix factorization with item co-occurrence, in: Proceedings of the 10th ACM Conference on Recommender Systems, in: RecSys '16, ACM, New York, NY, USA, 2016, pp. 59–66.
[18] F. Lin, X. Zhou, W.H. Zeng, F. Lin, X. Zhou, W. Zeng, Sparse online learning for collaborative filtering, Int. J. Comput. Commun. Control 11 (2) (2016) 248–258. ISSN
[19] G. Ling, H. Yang, I. King, M.R. Lyu, Online learning for collaborative filtering, in: The 2012 International Joint Conference on Neural Networks (IJCNN), 2012, pp. 1–8.
[20] C. Liu, S.C. Hoi, P. Zhao, J. Sun, E.-P. Lim, Online adaptive passive-aggressive methods for non-negative matrix factorization and its applications, in: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, in: CIKM '16, ACM, New York, NY, USA, 2016, pp. 1161–1170.
[21] C. Liu, T. Jin, S.C. Hoi, P. Zhao, J. Sun, Collaborative topic regression for online recommender systems: an online and Bayesian approach, Mach. Learn. 106 (5) (2017) 651–670.
[22] N.N. Liu, M. Zhao, E. Xiang, Q. Yang, Online evolutionary collaborative filtering, in: Proceedings of the Fourth ACM Conference on Recommender Systems, in: RecSys '10, ACM, New York, NY, USA, 2010, pp. 95–102.
[23] J. Lu, S.C.H. Hoi, J. Wang, P. Zhao, Second order online collaborative filtering, in: JMLR: Workshop and Conference Proceedings: ACML 2013, 14, 2013, pp. 325–340.
[24] X. Luo, M. Zhou, Y. Xia, Q. Zhu, An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems, IEEE Trans. Ind. Inform. 10 (2) (2014) 1273–1284.
[25] R. Pálovics, A.A. Benczúr, L. Kocsis, T. Kiss, E. Frigó, Exploiting temporal influence in online recommendation, in: Proceedings of the 8th ACM Conference on Recommender Systems, in: RecSys '14, ACM, New York, NY, USA, 2014, pp. 273–280.
[26] E. Palumbo, G. Rizzo, R. Troncy, Entity2rec: learning user-item relatedness from knowledge graphs for top-n item recommendation, in: Proceedings of the Eleventh ACM Conference on Recommender Systems, in: RecSys '17, ACM, New York, NY, USA, 2017, pp. 32–36.
[27] S. Sedhain, A.K. Menon, S. Sanner, L. Xie, Autorec: autoencoders meet collaborative filtering, in: Proceedings of the 24th International Conference on World Wide Web, in: WWW '15 Companion, ACM, New York, NY, USA, 2015, pp. 111–112.
[28] S. Shalev-Shwartz, Online learning and online convex optimization, Found. Trends Mach. Learn. 4 (2) (2012) 107–194.
[29] H. Wang, N. Wang, D.-Y. Yeung, Collaborative deep learning for recommender systems, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '15, ACM, New York, NY, USA, 2015, pp. 1235–1244.
[30] J. Wang, S.C. Hoi, P. Zhao, Z.-Y. Liu, Online multi-task collaborative filtering for on-the-fly recommender systems, in: Proceedings of the 7th ACM Conference on Recommender Systems, in: RecSys '13, ACM, New York, NY, USA, 2013, pp. 237–244.
[31] C.-Y. Wu, A. Ahmed, A. Beutel, A.J. Smola, H. Jing, Recurrent recommender networks, in: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, in: WSDM '17, ACM, New York, NY, USA, 2017, pp. 495–503.
[32] L. Xiong, X. Chen, T.-K. Huang, J.G. Schneider, J.G. Carbonell, Temporal collaborative filtering with Bayesian probabilistic tensor factorization., in: SDM, SIAM, 2010, pp. 211–222.
[33] X. Yang, H. Steck, Y. Liu, Circle-based recommendation in online social networks, in: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '12, ACM, New York, NY, USA, 2012, pp. 1267–1275.
[34] Z. Yang, B. Wu, K. Zheng, X. Wang, L. Lei, A survey of collaborative filtering-based recommender systems for mobile internet applications, IEEE Access 4 (2016) 3273–3287.
[35] P. Zhao, S.C.H. Hoi, R. Jin, DUOL: a double updating approach for online learning, in: Proceedings of the 22nd International Conference on Neural Information Processing Systems, in: NIPS'09, Curran Associates Inc., USA, 2009, pp. 2259–2267.

[36] X. Zhao, W. Zhang, J. Wang, Interactive collaborative filtering, in: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, in: CIKM '13, ACM, New York, NY, USA, 2013, pp. 1411–1420.
[37] Z. Zhao, H. Lu, D. Cai, X. He, Y. Zhuang, User preference learning for online social recommendation, IEEE Trans. Knowl. Data Eng. 28 (9) (2016) 2522–2534.
[38] X. Zhou, W. Shu, F. Lin, B. Wang, Confidence-weighted bias model for online collaborative filtering, Appl. Soft Comput. 70 (2018) 1042–1053.
[39] X. Zhou, S. Wu, Rating LDA model for collaborative filtering, Knowl.-Based Syst. 110 (2016) 135–143.