Neurocomputing 272 (2018) 606–618

Contents lists available at ScienceDirect

Neurocomputing

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

Incremental Slope-one recommenders ☆

Qing-Xian Wang [a,1], Xin Luo [b,c,1,*], Yan Li [b,c], Xiao-Yu Shi [b,c], Liang Gu [d,e], Ming-Sheng Shang [b,c]

[a] School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China
[b] Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China
[c] College of Computer Science and Engineering, Shenzhen University, Shenzhen 518060, China
[d] Jinan University, Guangzhou, Guangdong 510632 China
[e] Sangfor Technology Incorporation, Shenzhen, Guangdong 518057 China

ARTICLE INFO

Article history:
Received 5 March 2017
Revised 16 June 2017
Accepted 13 July 2017
Available online 20 July 2017

Communicated by Nianyin Zeng

Keywords:
Collaborative filtering
Slope-one
Recommender system
Dynamic datasets
Incremental recommenders

ABSTRACT

Collaborative filtering (CF)-based recommenders work by estimating a user's potential preferences on unobserved items referring to the other users' observed preferences. Slope-one, as a well-known CF recommender, is widely adopted in industrial applications owing to it's (a) competitive prediction accuracy for user's potential preferences, (b) high computational efficiency, and (c) ease of implementation. However, current Slope-one-based algorithms are all designed for static datasets, which are contradictory to real situations where dynamic datasets are mostly involved. This paper focuses on designing incremental Slope-one recommenders able to address dynamic datasets, reflecting their variations instantly without retraining the whole model. To do so, we have carefully analyzed the parameter training processing of Slope-one-based recommenders to design the incremental update rules for involved parameters reflecting data increments in dynamic environments. Three incremental Slope-one recommenders, including the incremental Slope-one, incremental weighted Slope-one, and incremental bi-polar slope one, are proposed. Experimental results on two large real datasets indicate that the proposed incremental slope-one recommenders can correctly reflect the increments of dynamic datasets with high computational efficiency.

1. Introduction

Recommender systems make personalized recommendations for information, products or services by applying information filtering techniques [1–4]. Hence, they are considered as the most promising technologies to cope with the information overload problem. Most of the famous IT companies, such as, Amazon, Netflix, YouTube, Taobao, and many others adopt recommender systems to improve user's shopping experience and satisfaction for increasing their profits further [5–6].

Until now, different kinds of recommenders have been proposed based on various methods, e.g., the content-based methods [7], the network-based methods [9], collaborative filtering (CF)-based methods [8] and hybrid methods [10]. Among them, CF-based recommenders occupy the mainstream in recommendation [11–21]. According to the recent progress in CF-based recommenders, they can group into two different types of recommendation including the entity relationship-based recommendation and model-based recommendation. Specifically, the entity relationship-based recommendation uses the known user-item ratings to directly estimate the ratings for new items. It can be done in two ways known as item-based and user-based recommendation [8]. Compared with entity relationship-based recommendation, model-based recommendation works by learning a predictive model to estimate the missing data (i.e., unknown ratings), such as slope one algorithm [22], matrix factorization-based algorithm [11–13].

Slope-one [22], as a well-known CF recommender, is widely adopted in industrial applications owing to it's (a) competi-

☆ This work was supported in part by the National Key Research and Development Program of China under grant 2017YFC0804002, in part by the Pioneer Hundred Talents Program of Chinese Academy of Sciences, in part by the International Joint Project through the Royal Society of the U.K. and the National Natural Science Foundation of China under Grant 61611130209, by the National Natural Science Foundation of China under Grant 61402198, Grant 61602434, Grant 91646114, and Grant 61672136, in part by the Young Scientist Foundation of Chongqing under Grant cstc2014kjrc-qnrc40005, in part by the Chongqing Research Program of Basic Research and Frontier Technology under Grant cstc2015jcyjB0244 and Youth Innovation Promotion Association CAS, No. 2017393.

* Corresponding author at: Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China

E-mail addresses: 617675424@qq.com (Q.-X. Wang), luoxin21@cigit.ac.cn, luoxin21@gmail.com (X. Luo), 1466581701@qq.com (Y. Li), xiaoyushisxy@cigit.ac.cn (X.-Y. Shi), os.liang@gmail.com (L. Gu), msshang@cigit.ac.cn (M.-S. Shang).

[1] Q.-X. Wang and X. Luo contributed equally to this work.

http://dx.doi.org/10.1016/j.neucom.2017.07.033
0925-2312/

tive prediction accuracy for user's potential preferences, (b) high computational efficiency, and (c) ease of implementation [22–32]. A Slope-one recommender generates predictions based on existing data via solving a linear system whose slope is fixed at one.

So far lots of advanced recommenders based on Slope-one algorithm have been proposed. For improving the prediction accuracy of recommender, You et al. [33] proposed a hybrid recommender to improve the recommendation accuracy by adopting Slope-one algorithm with the function of item clustering. Zhao and Ma. [34] developed an improved Slope-one algorithm with consideration of the frequency information of items' tags. Jiang and Lu. [35] presented a novel Slope-one algorithm that assigns different weights to items at different time. For dealing with the data sparsity issue, Li et al. [36] proposed a novel CF recommender that combines uncertain neighbors with slope-one algorithm, which shows better recommendation quality and robust than original Slope-one algorithm. Zhang et al. [37] introduced a novel user similarity measurement based on user favorite items, and integrated into Slope-one algorithm for rating prediction. Furthermore, in order to further meet the user's preference, Wang et al. [38] proposed an individualized recommender based on Slope-one algorithm that can adjust the slope parameter to each user.

However, current Slope-one-based algorithms are all designed for static datasets, which are contradictory to real situations where dynamic datasets are mostly involved [39–44]. Note that in real applications, data increments can arrive at every millisecond, making a target dataset constantly change. Therefore, incremental recommenders which are able to address such data dynamics are greatly desired [39–43]

An effective way to solve this problem is that making the recommender adaptive to rapid rating-variations. Hence, incremental models combine sequential processes in an interactive way become popular [39–43]. Papagelis et al. [39] proposed an incremental user-oriented K-Nearest-Neighborhood (KNN) CF recommender, which focus on the similarity between ratings. Based on this work, Luo et al. [42] proposed a boosted incremental KNN model. Regarding of model-based CF recommender, Luo et al. [43] also developed an incremental-and-static-combined scheme for model-based CF recommender, which can gain highly accurate and low computation cost. Compared with these existing works, we focus on designing the incremental model based on slope one algorithm. Owing to the effectiveness and popularity of the Slope-one model in implementing personalized recommendations, it is highly significant to implement incremental recommenders based on it. To do so, the algorithm of Slope-one should provide the predicated solution for the constantly changing data.

This work focuses on implementing incremental Slope-one recommenders, which are able to reflect the change of dynamic datasets instantly without re-processing a whole dataset. The main idea is to deduce the incremental update rules for parameters related with dynamic data variations, thereby making the whole model be aware of such data variations. The main contributions of this work include:

(1) Three incremental Slope-one recommenders which integrate the incremental information into the model by addressing the part of parameters related to the data increments only.
(2) Empirical validations on two large datasets collected by real recommender systems.

The rest of the paper is organized as follows. Section 2 provides a brief overview of the basic Slope-one scheme and notations which are used in this paper. Section 3 outlines our incremental algorithm. Section 4 presents and analyzes our experiment results. Finally, Section 5 concludes this paper.

Table 1
Symbol appointments.

Symbol	*Explanation*
U/I	The set of users or items
u,v,w	Represent specific user
i,j	Represent specific item
R	User-item rating matrix
U_i/I_u	The set of users who rated item i/items rated by user u.
$\bar{r}_u/\bar{r}_i$	Average rating for user $u \in U$/item $i \in I$
$r_{u,i}/\hat{r}_{u,i}$	The rating/predict rating of user u on item i.
$\bar{r}'_i/\bar{r}'_u$	Updated $\bar{r}_i/\bar{r}_u$
$r'_{u,i}/r'_{v,i}$	Updated $r_{u,i}/r_{v,i}$

Table 2
Notations used in the Slope-one model with user deviation.

Factors	*Comments*	*Related to*
$c_{u,v} = \lvert I_u \cap I_v \rvert$	$dev_{u,v}$ components	$dev_{u,v}$
$c^{like}_{u,v} = \lvert I^{like}_{u,v} \rvert$	$dev^{like}_{u,v}$ components	$dev^{like}_{u,v}$
$c^{dislike}_{u,v} = \lvert I^{dislike}_{u,v} \rvert$	$dev^{dislike}_{u,v}$ components	$dev^{dislike}_{u,v}$
$\bar{r}_j$	Average rating of item j	Each item j
$U^{like}_j = \{u \in U_j \lvert r_{u,j} > \bar{r}_j\}$	Users that like item j	Each item j
$U^{dislike}_j = \{u \in U_j \lvert r_{u,j} \le \bar{r}_j\}$	Users that dislike item j	Each item j
$I^{like}_{u,v} = \{j \in \chi \lvert u,v \in U^{like}_j\}$	Items that user $u,v \in U^{like}_j$	Each user u,v
$I^{dislike}_{u,v} = \{j \in \chi \lvert u,v \in U^{dislike}_j\}$	Items that user $u,v \in U^{dislike}_j$	Each user u,v

Table 3
Notations used in the Slope-one model based on item deviation.

Factors	*Comments*	*Related to*
$c_{j,i} = \lvert U_j \rvert \cap \lvert U_i \rvert$	$dev_{j,i}$ components	$dev_{j,i}$
$c^{like}_{i,j} = \lvert U^{like}_{i,j} \rvert$	$dev^{like}_{j,i}$ components	$dev^{like}_{j,i}$
$c^{dislike}_{i,j} = \lvert U^{dislike}_{i,j} \rvert$	$dev^{dislike}_{j,i}$ components	$dev^{dislike}_{j,i}$
$\bar{r}_u$	Average rating of each user u	Each user u
$I^{like}_u = \{j \in I_u \lvert r_{u,j} > \bar{r}_u\}$	Items that user u like	Each user u
$I^{dislike}_u = \{j \in I_u \lvert r_{u,j} \le \bar{r}_u\}$	Items that user u dislike	Each user u
$U^{like}_{i,j} = \{u \in \chi \lvert i,j \in I^{like}_u\}$	Users that item $i,j \in I^{like}_u$	Each item i,j
$U^{dislike}_{i,j} = \{u \in \chi \lvert i,j \in I^{dislike}_u\}$	Users that item $i,j \in I^{dislike}_u$	Each item i,j

2. Preliminaries

2.1. Symbols

Definitions. Given a user set U and an item set I, a user-item matrix R is a $|U| \times |I|$ matrix. This matrix consisting of the rating user u gives to item i, which can be denoted by $r_{u,i}$, while $\hat{r}_{u,i}$ denotes the predicted rating. Specifically, symbols like $r'_{u,i}$ and $\bar{r}'_u$ stand for their updated values. The symbol appointments are given in Table 1.

Notations. In this paper, the Slope-one algorithm is from the aspect of user-based and item-based deviation respectively, which can be proved to be updated in a similar. Even user-based algorithm mainly focuses on the average deviation among the ratings from one user while item-based algorithm focuses on deviation of items, the prediction processes of these two models are same. For different datasets, we can conduct an experiment by using specific algorithm for the accurate and efficient calculation. The factors related to the deviation are listed in Tables 2 and 3 from the above two aspects.

As defined in [5–16], for any two users u and v, the ratings from them are denoted as $r_{u,i}$ and $r_{v,i}$ separately for their preferences on item i. According to the principle of a Slope-one recommender [16], in order to predicate the ratings from the aspect of user-based deviation, we pre-compute the average deviations of user u to user

v in the set of $I_u \cap I_v$, $I_{u,v}^{like}$, $I_{u,v}^{dislike}$ as follows:

$$\begin{aligned} dev_{u,v} &= \sum_{i \in I_u \cap I_v} \frac{r_{u,i} - r_{v,i}}{|I_u \cap I_v|}, c_{u,v} = |I_u \cap I_v| \\ dev_{u,v}^{like} &= \sum_{i \in I_{u,v}^{like}} \frac{r_{u,i} - r_{v,i}}{card\left(I_{u,v}^{like}\right)}, c_{u,v}^{like} = card\left(I_{u,v}^{like}\right) \\ dev_{u,v}^{dislike} &= \sum_{i \in I_{u,v}^{dislike}} \frac{r_{u,i} - r_{v,i}}{card\left(I_{u,v}^{dislike}\right)}, c_{u,v}^{dislike} = card\left(I_{u,v}^{dislike}\right) \end{aligned}, \quad (1)$$

where $dev_{u,\,v}$, $dev_{u,v}^{like}$ and $dev_{u,v}^{dislike}$ stand for the average deviation in Slope-one algorithm based on user deviation.

Meanwhile, the average deviations of item j with respect to item i in the set of $U_i \cap U_j$, $U_{i,j}^{like}$, $U_{i,j}^{dislike}$ can be regarded as:

$$\begin{aligned} dev_{j,i} &= \sum_{u \in U_i \cap U_j} \frac{r_{u,j} - r_{u,i}}{|U_i \cap U_j|}, \; c_{j,i} = \left|U_i \cap U_j\right| \\ dev_{j,i}^{like} &= \sum_{u \in U_{i,j}^{like}} \frac{r_{u,j} - r_{u,i}}{card\left(U_{i,j}^{like}\right)}, \; c_{j,i}^{like} = card\left(U_{i,j}^{like}\right) \\ dev_{j,i}^{dislike} &= \sum_{u \in U_{i,j}^{dislike}} \frac{r_{u,j} - r_{u,i}}{card\left(U_{i,j}^{dislike}\right)}, c_{j,i}^{dislike} = card\left(U_{i,j}^{dislike}\right) \end{aligned}, \quad (2)$$

where $dev_{j,\,i}$, $dev_{j,i}^{like}$ and $dev_{j,i}^{dislike}$ stand for the average deviation in Slope-one algorithm based on item deviation.

2.2. Static Slope-one recommenders

In 2005, Lemire and Maclachlan. [22] proposed the Slope-one algorithm with three different scheme including basic Slope one scheme, the weighted slope one scheme and the bi-polar slope one scheme, respectively.

2.2.1. Slope-one scheme

The Slope-one Scheme based on user deviation relies on the average rating of target item and depends crucially on the users who have rated the item rather than the features of the individual item. Meanwhile, the Slope-one Scheme based on the item deviation depends on the average rating of users and the items that user has rated rather than how individual user rated [16]. Then the predicted formulas are given by:

$$\hat{r}_{u,j} = \frac{\sum_{v \in U_j - \{u\}} \left(dev_{u,v} + r_{v,j}\right)}{\left|U_j - \{u\}\right|} \quad (3)$$

$$\hat{r}_{u,j} = \frac{\sum_{i \in I_u - \{j\}} \left(dev_{j,i} + r_{u,i}\right)}{|I_u - \{j\}|}, \quad (4)$$

where $|U_j - \{u\}|$ stands for the user numbers who have rated item j except u, while $|I_u - \{j\}|$ is the item number that user u has rated except j.

2.2.2. Weighted Slope-one scheme

One of the shortcomings of the Slope-one Scheme is that there is no consideration of the numbers of common rating items for user-based deviation or common user numbers for item-based deviation [16]. Take the Weight Slope-one Scheme based on user deviation for example, in order to predict the rating, when user u and v have commonly rated 1000 items, whereas 100 items for u and w, it is obvious to take a higher weight for the users who share more common rating items with u namely v for a better prediction. Thus, we can modify (3) and (4) by adding a weight $c_{u,v}$. Consequently, we define the Weighted Slope-one prediction as the following weighted average:

$$\hat{r}_{u,j} = \sum_{v \in U_j - \{u\}} \left(dev_{u,v} + r_{v,j}\right) c_{u,v} / \sum_{v \in U_j - \{u\}} c_{u,v} \quad (5)$$

$$\hat{r}_{u,j} = \sum_{i \in I_u - \{j\}} \left(dev_{j,i} + r_{u,i}\right) c_{j,i} / \sum_{i \in I_u - \{j\}} c_{j,i}. \quad (6)$$

2.2.3. Bi-polar Slope-one scheme

Occasionally, there is a problem that the weight used in the Weighted Slope-one Scheme may serve as a contradict role, causing the higher or lower predicted rating than it ought to be. For example, users v share 1000 rating items with u denoted as $I_{u,v}$ while users w share 100 items denoted as $I_{u,w}$. Most of ratings of u in $I_{u,v}$ is higher than $\bar{r}_i$ ($i \subset I_{u,\,v}$) whilst most of ratings of v in $I_{u,v}$ is lower than $\bar{r}_i$ ($i \subset I_{u,\,v}$) in contract to a higher rating of w in $I_{u,w}$. If we adopt a relatively higher weight for v, it will lead to an inevitable error. Therefore, the Bi-Polar Slope-one Scheme is designed to divide users into optimistic users and pessimistic users [16] who like or dislike the item they rate. We also apply this kind of division to the Bi-Polar Slope-one Scheme into like items and dislike items [16]. Thus, (5) and (6) are turned into:

$$\hat{r}_{u,j} = \frac{\sum_{v \in U_j^{like} - \{u\}} \left(dev_{u,v}^{like} + r_{v,j}\right) c_{u,v}^{like} + \sum_{v \in U_j^{dislike} - \{u\}} \left(dev_{u,v}^{dislike} + r_{v,j}\right) c_{u,v}^{dislike}}{\sum_{v \in U_j^{like} - \{u\}} c_{u,v}^{like} + \sum_{v \in U_j^{dislike} - \{u\}} c_{u,v}^{dislike}} \quad (7)$$

$$\hat{r}_{u,j} = \frac{\sum_{i \in I_u^{like} - \{j\}} \left(dev_{j,i}^{like} + r_{u,i}\right) c_{j,i}^{like} + \sum_{i \in I_u^{dislike} - \{j\}} \left(dev_{j,i}^{dislike} + r_{u,i}\right) c_{j,i}^{dislike}}{\sum_{i \in I_u^{like} - \{j\}} c_{j,i}^{like} + \sum_{i \in I_u^{dislike} - \{j\}} c_{j,i}^{dislike}} \quad (8)$$

2.3. Incremental problem

Compared with the basic Slope-one Scheme, the improved scheme replaces the average deviation $dev_{u,\,v}$ or $dev_{i,\,j}$ with the quotient of two factors $b_{u,v}$, $c_{u,v}$ and $b_{i,j}$, $c_{i,j}$ respectively. When the dataset varies, the algorithm dynamically updates the two factors by caching each component and related factors. Thus, the corresponding deviation can be incrementally updated according to the change of ratings without retraining the whole model. The factors related to the incremental-update of Slope-one predictor are listed in Tables 4 and 5. Thereinto, CU_j^{like}, $CU_j^{dislike}$, NCU_j^{like}, $NCU_j^{dislike}$, CI_u^{like}, NCI_u^{like}, $CI_u^{dislike}$ and $NCI_u^{dislike}$ are cached users or items set used for marking changed users or items.

3. Incremental Slope-one models

3.1. Incremental Slope-one

Problem formulation. As mentioned above, $dev_{u,\,v}$ or $dev_{i,\,j}$ is equal to the quotient of two factors $b_{u,v}$, $c_{u,v}$ and $b_{i,j}$, $c_{i,j}$ for the Slope-one and Weighted Slope-one Scheme. Then (3) and (4) are changed into:

$$\hat{r}_{u,j} = \frac{\sum_{v \in U_j - \{u\}} \left(b_{u,v}/c_{u,v} + r_{v,j}\right)}{\left|U_j - \{u\}\right|} \quad (9)$$

$$\hat{r}_{u,j} = \frac{\sum_{i \in I_u - \{j\}} \left(b_{j,i}/c_{j,i} + r_{u,i}\right)}{|I_u - \{j\}|} \quad (10)$$

Update rule. As shown above, the incremental update rules of user-based and item-based deviation follow the same idea. Thus, in our paper, we mainly discuss about the rule from the user-based aspect. Regarding to a new rating, the increment rules can be

Table 4
Cached factors for incrementally updating user-based deviation.

Factors	*Comments*	*Related to*
$b_{u,v} = \sum_{j \in I_u \cap I_v} r_{u,j} - r_{v,j}$	$dev_{u,v}$ components	$dev_{u,v}$
$b^{like}_{u,v} = \sum_{j \in I^{like}_{u,v}} (r_{u,j} - r_{v,j})$	$dev^{like}_{u,v}$ components	$dev^{like}_{u,v}$
$b^{dislike}_{u,v} = \sum_{j \in I^{dislike}_{u,v}} (r_{u,j} - r_{v,j})$	$dev^{dislike}_{u,v}$ components	$dev^{dislike}_{u,v}$
$CU^{like}_j = \{u \in U^{like}_j \mid r_{u,j} > \bar{r}_j \text{ and } r_{u,j} \le \bar{r}'_j\}$	Users who like item $_j$ change to dislike user	$dev^{like}_{u,v}$,$dev^{dislike}_{u,v}$
$NCU^{like}_j = \{u \in U^{like}_j \mid r_{u,j} > \bar{r}_j \text{ and } r_{u,j} > \bar{r}'_j\}$	Users who like item $_j$ do not change to dislike user	$dev^{like}_{u,v}$, $dev^{dislike}_{u,v}$
$CU^{dislike}_j = \{u \in U^{dislike}_j \mid r_{u,j} \le \bar{r}_j \text{ and } r_{u,j} > \bar{r}'_j\}$	Users who dislike item $_j$ change to like user	$dev^{like}_{u,v}$, $dev^{dislike}_{u,v}$
$NCU^{dislike}_j = \{u \in U^{dislike}_j \mid r_{u,j} \le \bar{r}_j \text{ and } r_{u,j} \le \bar{r}'_j\}$	Users who dislike item $_j$ do not change to like user	$dev^{like}_{u,v}$, $dev^{dislike}_{u,v}$

Table 5
Cached factors for incrementally updating item-based deviation.

Factors	*Comments*	*Related to*
$b_{j,i} = \sum_{u \in U_i \cap U_j} r_{u,i} - r_{u,j}$	$dev_{j,i}$ components	$dev_{j,i}$
$b^{like}_{i,j} = \sum_{u \in U^{like}_{i,j}} (r_{u,i} - r_{u,j})$	$dev^{like}_{j,i}$ components	$dev^{like}_{j,i}$
$b^{dislike}_{i,j} = \sum_{u \in U^{dislike}_{i,j}} (r_{u,i} - r_{u,j})$	$dev^{dislike}_{j,i}$ components	$dev^{dislike}_{j,i}$
$CI^{like}_u = \{i \in I^{like}_u \mid r_{u,i} > \bar{r}_u \text{ and } r_{u,i} \le \bar{r}'_u\}$	Items that $_u$ like change to dislike	$dev^{like}_{j,i}$, $dev^{dislike}_{j,i}$
$NCI^{like}_u = \{i \in I^{like}_u \mid r_{u,i} > \bar{r}_u \text{ and } r_{u,i} > \bar{r}'_u\}$	Items that $_u$ like do not change to dislike	$dev^{like}_{j,i}$, $dev^{dislike}_{j,i}$
$CI^{dislike}_u = \{i \in I^{dislike}_u \mid r_{u,i} \le \bar{r}_u \text{ and } r_{u,i} > \bar{r}'_u\}$	Items that $_u$ dislike change to like	$dev^{like}_{j,i}$, $dev^{dislike}_{j,i}$
$NCI^{dislike}_u = \{i \in I^{dislike}_u \mid r_{u,i} \le \bar{r}_u \text{ and } r_{u,i} \le \bar{r}'_u\}$	Items that $_u$ dislike do not change to like	$dev^{like}_{j,i}$, $dev^{dislike}_{j,i}$

concluded as follows,

A. With new $r_{v,j}$

$$case1: u \notin U_j \text{: no update}$$

$$case2: u \in U_j: \begin{cases} b'_{u,v} = b_{u,v} + r_{u,j} - r_{v,j} \\ b'_{v,u} = -b'_{u,v} \\ c'_{u,v} = c_{u,v} + 1 \\ c'_{v,u} = c'_{u,v} \end{cases} \tag{11}$$

B. With changed $r'_{u,j}$

$$case1: v \notin U_j \text{: no update}$$

$$case2: v \in U_j: \begin{cases} b'_{u,v} = b_{u,v} + r'_{u,j} - r_{u,j} \\ b'_{v,u} = -b'_{u,v} \\ c'_{u,v} = c_{u,v} \\ c'_{v,u} = c'_{u,v} \end{cases} \tag{12}$$

In (11) and (12), we consider four different situations according to the updating requirements. For a rating, the incremental algorithm firstly judges whether u and v are in the set U_j or not. If they do not belong to it, then there is no update on their deviation with respect to the changed user. We consider the following two cases: (1) for a new rating, the main task is to take this rating into consideration and compute the increment on involved parameters; and (2) for a modified rating, since the older version is considered during the model-building process, the main task is to construct the difference on the involved parameters brought by this modification [16,27–31]. In addition, since $b_{u,v}$ stands for the deviation of u with respect to v, $b_{v,u}$ is equal to the opposite of $b_{u,v}$.

3.2. Incremental and weighted Slope-one

Problem formulation. According to the transform form of basic Slope-one scheme, it is easy to obtain the prediction rule of incremental and weighted Slope-one predictor as:

$$\hat{r}_{u,j} = \frac{\sum_{v \in U_j - \{u\}} \left(b_{u,v}/c_{u,v} + r_{v,j}\right) c_{u,v}}{\sum_{v \in U_j - \{u\}} c_{u,v}} \tag{13}$$

$$\hat{r}_{u,j} = \frac{\sum_{i \in I_u - \{j\}} \left(b_{j,i}/c_{j,i} + r_{u,i}\right) c_{j,i}}{\sum_{i \in I_u - \{j\}} c_{j,i}}. \tag{14}$$

Update Rule. Since the weighted Slope-one employs the same deviation with the Slope-one, the update rule is the same as the incremental Slope-one.

3.3. Incremental and Bi-polar Slope-one

Problem formulation. The Bi-Polar Scheme divides users into optimistic and pessimistic users. Therefore, the deviation consists of like and dislike factors accordingly. As a result, $dev^{like}_{u,v}$ and $dev^{like}_{i,j}$ are changed into quotient of two factors $b^{like}_{u,v}$, $c^{like}_{u,v}$ and $b^{like}_{i,j}$, $c^{like}_{i,j}$ while $dev^{dislike}_{u,v}$ and $dev^{dislike}_{i,j}$ are changed into quotient of two factors $b^{dislike}_{u,v}$, $c^{dislike}_{u,v}$ and $b^{dislike}_{i,j}$, $c^{dislike}_{i,j}$ separately. Consequently, (7)

Table 6
Incremental process for a new rating.

Step	Case	Branch	Description	Related equation
1	1		Insert user v into U_j^{like} when $r_{v,j} > \bar{r}'_j$	17
1	2		Insert user v into $U_j^{dislike}$ when $r_{v,j} < \bar{r}'_j$	17
2	1		Do nothing if each user u not belongs to U_j^{like} and $U_j^{dislike}$	18
2	2	a	Mark u in related cached factors listed in Table 4, then update deviation on the condition that $u \in U_j^{dislike}, r_{u,j} > \bar{r}'_j$ and $r_{v,j} > \bar{r}'_j$	19
2	2	b	Mark u in related cached factors in Table 4, update deviation on the condition that $u \in U_j^{dislike}, r_{u,j} < \bar{r}'_j$ and $r_{v,j} < \bar{r}'_j$.	20
2	3	a	Mark u in related cached factors in Table 4, update deviation on the condition that $u \in U_j^{like}, r_{u,j} > \bar{r}'_j$ and $r_{v,j} > \bar{r}'_j$.	21
3	3	b	Mark u in related cached factors in Table 4, update deviation on the condition that $u \in U_j^{like}, r_{u,j} < \bar{r}'_j$ and $r_{v,j} < \bar{r}'_j$.	22
3	1		Update $b_{u,v}^{like}$, $c_{u,v}^{like}$, $b_{u,v}^{dislike}$ and $c_{u,v}^{dislike}$ when $u, v \in CU_j^{like}$	23
3	2		Update $b_{u,v}^{like}$ and $c_{u,v}^{like}$ when $u \in CU_j^{like}$, $v \in NCU_j^{like}$	24
3	3		Update $b_{u,v}^{dislike}$ and $c_{u,v}^{dislike}$ when $u \in CU_j^{like}, v \in NCU_j^{dislike}$	25
3	4		Update $b_{u,v}^{like}$, $c_{u,v}^{like}$, $b_{u,v}^{dislike}$ and $c_{u,v}^{dislike}$ when $u, v \in CU_j^{dislike}$	26
3	5		Update $b_{u,v}^{dislike}$ and $c_{u,v}^{dislike}$ when $u \in CU_j^{dislike}$, $v \in NCU_j^{dislike}$	27
3	6		Update $b_{u,v}^{like}$ and $c_{u,v}^{like}$ when $u \in CU_j^{dislike}$, $v \in NCU_j^{like}$	28

Table 7
Incremental process for a changed rating.

Step	Case	Branch	Description	Related equation
1	1	a	Insert u into NCU_j^{like} when $r_{u,j} > \bar{r}_j, r'_{u,j} > \bar{r}'_j$	29
1	1	b	Remove u from U_j^{like} and insert it into $U_j^{dislike}$ and CU_j^{like} when $r_{u,j} > \bar{r}_j, r'_{u,j} < \bar{r}'_j$	29
1	2	a	Remove u from $U_j^{dislike}$ and insert it into U_j^{like} and $CU_j^{dislike}$ when $r_{u,j} > \bar{r}_j, r'_{u,j} > \bar{r}'_j$	30
1	2	b	Insert u into $NCU_j^{dislike}$ when $r_{u,j} > \bar{r}_j, r'_{u,j} < \bar{r}'_j$	30
2	1		Do nothing if each user u not belongs to U_j^{like} and $U_j^{dislike}$	31
2	2	a	Remove v from $U_j^{dislike}$ and insert it into U_j^{like} and $CU_j^{dislike}$ when $v \in U_j^{dislike}, r_{v,j} > \bar{r}'_j$	32
2	2	b	Insert v into $NCU_j^{dislike}$ when $v \in U_j^{dislike}, r_{v,j} < \bar{r}'_j$	32
2	3	a	Insert v into NCU_j^{like} when $v \in U_j^{like}, r_{v,j} > \bar{r}'_j$	33
2	3	b	Remove v from U_j^{like} and insert it into $U_j^{dislike}$ and CU_j^{like} when $v \in U_j^{like}, r_{v,j} < \bar{r}'_j$	33
3			Compute deviation as step 3 for the new rating	34

and (8) are reformulated as:

$$\hat{r}_{u,j} = \frac{\sum_{v \in U_j^{like} - \{u\}} \left(\frac{b_{u,v}^{like}}{c_{u,v}^{like}} + r_{v,j} \right) c_{u,v}^{like} + \sum_{v \in U_j^{dislike} - \{u\}} \left(\frac{b_{u,v}^{dislike}}{c_{u,v}^{dislike}} + r_{v,j} \right) c_{u,v}^{dislike}}{\sum_{v \in U_j^{like} - \{u\}} c_{u,v}^{like} + \sum_{v \in U_j^{dislike} - \{u\}} c_{u,v}^{dislike}} \tag{15}$$

$$\hat{r}_{u,j} = \frac{\sum_{i \in I_u^{like} - \{j\}} \left(\frac{b_{j,i}^{like}}{c_{j,i}^{like}} + r_{u,i} \right) c_{j,i}^{like} + \sum_{i \in I_u^{dislike} - \{j\}} \left(\frac{b_{j,i}^{dislike}}{c_{j,i}^{dislike}} + r_{u,i} \right) c_{j,i}^{dislike}}{\sum_{i \in I_u^{like} - \{j\}} c_{j,i}^{like} + \sum_{i \in I_u^{dislike} - \{j\}} c_{j,i}^{dislike}}. \tag{16}$$

Update rule. Different from incremental rules of the first two schemes, Bi-Polar Slope-one needs to take into account more complex circumstances. At the beginning, we need to identify the change source, which distinguish the changed rating from the new rating. In general, this process can be summarized as below. Firstly, we update the average rating, and add user v into U_j^{like} or $U_j^{dislike}$. Secondly, we deal with each user in U_j^{like} and $U_j^{dislike}$ to redistribute it and mark in related cached factors listed in Tables 4 and 5 with computing the difference brought by such change. Finally, incremental deviation factors should be recomputed on the basis of four changed set which is updated in step 2. Detailed processing steps are concluded in Tables 6 and 7.

Table 8
Tested models.

No.	Description
M_1	Base Slope-one based on user deviation.
M_2	Weighted Slope-one based on user deviation.
M_3	Bi-polar Slope-one based on user deviation.
M_4	Base Slope-one based on item deviation.
M_5	Weighted Slope-one based on item deviation.
M_6	Bi-polar Slope-one based on item deviation.
M_7	Incremental Slope-one based on user deviation.
M_8	Incremental and Weighted Slope-one based on user deviation.
M_9	Incremental and Bi-polar Slope-one based on user deviation.
M_10	Incremental Slope-one based on item deviation.
M_11	Incremental and Weighted Slope-one based on item deviation.
M_12	Incremental and Bi-polar Slope-one based on item deviation.

A. With new rating $r_{v,j}$

$$\begin{cases} \bar{r}'_j = \frac{r_{v,j} + \bar{r}_j \cdot n_j}{1 + n_j} \\ n'_j = n_j + 1 \\ step\ 1, case\ 1 (r_{v,j} > \bar{r}'_j): \\ U_j'^{like} = U_j^{like} + \{v\} \\ step\ 1, case\ 2 (r_{v,j} < \bar{r}'_j): \\ U_j'^{dislike} = U_j^{dislike} + \{v\} \end{cases} \tag{17}$$

In (17), a new rating $r_{v,j}$ is considered at first. Our prediction is presented for item j by user u on deviation from pairs of users

who have rated both item i and j with sharing a like or dislike on item i. Thus, it is necessary to divide user v into U_j^{like} or $U_j^{dislike}$. The second step is summarized as three cases in (18–22). In the first case, we take no account of users who do not belong to U_j^{like} or $U_j^{dislike}$. The rest two cases respectively illustrate the comparison of $r_{u,j}$, $r_{v,j}$ with updated $\bar{r}_j$ in U_j^{like} and $U_j^{dislike}$, and then update incremental factors between u and v. This process is formulated by:

$$\begin{aligned} &step\ 2(\forall u \in U_j), case\ 1: \\ &u \notin U_j^{like} \cap u \notin U_j^{dislike} : no\ update \end{aligned} \tag{18}$$

$$\begin{cases} \begin{cases} U_j'^{like} = U_j^{like} + \{u\} \\ U_j'^{dislike} = U_j^{dislike} - \{u\} \\ CU_j'^{dislike} = CU_j^{dislike} + \{u\} \end{cases} \\ if\ r_{v,j} > \bar{r}_j' \begin{cases} b_{u,v}'^{like} = b_{u,v}^{like} + r_{u,j} - r_{v,j} \\ b_{v,u}'^{like} = -b_{u,v}'^{like} \\ c_{u,v}'^{like} = c_{u,v}^{like} + 1 \\ c_{v,u}'^{like} = c_{u,v}'^{like} \end{cases} \end{cases} \tag{19}$$

$$\begin{cases} NC_j'^{dislike} = NC_j'^{dislike} + \{u\} \\ if\ r_{v,j} < \bar{r}_j' \begin{cases} b_{u,v}'^{dislike} = b_{u,v}^{dislike} + r_{u,j} - r_{v,j} \\ b_{v,u}'^{dislike} = -b_{u,v}'^{dislike} \\ c_{u,v}'^{dislike} = c_{u,v}^{dislike} + 1 \\ c_{v,u}'^{dislike} = c_{u,v}'^{dislike} \end{cases} \end{cases} . \tag{20}$$

According to (19) and (20), for dealing with $u \in U_j^{dislike}$, if $r_{u,j} > \bar{r}_j'$, it means that u no longer serves as a pessimistic user like before. Then it will be removed from $U_j^{dislike}$ to U_j^{like} and $CU_j^{dislike}$. After that, we need to discuss about the value of $r_{v,j}$ to update incremental factors. $b_{u,v}^{like}$, $c_{u,v}^{like}$, $b_{u,v}^{dislike}$ and $c_{u,v}^{dislike}$ are computed only when u and v in the same set. Therefore, any case that u and v exist in the same set during the update process will cause the change of four factors above. When the rating of u and v is simultaneously higher than $\bar{r}_j'$, the incremental algorithm automatically updates the value of $b_{u,v}^{like}$ and $c_{u,v}^{like}$. As for another case namely $r_{u,j} < \bar{r}_j'$, it means that u also serves as pessimistic user like before. After that, it will be added in $NC_j^{dislike}$. Next, if $r_{v,j} < \bar{r}_j'$, all related variables are updated according to (20). Another case is that user u exists in U_j^{like} which can be seen in (21) and (22).

$$\begin{cases} NC_j'^{like} = NC_j'^{like} + \{u\} \\ if\ r_{v,j} > \bar{r}_j' : \begin{cases} b_{u,v}'^{like} = b_{u,v}^{like} + r_{u,j} - r_{v,j} \\ b_{v,u}'^{like} = -b_{u,v}'^{like} \\ c_{u,v}'^{like} = c_{u,v}^{like} + 1 \\ c_{v,u}'^{like} = c_{u,v}'^{like} \end{cases} \end{cases} \tag{21}$$

$$\begin{cases} \begin{cases} U_j'^{like} = U_j^{like} - \{u\} \\ U_j'^{dislike} = U_j^{dislike} + \{u\} \\ CU_j'^{like} = CU_j^{like} + \{u\} \end{cases} \\ if\ r_{v,j} < \bar{r}_j' \begin{cases} b_{u,v}'^{dislike} = b_{u,v}^{dislike} + r_{u,j} - r_{v,j} \\ b_{v,u}'^{dislike} = -b_{u,v}'^{dislike} \\ c_{u,v}'^{dislike} = c_{u,v}^{dislike} + 1 \\ c_{v,u}'^{dislike} = c_{u,v}'^{dislike} \end{cases} \end{cases} \tag{22}$$

By comparing the processing step of user in U_j^{like} with user in $U_j^{dislike}$, we may find they are similar in thought. In (21), we update NC_j^{like} by inserting user u into it when $u \in U_j^{like}$ and $r_{u,j} > \bar{r}_j'$, denoting that u also serves as optimistic user like before. Meanwhile, we update $b_{u,v}^{like}$, $c_{u,v}^{like}$ when $r_{v,j} > \bar{r}_j'$, since u and v share a common set namely U_j^{like}.

In (22), u is removed from U_j^{like} to $U_j^{dislike}$ and CU_j^{like} separately when $r_{u,j} < \bar{r}_j'$, which means that u no longer serves as an optimistic user like before. Afterwards, the incremental algorithm automatically updates the value of $b_{u,v}^{dislike}$ and $c_{u,v}^{dislike}$ when the rating of u and v are simultaneously lower than $\bar{r}_j'$.

At last, the algorithm will update related factors of deviation between two users in different user sets which are updated in step 2. In order to obtain the increment of deviation between u and v, there are six cases listed. Detailed information can be seen in (23) to (28).

$$\begin{aligned} &step\ 3, case\ 1(u, v \in CU_j^{like}): \\ &\begin{cases} b_{u,v}'^{like} = b_{u,v}^{like} - \left(r_{u,j} - r_{v,j}\right) \\ b_{v,u}'^{like} = -b_{u,v}'^{like} \\ c_{u,v}'^{like} = c_{u,v}^{like} - 1, c_{v,u}'^{like} = c_{u,v}'^{like} \\ b_{u,v}'^{dislike} = b_{u,v}^{dislike} + r_{u,j} - r_{v,j} \\ b_{v,u}'^{dislike} = -b_{u,v}'^{dislike} \\ c_{u,v}'^{dislike} = c_{u,v}^{dislike} + 1, c_{v,u}'^{dislike} = c_{u,v}'^{dislike} \end{cases} \end{aligned} \tag{23}$$

According to (23), the first case is when $u, v \in CU_j^{like}$, denoting u and v to become pessimistic users from optimistic users simultaneously after the emergence of new rating. Thus, there are two facts. (1) u and v used to be a member of U_j^{like}. (2) u and v become a member of $U_j^{dislike}$ later. Therefore, the related factors are $b_{u,v}^{like}$, $c_{u,v}^{like}$, $b_{u,v}^{dislike}$ and $c_{u,v}^{dislike}$.

$$\begin{aligned} &step\ 3, case\ 2(u \in CU_j^{like},\ v \in NCU_j^{like}): \\ &\begin{cases} b_{u,v}'^{like} = b_{u,v}^{like} - \left(r_{u,j} - r_{v,j}\right) \\ b_{v,u}'^{like} = -b_{u,v}'^{like} \\ c_{u,v}'^{like} = c_{u,v}^{like} - 1, c_{v,u}'^{like} = c_{u,v}'^{like} \end{cases} \end{aligned} \tag{24}$$

The second case is when $u \in CU_j^{like}$, $v \in NCU_j^{like}$, which means that u turns into a pessimistic user while v is a optimistic user. Thus, $b_{u,v}^{like}$ and $c_{u,v}^{like}$ are updated based on such change.

$$\begin{aligned} &step\ 3, case\ 3(u \in CU_j^{like},\ v \in NCU_j^{dislike}): \\ &\begin{cases} b_{u,v}'^{dislike} = b_{u,v}^{dislike} + r_{u,j} - r_{v,j} \\ b_{v,u}'^{dislike} = -b_{u,v}'^{dislike} \\ c_{u,v}'^{dislike} = c_{u,v}^{dislike} + 1 \\ c_{v,u}'^{dislike} = c_{u,v}'^{dislike} \end{cases} \end{aligned} \tag{25}$$

As in (25), u, v share a deviation in $U_j^{dislike}$ on the premise that $u \in CU_j^{like}$, $v \in NCU_j^{dislike}$, and the related factors, i.e., $b_{u,v}^{dislike}$ and $c_{u,v}^{dislike}$ are subsequently updated.

When $u, v \in CU_j^{dislike}$, the situation is just opposite to that of case 1. Therefore, the increment rule is opposite accordingly as given in (26):

$$\begin{aligned} &step\ 3, case\ 4(u, v \in CU_j^{dislike}): \\ &\begin{cases} b_{u,v}'^{dislike} = b_{u,v}^{dislike} - \left(r_{u,j} - r_{v,j}\right) \\ b_{v,u}'^{dislike} = -b_{u,v}'^{dislike} \\ c_{u,v}'^{dislike} = c_{u,v}^{dislike} - 1, c_{v,u}'^{dislike} = c_{u,v}'^{dislike} \\ b_{u,v}'^{like} = b_{u,v}^{like} + r_{u,j} - r_{v,j} \\ b_{v,u}'^{like} = -b_{u,v}'^{like} \\ c_{u,v}'^{like} = c_{u,v}^{like} + 1, c_{v,u}'^{like} = c_{u,v}'^{like} \end{cases} \end{aligned} \tag{26}$$

Based on case 2 and case 3, we can easily find update principles of case 5 and case 6 are equivalent to them separately, since we deal with the changed element in $U_j^{dislike}$ while case 2 and case 3

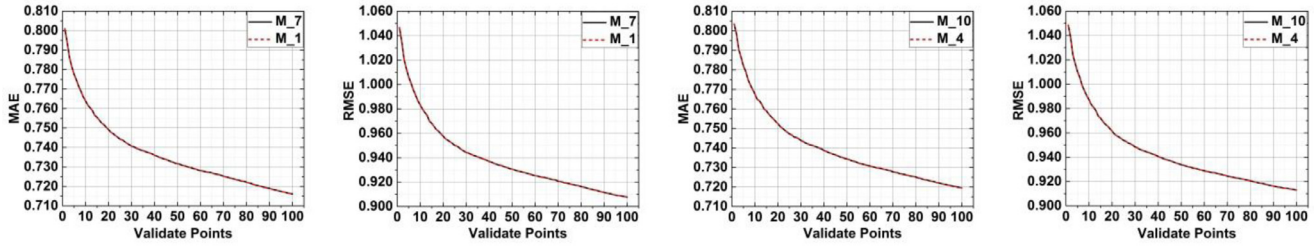

Fig. 1. MAE and RMSE comparison between the static and incremental Slope-one Scheme on ML1M: (a) MAE and (b) RMSE comparison on users' deviation, (c) MAE and (d) RMSE comparison on items' deviation.

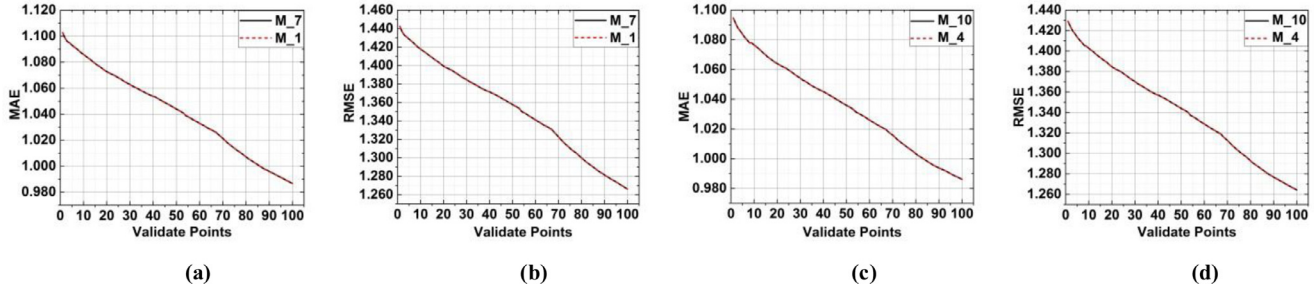

Fig. 2. MAE and RMSE comparison between the static and incremental Slope-one Scheme on EM2.8: (a) MAE and (b) RMSE comparison on users' deviation, (c) MAE and (d) RMSE comparison on items' deviation.

are mainly related to U_j^{like}. Thus, the corresponding update rules are given in (27) and (28).

$$step\ 3, case\ 5 (u \in CU_j^{dislike}, v \in NCU_j^{dislike}):$$
$$\begin{cases} b'^{dislike}_{u,v} = b^{dislike}_{u,v} - \left(r_{u,j} - r_{v,j}\right) \\ b'^{dislike}_{v,u} = -b'^{dislike}_{u,v} \\ c'^{dislike}_{u,v} = c^{dislike}_{u,v} - 1 \\ c'^{dislike}_{v,u} = c'^{dislike}_{u,v} \end{cases} \tag{27}$$

$$step\ 3, case\ 6 (u \in CU_j^{dislike}, v \in NCU_j^{like}):$$
$$\begin{cases} b'^{like}_{u,v} = b^{like}_{u,v} + r_{u,j} - r_{v,j} \\ b'^{like}_{v,u} = -b'^{like}_{u,v} \\ c'^{like}_{u,v} = c^{like}_{u,v} + 1 \\ c'^{like}_{v,u} = c'^{like}_{u,v} \end{cases} \tag{28}$$

Given a varying rating, we also consider the increment process in three steps, which is similar to that of new rating. The first step is to compute $\bar{r}'_j$, and divide user u into the right set, which are summarized in (29) and (30). The second step is to deal with existing users in U_j^{like} and $U_j^{dislike}$ as well as updating relevant cached user set mentioned above. The last step is to update the deviation factors as the same procedure of new rating. Detailed steps can be seen in Table 7.

B. With *changed rating* $r'_{u,j}$

$$step\ 1:$$
$$\bar{r}'_j = \bar{r}_j + \frac{r'_{u,j} - r_{u,j}}{n_j}$$
$$step\ 1,\ case\ 1, branch\ a (r_{u,j} > \bar{r}_j\ and\ r'_{u,j} > \bar{r}'_j):$$
$$NCU'^{like}_j = NCU^{like}_j + \{u\} \tag{29}$$
$$step\ 1,\ case\ 1, branch\ b (r_{u,j} > \bar{r}_j\ and\ r'_{u,j} < \bar{r}'_j):$$
$$\begin{cases} U'^{like}_j = U^{like}_j - \{u\} \\ U'^{dislike}_j = U^{dislike}_j + \{u\} \\ CU'^{like}_j = CU^{like}_j + \{u\} \end{cases}$$

$$step\ 1,\ case\ 2, branch\ a (r_{u,j} < \bar{r}_j\ and\ r'_{u,j} > \bar{r}'_j):$$
$$\begin{cases} U'^{like}_j = U^{like}_j + \{u\} \\ U'^{dislike}_j = U^{dislike}_j - \{u\} \\ CU'^{dislike}_j = CU^{dislike}_j + \{u\} \end{cases} \tag{30}$$
$$step\ 1,\ case\ 2, branch\ b (r_{u,j} < \bar{r}_j\ and\ r'_{u,j} < \bar{r}'_j):$$
$$NCU'^{dislike}_j = NCU^{dislike}_j + \{u\}$$

In (29) and (30), we consider four conditions in the first step: (1) user u is initially in U_j^{like}, and the rating is still higher than the updated $\bar{r}'_j$ afterwards. (2) u is initially in U_j^{like}, but the rating is lower than $\bar{r}'_j$ afterwards. (3) u is initially in $U_j^{dislike}$, and the rating is still lower than the updated $\bar{r}'_j$ afterwards. (4) u is initially in $U_j^{dislike}$, but the rating is lower than $\bar{r}'_j$ afterwards.

In (29), we insert u into NCU_j^{like} when $r_{u,j} > \bar{r}_j$, $r'_{u,j} > \bar{r}'_j$, denoting that u is still treated as optimistic user. Meanwhile, we update U_j^{like}, $U_j^{dislike}$ and CU_j^{like} when $r_{u,j} > \bar{r}_j$, $r'_{u,j} < \bar{r}'_j$, since u is no longer to be a optimistic user like before.

In (30), u is removed from $U_j^{dislike}$ and added to U_j^{like} and $CU_j^{dislike}$ when $r_{u,j} < \bar{r}_j$, $r'_{u,j} > \bar{r}'_j$, denoting that u is no longer be a pessimistic user. Meanwhile, we update $NCU_j^{dislike}$ when $r_{u,j} < \bar{r}_j$, $r'_{u,j} < \bar{r}'_j$, since u can still be a pessimistic user.

The second step is to deal with the user in U_j^{like} or $U_j^{dislike}$ as we talked above. Likewise, there are three cases considered as follows:

$$step\ 2 (v \in U_j),\ case\ 1:$$
$$v \notin U_j^{like} \cap v \notin U_j^{dislike}: no\ update \tag{31}$$

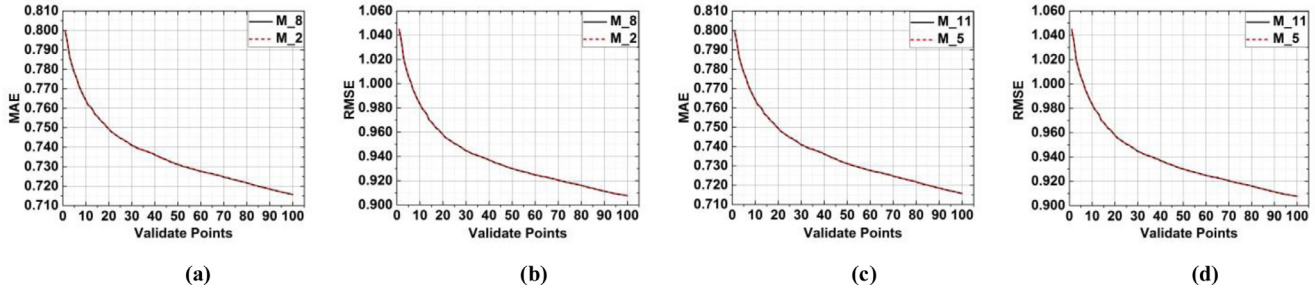

Fig. 3. MAE and RMSE comparison between the static and incremental Weighted Slope-one Scheme on ML1M: (a) MAE and (b) RMSE comparison on users' deviation, (c) MAE and (d) RMSE comparison on items' deviation.

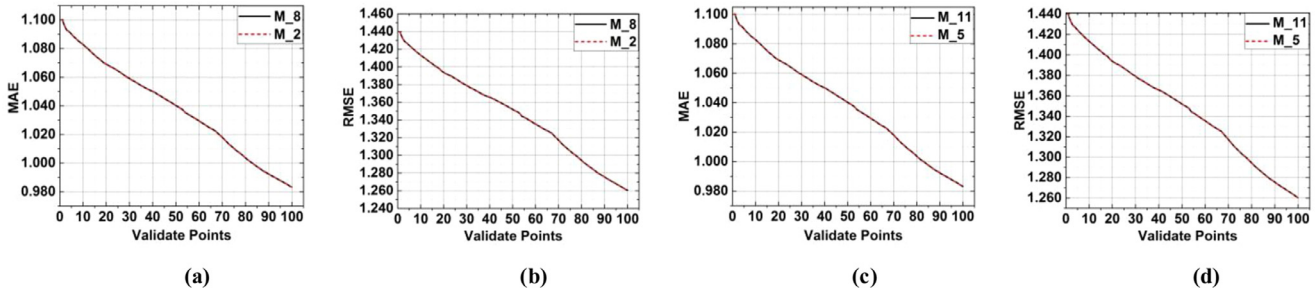

Fig. 4. MAE and RMSE comparison between the static and incremental Weighted Slope-one Scheme on EM2.8M: (a) MAE and (b) RMSE comparison on users' deviation, (c) MAE and (d) RMSE comparison on items' deviation.

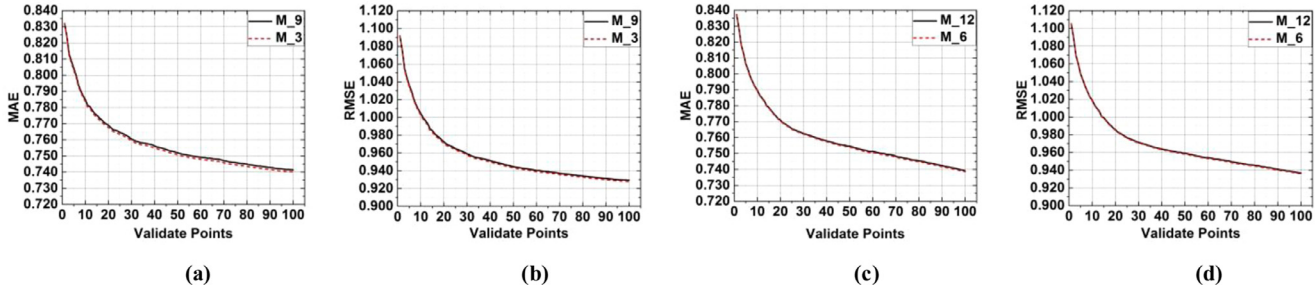

Fig. 5. MAE and RMSE comparison between the static and incremental Bi-Polar Slope-one Scheme on ML1M: (a) MAE and (b) RMSE comparison on users' deviation, (c) MAE and (d) RMSE comparison on items' deviation.

$$
\begin{aligned}
&step\ 2(v \in U_j),\ case\ 2, branch\ a(v \in U_j^{dislike}\ and\ r_{v,j} > \bar{r}'_j):\\
&\begin{cases} U_j'^{like} = U_j^{like} + \{v\} \\ U_j'^{dislike} = U_j^{dislike} - \{v\} \\ CU_j'^{dislike} = CU_j^{dislike} + \{v\} \end{cases}\\
&step\ 2(v \in U_j),\ case\ 2, branch\ b(v \in U_j^{dislike}\ and\ r_{v,j} < \bar{r}'_j):\\
&NCU_j'^{dislike} = NCU_j^{dislike} + \{v\}
\end{aligned} \tag{32}
$$

$$
\begin{aligned}
&step\ 2(v \in U_j),\ case\ 3, branch\ a(v \in U_j^{like}\ and\ r_{v,j} > \bar{r}'_j):\\
&NCU_j'^{like} = NCU_j^{like} + \{v\}\\
&step\ 2(v \in U_j),\ case\ 3, branch\ b(v \in U_j^{like}\ and\ r_{v,j} < \bar{r}'_j):\\
&\begin{cases} U_j'^{like} = U_j^{like} - \{v\} \\ U_j'^{dislike} = U_j^{dislike} + \{v\} \\ CU_j'^{like} = CU_j^{like} + \{v\} \end{cases}
\end{aligned} \tag{33}
$$

$$
step\ 3, the\ same\ as\ new\ rating\ r_{v,j}\ step\ 3. \tag{34}
$$

The first case is when v does not belong to U_j^{like}, $U_j^{dislike}$. By comparison, the processes of case 2 and case3 are the same as the first one. The reason is that u and v are already existed in the user set. Thus, the main idea is to insert them into the four cached user set. Regarding to the last step, it is also the same as the process of new rating.

4. Experiments and results

Dataset. The experiments are conducted on two datasets including Movielens (hereafter referred to as the ML1M) and EachMovie (hereafter referred to as the EM2.8 M) which are collected by Camra2011 Competition Group and Compaq Research respectively. ML1M has a total of one million ratings from approximate 6400 users on 3900 movies, while EM2.8 M contains nearly 2800 thousand ratings from 73000 users on 1682 movies. The ratings of the datasets range from 0 to 5 for ML1M. The ratings of the datasets range from 0 to 1 for EM2.8 M, which is mapped into the interval of [0, 5] for sake of keeping consistent with the ML1M ratings. In this paper, we utilize the first three fields of every rating data to conduct experiments, i.e. user id, item id and rating. Initially, datasets are arranged in an ascending sort order on the basis of user id or item id. After that, these datasets are randomly distributed into test datasets and training datasets with equal probability of 0.5 while training datasets consists of 80% increment dataset and 20% base dataset. Since the original datasets has been ordered, the extracted datasets for experiments have a balance of

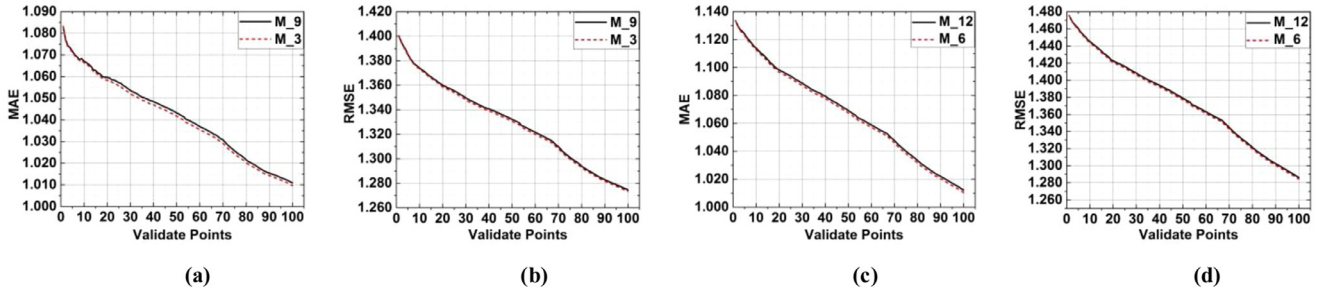

Fig. 6. MAE and RMSE comparison between the static and incremental Bi-Polar Slope-one Scheme on EM2.8M: (a) MAE and (b) RMSE comparison on users' deviation, (c) MAE and (d) RMSE comparison on items' deviation.

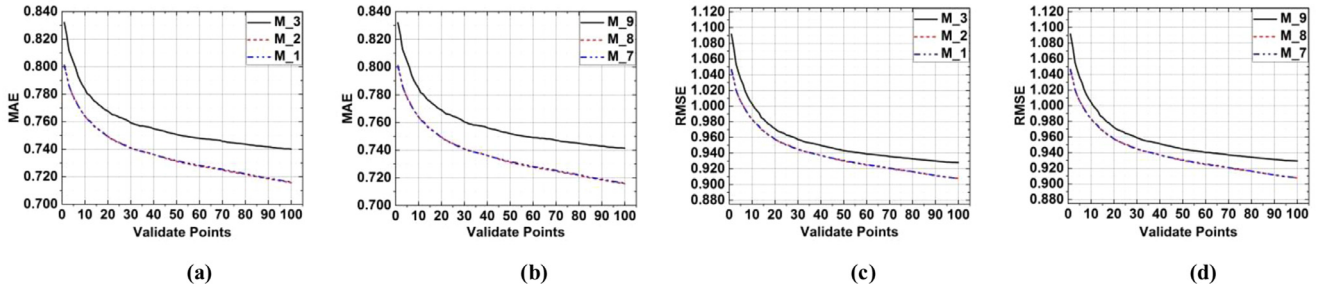

Fig. 7. MAE and RMSE comparison between the static and incremental Slope-one Scheme based on users' deviation on ML1M: (a) MAE comparison of three static schemes, (b) MAE comparison of three incremental schemes, (c) RMSE comparison of three static schemes, and (d) RMSE comparison of three incremental schemes.

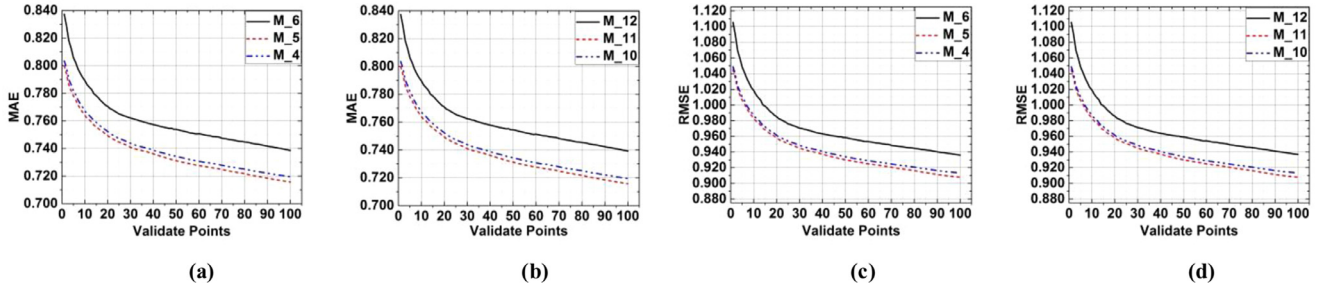

Fig. 8. MAE and RMSE comparison between the static and incremental Slope-one Scheme based on items' deviation on ML1M: (a) MAE comparison of three static schemes, (b) MAE comparison of three incremental schemes, (c) RMSE comparison of three static schemes, and (d) RMSE comparison of three incremental schemes.

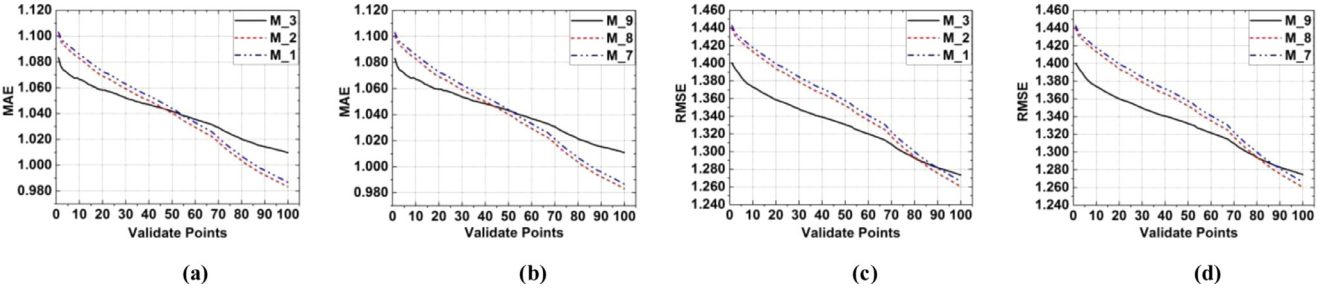

Fig. 9. MAE and RMSE comparison between the static and incremental Slope-one Scheme based on users' deviation on EM2.8M: (a) MAE comparison of three static schemes, (b) MAE comparison of three incremental schemes, (c) RMSE comparison of three static schemes, and (d) RMSE comparison of three incremental schemes.

user's rating numbers and rating users for every item with original datasets. In the process of displaying the experiment results, we extract 100 incremental validate points in contrast with 100 points based on the static algorithm, eventually succeed in accomplishing ten experiments.

Evaluation metrics. In order to validate the accuracy and efficiency of the proposed increment Slope-one Scheme, we adopt the root mean squared error (RMSE) and mean absolute error(MAE) as the recommendation accuracy metrics, both of which are widely used for evaluating the statistical accuracy of collaborative filtering models [5–16]. Let R_{test} denote the validation data set. Then we have the following equation:

$$RMSE = \sqrt{\frac{\sum_{r_{u,i}\in R_{test}} \left(r_{u,i} - \hat{r}_{u,i}\right)^2}{|R_{test}|}}, MAE = \frac{\sum_{r_{u,i}\in R_{test}} \left|r_{u,i} - \hat{r}_{u,i}\right|}{|R_{test}|} \tag{35}$$

where u refers to user, i means item, and r is rating. Thus, $r_{u,i}$ is the original rating of i by user u; $\hat{r}_{ui}$ is the predict rating of i by user u; $|R_{test}|$ is the number of elements in R_{test}.

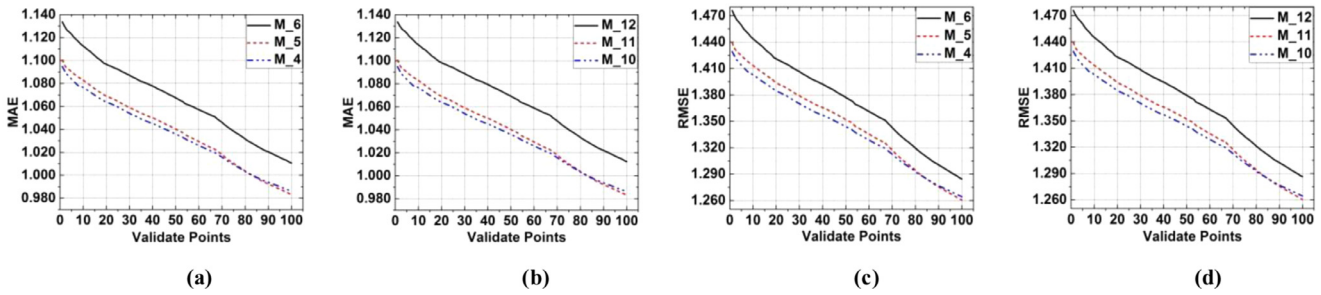

Fig. 10. MAE and RMSE comparison between the static and incremental Slope-one Scheme based on items' deviation on EM2.8M: (a) MAE comparison of three static schemes, (b) MAE comparison of three incremental schemes, (c) RMSE comparison of three static schemes, and (d) RMSE comparison of three incremental schemes.

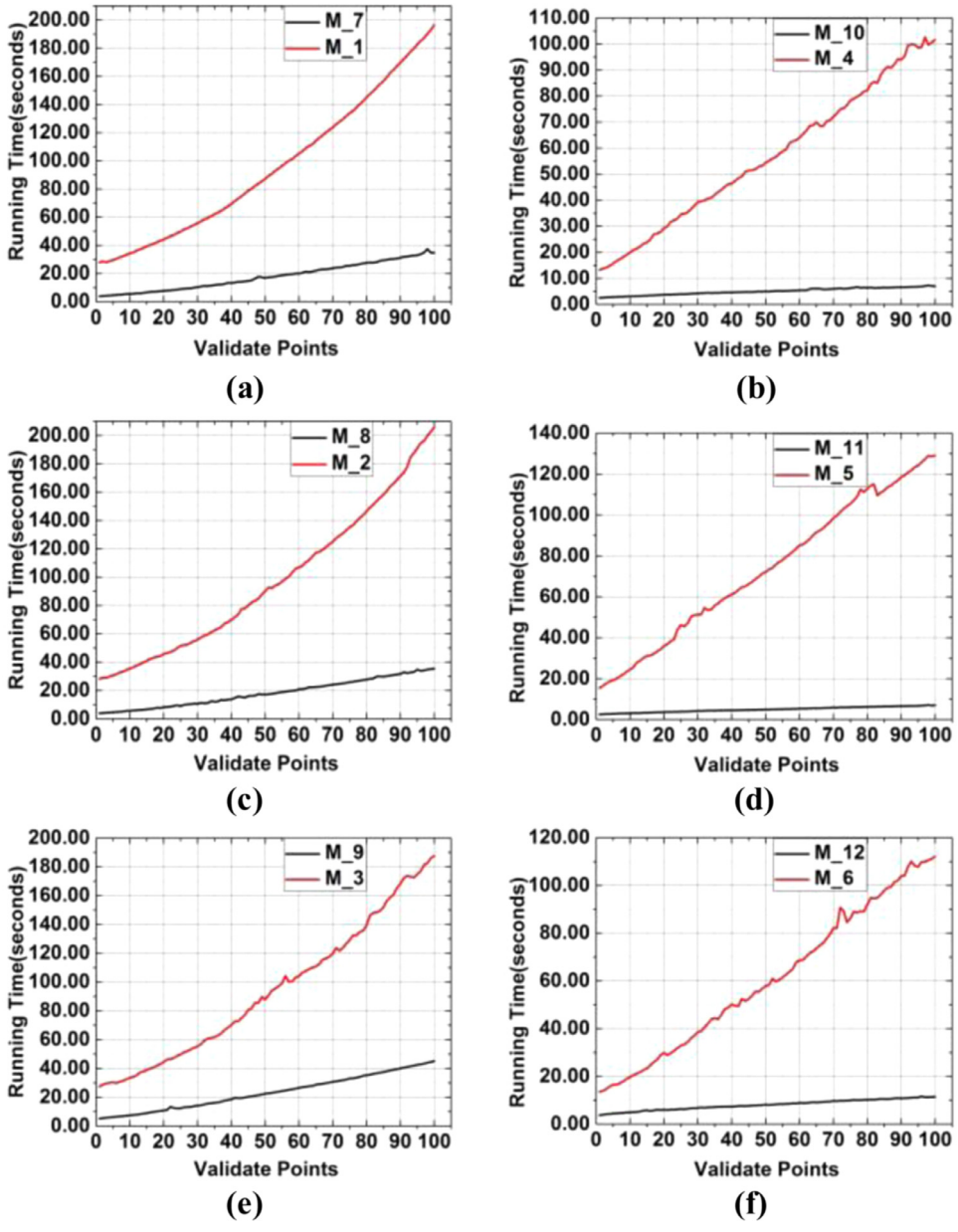

Fig. 11. Efficiency comparison between incremental and static Slope-one models on ML1M.

Accuracy comparison between incremental and static slop-one models. In our ten experiments, ML1M and EM2.8 M are assigned to five experiments separately. The former three experiments are about the comparison between the Basic Slope-one Scheme and Incremental Slope-one Scheme on these two datasets. The last two experiments are related to the comparison of three different recommenders from user deviation and item deviation for the Basic Slope-one Scheme and Incremental Slope-one Scheme. Symbols used in result figures are listed in Table 8.

As we can see, Figs. 1–6 show the results of RMSE and MAE from user deviation aspect and item deviation aspect separately. The line direction and the value of the validation points of the Basic Slope-one Scheme are almost the same as those of the Incremental Slope-one. Meanwhile, for Incremental Bi-polar Slope-one algorithm, the value of each validation point has slight error which is less than 0.2% of the value of Basic Bi-polar Slope-one. Thus, the above phenomenon can be accepted to prove the correctness of the algorithm. Besides, there is a steadily declining tendency for RMSE and MAE accompanied by the enlarging of the dataset, which can indicates the increasing performance of algorithm and validates the proposed update algorithm. The similar situation can be found from the experiment results on EM2.8 M

Figs. 7 and 8 show the comparison among three kinds of Slope-one from user deviation and item deviation separately on ML1M while Figs. 9 and 10 responding to EM2.8 M We can find that the line directions of three schemes are almost the same. As the enlarging of the dataset, there is a steadily declining tendency for RMSE and MAE while the former two schemes declined steadily and they are superior to the third one. In addition, the second scheme has a slightly better performance than the first one on ML1M in Fig. 8. On the other hand, in Fig. 10, the first scheme has a slightly better performance than the second one while the advantage fade away on specific points. Additionally, the third scheme is superior to the former two schemes on EM2.8 M based on user deviation initially, but this phenomenon can be ignored because the advantage fade away on specific points in Fig. 9.

From the above comparison, we can summarize that the proposed incremental recommenders can provide efficient predictions as accurate as original static ones, indicating that the incremental update rules are correct.

Efficiency comparison between incremental and static slop-one models. Fig. 11 depicts the time cost of incremental and static Slope-one models on ML1M. Note that similar situations can be found on EM2.8 M As shown in Fig. 11, we find that the computational efficiency of incremental Slope-one models is much lower than that of the corresponding static models. This phenomenon is reasonable: incremental models only update the parameters related to the data increments; static models have to retrain the whole model, and rebuild all parameters to reflect the data variations.

Summary. To summarize, the proposed incremental Slope-one models are able to update the parameters correctly for reflecting the variations in the data increments, while their time cost is much lower than that of corresponding static models. Hence, the incremental Slope-one models are more practical for real applications than static Slope-one models.

5. Conclusions

The recommender with incremental update scheme, as one of effective way to solve the rapid data-variation problem, becomes more and more popular in recommender systems. Different with existing works put much energies on designing recommender with high accuracy on static data, we focus on the recommendation issue on the ever-changing data. In this paper, we propose three incremental Slope-one recommenders, which involves both user-based and item-based deviation aspects. In order to illustrate the prediction accuracy, we conduct five experiments on ML1M and EM2.8 datasets respectively by presenting comparative results of Static recommenders and Incremental recommenders. The results demonstrate that our three incremental algorithms have an outstanding prediction accuracy and updatability to the change of ratings.

Future efforts regarding this work will be mainly put on integrating the principles of collection intelligence framework, like the particle swarm optimization [45–49], into the incremental recommenders for further enhancing its performance.

References

[1] J. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative-filtering, in: Proceedings of the Fourteenth International Conference on Uncertainty in Artificial Intelligence, San Francisco, California, USA, 1998, pp. 43–52.
[2] B. Sarwar, G. Karypis, J. Konstan, J. Reidl, Item based collaborative-filtering recommendation algorithms, in: Proceedings of the of the Tenth International Conference on World Wide Web, PRC, Hong Kong, 2001, pp. 285–295.
[3] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. 17 (2005) 734–749 Jun.
[4] D.-S. Li, Q. Lv, L. Shang, N. Gu, Efficient privacy-preserving content recommendation for online social communities, Neurocomputing 219 (2017) 440–454.
[5] J. Davidson, B. Liebald, J. Liu, P. Nandy, T.V. Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, D. Sampath, The YouTube video recommendation system, in: Proceedings of the Fourth ACM Conference on Recommender Systems, Barcelona, Spain, 2010, pp. 293–296.
[6] G. Linden, B. Smith, J. York, Amazon.com recommendations: item-to-item collaborative filtering, IEEE Internet Comput. 7 (1) (2003) 76–80.
[7] A. Van den Oord, S. Dieleman, B. Schrauwen, Deep content-based music recommendation, Advances in Neural Information Processing Systems, Lake Tahoe, Nevada, USA, 2013, pp. 2643–2651.
[8] G. Linden, B. Smith, J. York, Amazon.com recommendations: item-to-item collaborative filtering, IEEE Internet Comput. 7 (1) (2003) 76–80.
[9] M-S. Shang, Z. Zhang, T. Zhou, Y-C. Zhang, Collaborative filtering with diffusion-based similarity on tripartite graphs, Phys. A Stat. Mech. Appl. 389 (6) (2010) 1259–1264.
[10] T. Zhou, Z. Kuscsik, J. Liu, M. Medo, J.R. Wakeling, Y-C. Zhang, Solving the apparent diversity-accuracy dilemma of recommender systems, Proc. Natl. Acad. Sci. 107 (10) (2010) 4511–4515.
[11] R. Salakhutdinov, A. Mnih, Probabilistic matrix-factorization, Adv. Neural Inf. Process. Syst. 20 (2008) 1257–1264.
[12] Y.-F. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets, in: Proceedings of the Eighth IEEE International Conference on Data Mining, Pisa, 2008, pp. 263–272.
[13] Y. Koren, R. Bell, C. Volinsky, Matrix-factorization techniques for recommender systems, IEEE Comput. 42 (8) (2009) 30–37.
[14] G. Takács, I. Pilászy, B. Németh, D. Tikk, Scalable collaborative filtering approaches for large recommender systems, J. Mach. Learn. Res. 10 (2009) 623–656.
[15] Z. Zhang, K. Zhao, H. Zha, Inducible regularization for low-rank matrix factorizations for collaborative filtering, Neurocomputing 97 (0) (2012) 52–62.
[16] X. Luo, M.-C. Zhou, Y.-N. Xia, Q.-S. Zhu, An efficient non-negative matrix-factorization-based approach to collaborative-filtering for recommender systems, IEEE Trans. Ind. Inf. 10 (2) (2014) 1273–1284.
[17] X. Luo, M.-C. Zhou, S. Li, Z.-H. You, Y.-N. Xia, Q.-S. Zhu, H. Leung, An efficient second-order approach to factorizing sparse matrices in recommender systems, IEEE Trans. Ind. Inf. 11 (4) (2015) 946–956.
[18] X. Luo, M.-C. Zhou, S. Li, Z.-H. You, Y.-N. Xia, Q.-S. Zhu, A non-negative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method, IEEE Trans. Neural Netw. Learn. Syst. 27 (3) (2016) 524–537.
[19] X. Luo, M.-C. Zhou, Y.-N. Xia, Q.-S. Zhu, Generating highly accurate predictions for missing QoS-data via aggregating non-negative latent factor models, IEEE Trans. Neural Netw. Learn. Syst. 27 (3) (2016) 579–592.
[20] S. Li, Z.-H. You, H. Guo, X. Luo, Z.-Q. Zhao, Inverse-free extreme learning machine with optimal information updating, IEEE Trans. Cybern. 46 (5) (2016) 1229–1241.
[21] W.-K. Pan, L. Chen, Group Bayesian personalized ranking with rich interactions for one-class collaborative filtering, Neurocomputing 207 (2016) 501–510.
[22] D. Lemire, A. Maclachlan, Slope one predictors for online rating-based collaborative filtering, in: Proceedings of the SIAM International Conference on Data Mining, 2007, pp. 471–480.
[23] M. Gao, Z.-F. Wu, F. Jiang, Userrank for item-based collaborative filtering recommendation, Inf. Process. Lett. 111 (9) (2011) 440–446.
[24] Y. Zheng, X. Xie, Learning travel recommendations from user-generated GPS traces, ACM Trans. Intell. Syst. Technol. 2 (1) (2011) Article 29.
[25] X.-Y. Zhao, Z.-D. Niu, W. Chen, Interest before liking: two-step recommendation approaches, Knowl. Based Syst. 48 (2013) 46–56.
[26] D. Menezes, A. Lacerda, L. Silva, A. Veloso, N. Ziviani, Weighted slope one predictors revisited, in: Proceedings of the Twenty-Second International Conference on World Wide Web, Rio de Janeiro, Brazil, 2013.
[27] F. Xie, Z. Chen, J.-X. Shang, G.C. Fox, Grey Forecast model for accurate recommendation in presence of data sparsity and correlation, Knowl. Based Syst. 69 (2014) 179–190.
[28] K. Lin, J. Liu, M. Qiu, K. Guo, Location-based personalized mobile search, in: Proceedings of the Ninth International Conference on Computer Science and Education, Vancouver, British Columbia, Canada, 2014, pp. 536–539.

[29] Q. Tang, B. Pejó, H. Wang, Protect both integrity and confidentiality in outsourcing collaborative filtering computations, in: Proceedings of the Ninth IEEE International Conference on Cloud Computing, New York City, NY, USA, 2016, pp. 941–946.
[30] M. Zheng, F. Min, H.-R. Zhang, W.-B. Chen, Fast recommendations with the M-Distance, IEEE Access 4 (2016) 1464–1468.
[31] X. Cheng, D. He, M. Fang, A hybrid collaborative filtering recommendation algorithm, in: Proceedings of the International Conference on Intelligent Information Processing, Wuhan, China, 2016.
[32] B. Jin, C. Che, K. Yu, Y. Qu, L. Guo, C. Yao, R. Yu, Minimizing legal exposure of high-tech companies through collaborative filtering methods, in: Proceedings of the Twenty-Second ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, California, USA, 2016, pp. 127–136.
[33] H. You, H. Li, Y. Wang, Q. Zhao, An improved collaborative filtering recommendation algorithm combining item clustering and slope one scheme, in: Proceedings of the International Multi Conference of Engineers and Computer Scientists, Hongkong, 2015, pp. 18–20.
[34] J. Zhao, J. Ma, An improved slope one algorithm based on tag frequency, in: Proceedings of the Third International Conference on Computer Science and Network Technology (ICCSNT), Dalian, China, 2013, pp. 369–372.
[35] T.Q. Jiang, W. Lu, Improved slope one algorithm based on time weight, Appl. Mech. Mater. 347 (2013) 2365–2368.
[36] J. Li, L. Sun, J. Wang, A slope one collaborative filtering recommendation algorithm using uncertain neighbors optimizing, in: Proceedings of the International Conference on Web-Age Information Management, Berlin Heidelberg, 2011, pp. 160–166.
[37] Z Zhang, X. Tang, D. Chen, Applying user-favorite-item-based similarity into slope one scheme for collaborative filtering, in: Proceedings of the IEEE World Congress on Computing and Communication Technologies (WCCCT), Trichirappalli, India, 2014, pp. 5–7.
[38] Q. Wang, J. Zhang, X. Shi, M. Shang, User heterogeneity and individualized recommender, Chin. Phys. Lett. 34 (6) (2017) 68902.
[39] M. Papagelis, R. Ioannis, D. Plexousakis, E. Theoharopoulos, Incremental collaborative filtering for highly-scalable recommendation algorithms, in: Proceedings of the Fifteenth International Symposium on Methodologies for Intelligent Systems, New York, NY, United States, 2005, pp. 553–561.
[40] M. Khoshneshin, W. Street, Incremental collaborative filtering via evolutionary co-clustering, in: Proceedings of the Fourth ACM Conference on Recommender Systems, Barcelona, Spain, 2010, pp. 325–328.
[41] X. Luo, Y.-N. Xia, Q.-S. Zhu, Incremental collaborative filtering recommender based on regularized matrix factorization, Knowl. Based Syst. 27 (2012) 271–280.
[42] X. Luo, Y.-N. Xia, Q.-S. Zhu, Y. Li, Boosting the K-nearest-neighborhood based incremental collaborative filtering, Knowl. Based Syst. 53 (2013) 90–99.
[43] X. Luo, M.-C. Zhou, Y.-N. Xia, Q.-S. Zhu, An incremental-and-static-combined scheme for matrix-factorization-based collaborative filtering, IEEE Trans. Autom. Sci. Eng. 13 (1) (2016) 333–343.
[44] X. Luo, M.-C. Zhou, Z.-D. Wang, Y.-N. Xia, Q.-S. Zhu, An effective QoS estimating scheme via alternating direction method-based matrix factorization, IEEE Trans. Serv. Comput., doi:10.1109/TSC.2016.2597829.
[45] N. Zeng, Z. Wang, Y. Li, M. Du, X. Liu, A hybrid EKF and switching PSO algorithm for joint state and parameter estimation of lateral flow immunoassay models, IEEE/ACM Trans. Comput. Biol. Bioinf. 9 (2) (2012) 321–329 (ISSN 1545-5963).
[46] N. Zeng, Z. Wang, Y. Li, M. Du, X. Liu, Inference of nonlinear state-space models for sandwich-type lateral flow immunoassay using extended Kalman filtering, IEEE Trans. Biomed. Eng. 58 (7) (2011) 1959–1966 (ISSN 0018-9294).
[47] N. Zeng, Z. Wang, Y. Li, M. Du, X. Liu, Identification of nonlinear lateral flow immunoassay state-space models via particle filter approach, IEEE Trans. Nanotechnol. 11 (2) (2012) 321–327 (ISSN 1536-125X).
[48] N. Zeng, Z. Wang, Y. Li, M. Du, J. Cao, X. Liu, Time series modeling of nano-gold immunochromatographic assay via expectation maximization algorithm, IEEE Trans. Biomed. Eng. 60 (12) (2013) 3418–3424 (ISSN 0018-9294).
[49] N. Zeng, Y.S. Hung, Y. Li, M. Du, A novel switching local evolutionary PSO for quantitative analysis of lateral flow immunoassay, Expert Syst. Appl. 41 (4) (2014) 1708–1715 (ISSN 0957-4174).

Qing-Xian Wang received her Ph.D. degree in computer science from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2006. She joined the School of Computer Science and Engineering, UESTC, Chengdu, China, in 2002, then worked in the School of Information and Software Engineering in 2015, and is now an associate professor of Information and Software Engineering at UESTC. Her research interests are in recommender systems, artificial intelligence, automatic reasoning, and information security.

Xin Luo received the B.S. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2005, and the Ph.D. degree in computer science from Beihang University, Beijing, China, in 2011. He joined the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China in 2016, and is currently a Professor of computer science and engineering. His research interests include big data analysis and intelligent control. He has published 70+ papers (including 17 IEEE TRANSACTIONS papers) in his related areas.

Yan Li received the B.S. degree in Computer Science and Technology from Dalian Maritime University of China, Dalian, China, in 2015. She is currently a graduate student in computer technology at Chongqing University.

Xiaoyu Shi received the B.S. degree in computer science from the PLA Information Engineering University, Zhengzhou, China, in 2007, and the Ph.D. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2015. He joined the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China in 2015. His research interests include recommender system, cloud computing, green computing and big data applications.

Liang Gu received the B.S. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2005, and the Ph.D. degree in computer science from Peking University, Beijing, China, in 2011. He worked as postdoctoral researcher at Yale University from 2011–2013 and then was promoted as the associate research scientist from 2013–2015. He is currently the Chief Technology Officer of R&D at Sangfor Technology Inc. and adjunct research professor at Jinan University. His research interests include network security and cloud computing.

Mingsheng Shang received his Ph.D. degree in Computer Science from University of Electronic Science and Technology of China in Chengdu (UESTC), China in 2007. He joined the School of Computer Science and Engineering, UESTC, Chengdu, China, in 2002, and was a professor with UESTC. He is currently a professor at the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China, and served as deputy director of Institute of Electronic Information Technology, director of Big Data Mining Center. His research interests are in complex network analysis and big data applications.