Knowledge-Based Systems 53 (2013) 90–99

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Boosting the K-Nearest-Neighborhood based incremental collaborative filtering

Xin Luo [a,c,*], Yunni Xia [a], Qingsheng Zhu [a], Yi Li [b]

[a] College of Computer Science, Chongqing University, Chongqing 400044, China
[b] People's Liberation Army Troop 95430, Chengdu, Sichuan 610081, China
[c] Chongqing Key Laboratory of Software Theory & Technology, Chongqing 400044, China

ARTICLE INFO

Article history:
Received 20 November 2012
Received in revised form 6 August 2013
Accepted 13 August 2013
Available online 21 August 2013

Keywords:
Recommender system
Incremental recommendation
Collaborative filtering
Rating similarity
K-Nearest-Neighborhood

ABSTRACT

Recommender systems which can automatically match users with their potential favorites usually rely on Collaborative Filtering (CF). Since in real-world applications the data of historical user behavior are ever growing, it is important to study the incremental CF models which can adapt to this data explosion quickly and flexibly. The rating similarity based K-Nearest-Neighborhood (RS-KNN) is a classical but still popular approach to CF; therefore, to investigate the RS-KNN based incremental CF is significant. However, current incremental RS-KNN (I-KNN) models have the drawbacks of high storage complexity and relatively low prediction accuracy. In this work, we intend to boost the RS-KNN based incremental CF. We focus on two points which are respectively (a) reducing the storage complexity while maintaining the prediction accuracy by employing the generalized Dice coefficients, and (b) improving the prediction accuracy by integrating the similarity support and linear biases as well as implementing the corresponding incremental update. The efficiency of our strategies is supported by the positive results of the experiments conducted on two real datasets.

1. Introduction

Recommender systems can enable the convenience of automatic information matching in web applications; people can be connected with items they might be interested in with the help of such systems according to their historical behaviors. Since recommender systems can free users from endless searches for favorites and thus greatly improve the popularity and user satisfaction of the corresponding web providers, they are becoming more and more important with web data ever exploding [1–4]. In the meanwhile, with the rapid progress of E-learning, E-service and Social Networks, recommender systems, which can play the role of automatic information filter, are also becoming indispensable in these areas [5,6].

In general, a recommender system can be implemented through several ways, among which a very popular and successful branch is the Collaborative Filtering [2–6]. Inside a CF recommender the historical user behaviors (e.g., scores, clicks, purchase and rental records, etc.) on involved items (e.g., movies, books, news, commodities, etc.) are modeled into a user-item rating matrix, where high ratings usually represent for strong preferences and vice versa. Due to the huge amounts of users and items, this matrix is usually very sparse where most entries are unknown [2–6]. Therefore, recommendations can be generated via properly estimating the unknown entries [2–6] based on the given ones.

According to the recent progress in recommender systems, the Neighborhood Based Model (NBM) and the Latent Factor Model (LFM) are two widely-employed categories of CF models [4,15]. LFM based recommenders work by mapping both items and users into the same latent factor space, and then train the feature factors to fit the known ratings; the predictions for unknown ratings will depend on the inner products of the corresponding user-item feature-vector pairs and some other factors [7–15]. NBM based recommenders, on the other hand, will firstly model the neighborhoods of users/items; the predictions for unknown ratings depend on the existing ratings by the active users' neighbors/on the neighbors of the active items [16–23]. During the NetFlix prize, LFM based recommenders have proved to be highly accurate and scalable [10–15]. Nevertheless, NBM based recommenders are explainable and easy to implement [2–6,15–20]. Thus, these two kinds of models can be employed in different occasions depending on the detailed requirements.

However, most of the current CF recommenders address this problem of missing-rating estimation with a static precondition, which assumes that the rating matrix is fixed without incremental

* Corresponding author. Present address: College of Computer Science, Chongqing University, Chongqing 400044, China. Tel.: +86 13996169379.
E-mail addresses: luoxin21@cqu.edu.cn (X. Luo), xiayunni@yahoo.com.cn (Y. Xia), qszhu@cqu.edu.cn (Q. Zhu), li_yi608@163.com (Y. Li).

0950-7051/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2013.08.016

data. For real-world applications, this attitude is lack of practicability since user behaviors accumulate at every millisecond and the corresponding ratings are ever-growing. Once a recommender could make quick response to this data explosion, the corresponding user interest models can be structured more flexibly and specifically, and the user loyalty and satisfaction can be further improved. Nonetheless, with static recommenders the usual way to implement this desired adaptation is to retrain the whole model when rating variations accumulate over certain thresholds; this approach, however, will lead to high computational complexity along with inflexible reactions to the user behavior changes.

Naturally, the ideal way to cope with this rapid data variation is to enable recommenders being adaptive to the incremental ratings. Therefore, the incremental design in recommenders becomes very important. Since the CF based recommenders are highly efficient and popular, to implement incremental CF models is attractive; and there are several relevant pioneering works. Agarwal et al., propose the incremental recommender named FOBFM based on bilinear factors [24]. Sawar et al., propose the incremental SVD based recommender via the fold-in technique [25]. Takacs et al. [12], and Rendle et al. [26], suggest implementing the incremental update by fixing the current model and retraining the latent factors of new users/items on the arrival of corresponding feedbacks. Luo et al. [27], propose the incremental RMF model (hereafter referred to as the I-RMF model) by tracking the influence by the newly arriving ratings on the already trained model.

The aforementioned incremental recommenders are all LFM based recommenders. Nonetheless, as we mentioned above, the NBM based recommenders are also applicable to real-world systems due to the explainable nature and ease of implementation. Therefore, to investigate the NBM based incremental CF is significant. In this work we intend to investigate the incremental implementation on the classical NBM based CF recommender relying on the rating similarity (hereafter interchangeable with RS) based K-Nearest-Neighborhood (this model is hereafter referred to as the RS-KNN model). As reported by pioneering researchers, the RS-KNN model can be outperformed by global-optimization based NBM in accuracy in some circumstances [10–15]. However, the main parameters of the RS-KNN model, which are the rating similarities, are independent on each other. Therefore, the parallelization of the model training can be implemented easily, along with the possibility to carry out precise incremental updates. Thus, this model can be complementary to global-optimization based NBM under distributed/incremental conditions. Therefore, implementing the incremental RS-KNN model is still attractive and meaningful.

There are also some pioneering works focusing on building such models. Papagelis et al., propose the incremental user-oriented RS-KNN model [28]. Miranda et al., propose the incremental item-oriented RS-KNN model [29] based on the work in [28]. Since these two models are both based on implementing the incremental update of Pearson-correlation coefficient (hereafter interchangeable with PCC), we refer to them as the I-KNN model uniformly. The principle of I-KNN is to decompose the PCC coefficients into cached factors, and update them incrementally according to rating variations. The corresponding rating similarities are computed instantly when needed. The I-KNN model can effectively carry out incremental updates according to rating variations; however, it has two defects. The first is the requirement for huge memory. For implementing the incremental update of PCC, several factors (6 in total according to [28]) in correspondence to each rating similarity need to be cached. Please note that for NBM based recommenders, the number of the neighborhood coefficients is quadratically related to the number of users/items. Thus, with a huge user/item count the memory required for caching these factors will be considerable. The other drawback of I-KNN is the relatively low accuracy, since it generates predictions totally based on the similarity weighted average. According to the recent research on the RS-KNN model, more factors including the linear biases [15,20] (hereafter interchangeable with LB) and the similarity support [3,20,30] (hereafter interchangeable with SS) can be integrated into this model to obtain higher accuracy.

During the implementation of real-world recommender systems, we have found that the above-mentioned drawbacks of I-KNN gravely affect the practicability. On the other hand, if we can reduce the storage complexity while improve the prediction accuracy, the I-KNN model can become more feasible for engineering use. The storage complexity comes first, since in real applications the counts of users/items are numerous that we usually cannot afford the required caching memory. Since these memory are required to cache PCC components, we can free part of them by replacing PCC with other RS metrics with simple formula on the premise of causing no compromise in prediction accuracy. With regard to the accuracy issue, we plan to take SS and LB into consideration, explore the incremental updating rule of these factors and integrate them into the I-KNN model. To summarize, the main motivation of this work is to boost the performance of the I-KNN model to make it more practical in real-world applications. The main contributions include,

- We investigate the similarity measurement, and replace PCC with the generalized Dice coefficient (hereafter interchangeable with GDC). Through empirical validations, we find that this strategy can maintain the prediction accuracy while obviously decrease the storage complexity.
- We integrate the SS and LB into the incremental KNN based CF model to obtain higher and more stable prediction accuracy in incremental circumstances.
- We have conducted experiments on two real datasets for validating the performance of the proposed strategies.

The remainder of this paper is organized as follows. Section 2 discusses the base model and the related works; Section 3 states the methods of this work; the empirical validations are given in Section 4; finally, Section 5 concludes.

Please note that in the following we have used several acronyms. In the Appendix Section we have provided a simple table to explain these acronyms for avoiding confusion.

2. Warming up

2.1. Definitions and symbols

The CF recommenders always quantize the existed user behaviors into a user-item rating matrix, which can be defined as [4,15,27]:

Definition 1. The user-item rating matrix. Given the item set I and the user set U, the user-item rating matrix R is a $|U| \times |I|$ matrix where each element $r_{u,i}$ is proportional to user u's preference on item i.

Thus, the problem of CF can be regarded as the missing data estimation defined as [4,15,27]:

Definition 2. The problem of CF. Let R_K and R_U respectively denote the known and unknown entry sets in R; given $T \subseteq R_K$ as the training dataset; the problem of CF is how to construct an estimator $\widehat{R}$ which can achieve or approximate $\arg\min\left(\sum_{(u,i)\in R_U}|\hat{r}_{u,i} - r_{u,i}|\right)$.

With the RS-KNN model, the problem of CF can be addressed through (1) modeling the rating similarities between users/items, (2) picking up the K-Nearest-Neighbors of each user/item accord-

ing to the rating similarities, and then (3) predicting unknown ratings based on the known ratings by the active users' neighbors/ on the neighbors of the active item weighted by the corresponding rating similarities. Since in a real-world application the number of items is usually much less than that of users, we follow the common attitude to model the item-item rating similarities in this work. The symbols needed hereinafter are appointed in Table 1.

2.2. Related models

The RS-KNN model is a classical and still-popular CF model, for the ease of implementation and maintenance along with the intuitive nature. In this model, to construct the rating similarities between items/users is the precondition for predicting unknown user-item pairs; this task can be done in either user-oriented or item-oriented way. Usually, in a real world application the count of items is far less than that of users; therefore, modeling the item-item relationships is a common choice [4,7,15,18]. In this paper we also present our work based on the item-oriented attitude; however, the proposed strategies can also work with the user-oriented attitude.

Since RS-KNN measures the relationships between items with rating similarities, the appropriate modeling of RS is very important. The usual measurements of RS include the Cosine similarity, the Adjusted Cosine similarity and the PCC similarity [2–4]. According to pioneering works [2–4,20–23,31–33], recommenders with PCC can usually obtain relatively high prediction accuracy. Formally, the RS between two items i and j can be measured by PCC as follows:

$$s_{i,j} = s_{i,j}^{(PCC)} = \frac{\sum_{u\in U(i)\cap U(j)}(r_{u,i}-\bar{r}_i)\cdot(r_{u,j}-\bar{r}_j)}{\sqrt{\sum_{u\in U(i)\cap U(j)}(r_{u,i}-\bar{r}_i)^2}\cdot\sqrt{\sum_{u\in U(i)\cap U(j)}(r_{u,j}-\bar{r}_j)^2}}. \quad (1)$$

Afterwards, the system can build the KNN set of each item according to the obtained rating similarities. The estimation for an unknown rating relies on (a) the rating set of the active user u, (b) the KNN set of the specified item i, and (c) the corresponding rating similarities; formally given by [2–4]

$$\hat{r}_{u,i} = \bar{r}_i + \frac{\sum_{j\in KNN(i)\bigcap I(u)} s_{i,j}(r_{u,j}-\bar{r})}{\sum_{j\in KNN(i)\bigcap I(u)} s_{i,j}}. \quad (2)$$

Based on the prediction rule (2) and the PCC similarity (1), Papagelis et al. [28], and Miranda et al. [29], propose the incremental RS-KNN model (hereafter referred to as the I-KNN model). The main idea of this model is to decompose the PCC similarity into several components as follows,

$$s_{i,j}^{(PCC)} = \frac{B}{\sqrt{C}\cdot\sqrt{D}},$$
$$B = \sum_{u\in U(i)\cap U(j)}(r_{u,i}-\bar{r})_i\cdot(r_{u,j}-\bar{r}_j),\quad C = \sum_{u\in U(i)\cap U(j)}(r_{u,i}-\bar{r}_i)^2,\quad D = \sum_{u\in U(i)\cap U(j)}(r_{u,j}-\bar{r}_j)^2. \quad (3)$$

By caching each component and related factors, the corresponding PCC similarity can be incrementally updated according to the change of R_K without retraining the whole model. As described in [28], the factors related to the incremental-update of $s_{i,j}^{(PCC)}$ in I-KNN are listed in Table 2.

From Table 2 we see that the first 6 cached factors are connected with each PCC similarity. Consequently, the memory required by the I-KNN model will be 6 times as much as that required by the RS-KNN model. Since in the RS-KNN model the count of RS is quadratically related to the count of items, the extra memory required by the I-KNN model can be very large. In contrast, if we can decrease the number of the cached factors as well as maintaining the prediction accuracy and the incremental nature, the practicability of I-KNN can be improved. Besides, factors like the LB [15,20] and SS [3,20,30] have proved to be effective in improving the performance of the RS-KNN model. Thus, we can probably boost the performance of I-KNN by incrementally integrating these factors.

3. Method

3.1. Tuning the rating similarity

3.1.1. The generalized Dice coefficient

In the RS-KNN model, RS is the key parameter which directly decides the KNN set and affects the prediction accuracy. According to pioneering research, PCC is reliable to achieve relatively high prediction accuracy. However, as analyzed in Section 2, the formula of PCC is quite complex due to the dependence on the element-wise averages and the product of the norms, of the two involved rating vectors. As a result, the I-KNN model needs to cache 6 factors for implementing the incremental update of 1 rating similarity, and the required memory can be very large.

Here our objective is to decrease the cached factors while causing little harm on the prediction accuracy. Please note that in Eq. (1) the correlation between two rating vectors mainly depends on the covariance of the co-rated parts; the standard deviation is introduced mainly for normalization. Thus, by maintaining the numerator while manipulating the denominator of Eq. (1), the information of the correlation can be maintained while the formula can be simplified. Let $a = \sqrt{\sum_{u\in U(i)\cap U(j)}(r_{u,i}-\bar{r}_i)^2}$ and

Table 1
Symbol appointments.

Symbol	Comment
U/I	The set of users/items
u, v, w	To identify a specific user
i, j, k	To identify a specific item
R	The rating matrix
R_U/R_K	The set of unknown/known ratings in R
$r_{u,i}/\hat{r}_{u,i}$	The rating/estimation by user u on item i
$\bar{r}_i$	The average rating on item i
$R(u)/R(i)$	The rating set by user u/on item i
$U(i)/I(u)$	The set of users who rated item i/items rated by user u
$s_{i,j}$	The rating similarity between items i, j
K	The number specifying the nearest-neighborhood
$KNN(i)$	The KNN item set of item i
$ss_{i,j}/sw_{i,j}$	The similarity support/similarity support weight between items i, j
μ_r	The global average of observed ratings
b_u/b_i	The observed rating bias of user u/item i

Table 2
Cached factors for incrementally updating PCC similarity.

Factors	Comments	Related to
$B = \sum_{u\in U(i)\cap U(j)}(r_{u,i}-\bar{r})_i\cdot(r_{u,j}-\bar{r}_j)$	PCC components	Each RS $s_{i,j}$
$C = \sum_{u\in U(i)\cap U(j)}(r_{u,i}-\bar{r}_i)^2$	PCC components	Each RS $s_{i,j}$
$D = \sum_{u\in U(i)\cap U(j)}(r_{u,j}-\bar{r}_j)^2$	PCC components	Each RS $s_{i,j}$
$S_i^{i,j} = \sum_{u\in U(i)\cap U(j)} r_{u,i}$	Rating sum on item i by who have co-rated i, j	Each RS $s_{i,j}$
$S_j^{i,j} = \sum_{u\in U(i)\cap U(j)} r_{u,j}$	Rating sum on item j by who have co-rated i, j	Each RS $s_{i,j}$
$n_{i,j} = \lvert U(i)\cap U(j)\rvert$	Number of users who have co-rated i, j	Each RS $s_{i,j}$
$\bar{r}_i$	Average ratings on each item i	Each item i
$n_i = \lvert R(i)\rvert$	Count of ratings on each item i	Each item i

$b = \sqrt{\sum_{u\in U(i)\cap U(j)}(r_{u,j}-\bar{r}_j)^2}$. Since we have $a>0$ and $b>0$ (or the corresponding PCC value does not exist), we can proportionally amplify the denominator in Eq. (1) by introducing the coefficient $\left(\frac{b}{a}+\frac{a}{b}\right)$, lead to

$$s_{i,j}=s_{ij}^{(GDC)}=\frac{\sum_{u\in U(i)\cap U(j)}(r_{u,i}-\bar{r}_i)\cdot(r_{u,j}-\bar{r}_j)}{\sum_{u\in U(i)\cap U(j)}(r_{u,i}-\bar{r}_i)^2+\sum_{u\in U(i)\cap U(j)}(r_{u,j}-\bar{r}_j)^2}. \tag{4}$$

The similarity described by Eq. (4) can be actually interpreted as a kind of generalized Dice coefficients [34]. Note that this GDC similarity lies in the range of $[-0.5, 0.5]$, which is narrower than that of PCC in $[-1, 1]$. Nevertheless, since the RS values are used to sort neighbors and compute weighted averages, this range difference will be canceled and thus can be ignored. Through a simple deduction we can find that the GDC similarity can be incrementally updated through caching the factors summarized in Table 3.

With these cached factors, we can incrementally build the changes on each factor brought by the rating variations as follows,

$$\begin{cases} \textit{new } r_{v,i}\begin{cases} r_{v,j}\textit{ does not exist}\begin{cases} \bar{r}'_i=\frac{r_{v,j}+\bar{r}_i\cdot n_i}{1+n_i},\ n'_i=n_i+1,\\ \Delta\bar{r}_i=\bar{r}'_i-\bar{r}_i,\ B'=B-\Delta\bar{r}_i\cdot\left(S_j^{i,j}-n_{i,j}\cdot\bar{r}_j\right),\\ E'=E+n_{i,j}\cdot(\Delta\bar{r}_i)^2-2\Delta\bar{r}\cdot\left(S_j^{i,j}-n_{i,j}\cdot\bar{r}_i\right);\end{cases}\\ r_{v,j}\textit{ exists}\begin{cases} \bar{r}'_i=\frac{r_{v,j}+\bar{r}_i\cdot n_i}{1+n_i},\ n'_i=n_i+1,\ n'_{i,j}=n_{i,j}+1,\\ S_i^{i,j}=S_i^{i,j}+r_{v,i},\ S_j^{i,j}=S_j^{i,j}+r_{v,j},\\ \Delta\bar{r}_i=\bar{r}'_i-\bar{r}_i,\ B'=B-\Delta\bar{r}_i\cdot\left(S_j^{i,j}-n_{i,j}\cdot\bar{r}_j\right)+\left(r_{v,i}-\bar{r}'_i\right)\cdot(r_{v,j}-\bar{r}_j),\\ E'=E+n_{i,j}\cdot(\Delta\bar{r}_i)^2-2\Delta\bar{r}_i\cdot\left(S_j^{i,j}-n_{i,j}\cdot\bar{r}_i\right)+\left(r_{v,i}-\bar{r}'_i\right)^2+(r_{v,j}-\bar{r})^2;\end{cases}\end{cases}\\ \textit{changed } r'_{u,i}\begin{cases} r_{v,j}\textit{ does not exist}\begin{cases} \Delta r_{u,i}=r'_{u,i}-r_{u,i},\ \bar{r}'_i=\bar{r}_i+\frac{\Delta r_{u,i}}{n_i},\\ \Delta\bar{r}_i=\bar{r}'_i-\bar{r}_i,\ B'=B-\Delta\bar{r}_i\cdot\left(S_j^{i,j}-n_{i,j}\cdot\bar{r}_j\right),\\ E'=E+n_{i,j}\cdot(\Delta\bar{r}_i)^2-2\Delta\bar{r}\cdot\left(S_j^{i,j}-n_{i,j}\cdot\bar{r}_i\right);\end{cases}\\ r_{v,j}\textit{ exists}\begin{cases} \Delta r_{u,i}=r'_{u,i}-r_{u,i},\ \bar{r}'_i=\bar{r}_i+\frac{\Delta r_{u,i}}{n_i},\ S_i^{i,j}=S_i^{i,j}+\Delta r_{u,i},\\ \Delta\bar{r}_i=\bar{r}'_i-\bar{r}_i,\ B'=B-\Delta\bar{r}_i\cdot\left(S_j^{i,j}-n_{i,j}\cdot\bar{r}_j\right)+\left(r'_{u,i}-r_{u,i}\right)\cdot(r_{u,j}-\bar{r}),\\ E'=E+n_{i,j}\cdot(\Delta\bar{r}_i)^2-2\Delta\bar{r}_i\cdot\left(S_j^{i,j}-n_{i,j}\cdot\bar{r}_i\right)+(\Delta r_{u,i})^2+2\Delta r_{u,i}\cdot(r_{u,i}-\bar{r}'_i).\end{cases}\end{cases}\end{cases} \tag{5}$$

Please note that like I-KNN proposed in [28], in Eq. (5) we also consider four different situations according to the updating requirements. For a newly arrived rating, the incremental update firstly depend on whether this rating is already existing in R_K; (1) for a new rating, the main task is to take this rating into consideration and compute the increment on involved parameters; (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. Further, from Eq. (4), we can find that the GDC similarity fully relies on the co-rated part of the rating vectors on two items. Therefore, when we update $s_{i,j}^{(GDC)}$ according to a new rating $r_{v,i}/r_{u,i}$, it is also important to distinguish if the corresponding user v/u also rated item j. If $r_{v,j}/r_{u,j}$ exists, then the rating pair $(r_{v,i}, r_{v,j})/(r_{u,i}, r_{u,j})$ should be considered when update $s_{i,j}$; or else only the impact on the rating average should be taken into account.

After updating the factors shown in Eq. (5), the latest version of the corresponding rating similarity can be easily obtained. Moreover, as shown in Table 3, there are 5 cached factors connecting with each similarity when employing GDC to implement the incremental model. By comparing the cached factors listed in Tables 2 and 3 respectively, we can naturally infer the comparison in memory costs by the original I-KNN and the incremental model based on GDC as follows,

$$\frac{M_{I-KNN}}{M_{I-GDC}}=\frac{N_B+N_C+N_D+N_{S_i^{i,j}}+N_{S_j^{i,j}}+N_{n_{i,j}}+N_{\bar{r}_i}+N_{n_i}}{N_B+N_E+N_{S_i^{i,j}}+N_{S_j^{i,j}}+N_{n_{i,j}}+N_{\bar{r}_i}+N_{n_i}}, \tag{6}$$

where N_* denotes the total count of the corresponding cached factor listed in Tables 2 and 3. As analyzed in the former section, the count of RS in an RS-KNN model is quadratically related to the count of items/users. Thus, we can transform Eq. (6) into

$$\frac{M_{I-KNN}}{M_{I-GDC}}=\frac{6|I|^2+2|I|}{5|I|^2+2|I|}, \tag{7}$$

where $|I|$ denotes the cardinality of the item set. With the count of items large enough (which is usually satisfied in real-world applications), we can reasonably ignore the lower-order-terms $2|I|$, and arrive at the fact that with the GDC similarity the required memory for carrying out the incremental update will be about 16% less than that required by the original I-KNN model.

Further, if we compute the similarity with original ratings instead of residuals, Eq. (4) can be reformulated by

$$s_{i,j}=s_{i,j}^{(GDC2)}=\frac{\sum_{u\in U(i)\cap U(j)}r_{u,i}\cdot r_{u,j}}{\sum_{u\in U(i)\cap U(j)}r_{u,i}^2+\sum_{u\in U(i)\cap U(j)}r_{u,j}^2}, \tag{8}$$

where the similarity does not depend on the average rating of each item (this similarity is hereafter referred to as the GDC2 similarity). Through a simple induction we summarize the factors needed to incrementally update the GDC2 similarity in Table 4. These factors can be incrementally updated as follows,

$$\begin{cases} \textit{new } r_{v,i}\begin{cases} r_{v,j}\textit{ does not exist}: \textit{no update};\\ r_{v,j}\textit{ exists}\begin{cases} n'_i=n_i+1,\\ F'=F+r_{v,i}\cdot r_{v,j},\\ G'=G+r_{v,i}^2+r_{v,j}^2;\end{cases}\end{cases}\\ \textit{changed } r'_{u,i}\begin{cases} r_{u,j}\textit{ does not exist}: \textit{no update};\\ r_{u,j}\textit{ exists}\begin{cases} \Delta r_{u,i}=r'_{u,i}-r_{u,i},\ F'=F+\Delta r_{u,i}\cdot r_{u,j},\\ G'=G+2r_{u,i}\cdot\Delta r_{u,i}+(\Delta r_{u,i})^2.\end{cases}\end{cases}\end{cases} \tag{9}$$

Table 3
Cached factors for incrementally updating GDC similarity.

Factors	Comments	Related to
$B=\sum_{u\in U(i)\cap U(j)}(r_{u,i}-\bar{r}_i)\cdot(r_{u,j}-\bar{r}_j)$	GDC components	Each RS $s_{i,j}$
$E=\sum_{u\in U(i)\cap U(j)}(r_{u,i}-\bar{r}_i)^2+\sum_{u\in U(i)\cap U(j)}(r_{u,j}-\bar{r}_j)^2$	GDC components.	Each RS $s_{i,j}$
$S_i^{i,j}=\sum_{u\in U(i)\cap U(j)}r_{u,i}$	Rating sum on item i by who have co-rated i, j	Each RS $s_{i,j}$
$S_j^{i,j}=\sum_{u\in U(i)\cap U(j)}r_{u,j}$	Rating sum on item j by who have co-rated i, j	Each RS $s_{i,j}$
$n_{i,j}=\lvert U(i)\cap U(j)\rvert$	Number of users who have co-rated i, j	Each RS $s_{i,j}$
$\bar{r}_i$	Average rating on each item i	Each item i
$n_i=\lvert R(i)\rvert$	Count of ratings on each item i	Each item i

Table 4
Cached factors for incrementally updating GDC2 similarity.

Factors	Comments	Related to
$F=\sum_{u\in U(i)\cap U(j)}r_{u,i}\cdot r_{u,j}$	GDC2 components	Each RS $s_{i,j}$
$G=\sum_{u\in U(i)\cap U(j)}r_{u,i}^2+\sum_{u\in U(i)\cap U(j)}r_{u,j}^2$	GDC2 components	Each RS $s_{i,j}$

As shown in Table 4, only 2 parameters are related to each similarity when employing GDC2 similarity to implement the incremental KNN based CF. Similar to Eqs. (6) and (7), based on Tables 2 and 4 we can infer the memory-cost-comparison between the original I-KNN and the incremental model employing GDC2 as follows,

$$\frac{M_{I-KNN}}{M_{I-GDC2}} = \frac{N_B + N_C + N_D + N_{S_i^{i,j}} + N_{S_j^{i,j}} + N_{n_{i,j}} + N_{\bar{r}_i} + N_{n_i}}{N_F + N_G} = \frac{6|I|^2 + 2|I|}{2|I|^2} \tag{10}$$

Thus, with the value of $|I|$ large enough, the memory required will be approximately 34% of that required by the I-KNN model.

3.1.2. Validating the GDC similarities

As described in the former section, we can decrease the storage complexity of I-KNN via employing the GDC similarities. However, the premise of taking this strategy is the maintenance of performance; in other words, replacing the PCC coefficient with the GDC similarities should not result in the compromise in recommendation accuracy. For checking this issue, we conducted a simple experiment on a toy set. The experiment settings are given in Table 5.

Three models, which respectively employ PCC, GDC and GDC2 as the RS metric and Eq. (2) to estimate the unknown ratings, were tested in the experiment. Please note that the recommendations generated by these three models only differ in the RS metrics; so the comparison of MAE can reflect whether or not the employment of GDC may impact the prediction accuracy of the RS-KNN model.

The experiment results are given in Fig. 1. We can find that recommenders employing the GDC and GDC2 similarities kept outperforming the recommender with the PCC similarity during our experiment. Therefore, we can infer that the employment of GDC similarities will not impair the prediction accuracy of the RS-KNN model.

Table 5
Settings of experiment for validating GDC similarity.

	Contents	Comments
Dataset	ML10K	The MovieLens 10 K dataset consists of 10 K ratings by 943 users on 1682 items. The rating scale lies in [0, 5], and the density of the corresponding rating matrix is 6.30%. During the experiment the 5-fold cross-validation was employed
Metric	MAE [35,36]	$MAE = \sum_{(u,i)\in V}\|r_{u,i} - \hat{r}_{u,i}\|/\|V\|$, the lower the better
Parameter	K	Tested values rage from 50 to 1650

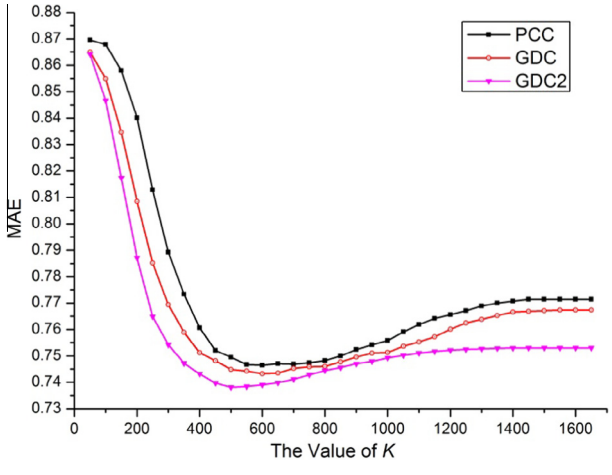

Fig. 1. The comparison of accuracy among all 3 tested models.

3.2. Integrating the similarity support

SS is a very useful factor which can be used to scale rating similarities and diminish the unreasonable recommendations [3,20,30]. For integrating SS into the I-KNN model, it is necessary to explore the incrementally updating rule of this factor with regard to rating variations.

Generally, SS is the count of the supporting samples for each similarity and can be given by

$$ss_{i,j} = n_{i,j} = |U(i) \cap U(j)|. \tag{11}$$

According to Eq. (11), SS can be easily updated according to the rating variation by each user. However, since we intend to employ SS as a shrinkage factor to scale each rating similarity and improve the KNN reliability of each item, we choose to model the SS values as random variables from a Gaussian distribution of which the mean and variation can be respectively estimated by [30]

$$\hat{\mu}_{ss} = \overline{ss} = \frac{\sum ss_{i,j}}{N}, \quad \hat{\sigma}_{ss}^2 = \frac{\sum ss_{i,j}^2}{N} - \hat{\mu}_{ss}^2. \tag{12}$$

With $\hat{\mu}_{ss}$ and $\hat{\sigma}_{ss}^2$ we can use the cumulative distributive function to estimate the probability that $ss_{i,j}$ is greater than or equal to the others as the SS weight for each RS, given by

$$sw_{i,j} = F(ss_{i,j}|\hat{\mu}_{ss}, \hat{\sigma}_{ss}^2) = \int_{-\infty}^{ss_{ij}} f(ss|\hat{\mu}_{ss}, \hat{\sigma}_{ss}^2)dss. \tag{13}$$

Note that we can reduce the computational cost of (13) via Taylor's approximation. And then the solved SS weight can be applied to scaling the corresponding rating similarity as

$$s'_{i,j} = s_{i,j} \cdot sw_{i,j}. \tag{14}$$

If we integrate SS into our incremental model, it is necessary to implement the incremental update for each SS weight when data in R_U vary. Fortunately, this can be easily implemented by caching the current $\hat{\mu}_{ss}, \hat{\sigma}_{ss}^2, N$ and the SS values. With the variation of the rating data, $\hat{\mu}_{ss}$ and $\hat{\sigma}_{ss}^2$ can be updated by

$$\begin{cases} \text{with a new } ss_{i,j},\ \hat{\mu}'_{ss} = \frac{\hat{\mu}_{ss}\cdot N + ss_{i,j}}{N+1},\ (\hat{\sigma}'_{ss})^2 = \frac{(\hat{\sigma}_{ss}^2 + \hat{\mu}_{ss}^2)\cdot N + ss_{i,j}^2}{N+1} - (\hat{\mu}'_{ss})^2; \\ \text{with a changed } ss'_{i,j},\ \begin{cases} \Delta ss = ss'_{i,j} - ss_{i,j}, \\ \hat{\mu}'_{ss} = \hat{\mu}_{ss} + \frac{\Delta ss}{N},\ (\hat{\sigma}'_{ss})^2 = \hat{\sigma}_{ss}^2 + \frac{2\Delta ss\cdot ss_{i,j} + \Delta ss^2}{N} + \hat{\mu}_{ss}^2 - (\hat{\mu}'_{ss})^2; \end{cases} \end{cases} \tag{15}$$

and the updated SS weights can be obtained via the renewed mean and variation to implement the incremental update of the whole model. For reducing the computational complexity, we can cache SS values and compute the corresponding SS weights when required; $\hat{\mu}_{ss}$ and $\hat{\sigma}_{ss}^2$ can be instantly updated with the change of R_K. The factors required to implement the incremental update of the CF model with SS is summarized in Table 6.

Table 6
Cached factors for integrating SS.

Factors	Comments
$ss_{i,j}(equal\ to\ n_{i,j} = \|U(i) \cap U(j)\|)$	The similarity support for $s_{i,j}$. Equal to the number of users who have co-rated items i, j
$\hat{\mu}_{ss}$	The estimation of the Gaussian mean
$\hat{\sigma}_{ss}^2$	The estimation of the Gaussian variation
N	Total number of the similarity supports in the current model

3.3. Integrating the linear biases

According to pioneer research [15,20], the integration of LB has obvious effect in the further improvement of the RS-KNN model. Therefore, it is expected to improve the accuracy of the I-KNN model with the same integration as well as implementing the corresponding incremental updates. As described in [20], the linear biases beneficial to the prediction accuracy of the RS-KNN model mainly include the global rating average μ_r, the observed user bias b_u and the observed item bias b_i. These three factors can be respectively computed by [20]

$$\begin{aligned} \mu_r &= \sum_{(u,i)\in R_K} r_{u,i}/|R_K|, \\ b_i &= \sum_{(u,i)\in R(i)} (r_{u,i}-\mu)/(\beta_1+|R(i)|), \\ b_u &= \sum_{(u,i)\in R(u)} (r_{u,i}-\mu-b_i)/(\beta_2+|R(u)|). \end{aligned} \tag{16}$$

And with the integration of these 3 factors, the prediction rule (2) is reformulated into:

$$\hat{r}_{u,i} = \mu_r + b_u + b_i + \frac{\sum_{j\in KNN(i)\bigcap I(u)} s_{i,j}(r_{u,j}-\mu_r-b_u-b_j)}{\sum_{j\in KNN(i)\bigcap I(u)} s_{i,j}}, \tag{17}$$

Note that in our design RS will not be affected by the integration of linear biases. So, we only have to obtain the updating rule for each integrated factor. First of all, we can incrementally update μ_r according to the rating variation as

$$\begin{cases} \text{with a new } r_{i,j},\ \mu_r' = \frac{\mu_r\cdot|R_U|}{1+|R_U|}; \\ \text{with a changed } r_{i,j}',\ \Delta r = r_{i,j}' - r_{i,j},\ \mu_r' = \mu_r + \frac{\Delta r}{|R_U|}. \end{cases} \tag{18}$$

Table 7
Cached factors for integrating LB.

Factors	Comments
μ_r	The current global rating average
$\|R_U\|$	Total count of known ratings
S_u	The sum of ratings by each user
S_i	The sum of ratings on each item
$n_u = \|R(u)\|$	Count of ratings by each user
$n_i = \|R(i)\|$	Count of ratings on each item

In terms of b_u and b_i, we can employ the strategy described in [27] to cache S_u, S_i, $|R(u)|$ and $|R(i)|$ (please refer to Table 7) which can be incrementally updated easily, and then compute the updated observed biases when needed as follows,

$$b_i = \frac{S_i-\mu_r\cdot|R(i)|}{\beta_1+|R(i)|}, \quad b_u = \frac{S_u-\mu_r\cdot|R(u)|-\sum_{i\in I(u)} b_i}{\beta_2+|R(u)|}. \tag{19}$$

To implement the integration mentioned above, we have to cache the factors summarized in Table 7.

3.4. The boosted KNN based incremental CF

Based on the investigation through RS, SS and LB, we are able to design the boosted incremental RS-KNN models (hereafter referred to as the BI-KNN models). In the proposed models, we choose to employ the two GDC similarities separately, for lowering the storage complexity; thus, the RS values will be solved by Eq. (4) or Eq. (8). Since we intend to employ the SS along with the LB, the BI-KNN models will scale rating similarities with SS weights according to Eq. (14), and make predictions with Eq. (17). With the rating variation, the BI-KNN models will (1) update the corresponding rating similarities incrementally according to Eq. (5) or Eq. (9), (2) update the factors related to SS according to Eq. (15) and (3) update factors related to LB according to Eqs. (18) and (19).The required factors in the BI-KNN models are summarized in Table 8.

Due to the integration of SS and LB, more factors need to be cached in the BI-KNN models than in the I-KNN model. However, from Table 8 we see that depending on the employed RS metrics, The BI-KNN models need to cache 5 or 3 factors quadratically related to the item count while the others are either linearly related to the item count or single parameters; thus, by reasonably ignoring these lower order terms we can conclude that the storage complexity of the BI-KNN model is much less than the I-KNN model which need to cache 6 factors quadratically related to the item count.

However, besides the storage complexity, we are also concerning about the prediction accuracy. It is important to check the effect brought by each integrated factor, and validate if the proposed models can provide recommendations as well as, or better than the I-KNN model. This issue will be examined in the next section.

Table 8
Cached factors in the BI-KNN models.

Purpose	BI-KNN	BI-KNN2	Related to
For RS	$B = \sum_{u\in U(i)\cap U(j)}(r_{u,i}-\bar{r}_i)\cdot(r_{u,j}-\bar{r}_j)$	$F = \sum_{u\in U(i)\cap U(j)} r_{u,i}\cdot r_{u,j}$	Each RS $s_{i,j}$
For RS	$E = \sum_{u\in U(i)\cap U(j)}(r_{u,i}-\bar{r}_i)^2 + \sum_{u\in U(i)\cap U(j)}(r_{u,j}-\bar{r}_j)^2$	$G = \sum_{u\in U(i)\cap U(j)} r_{u,i}^2 + \sum_{u\in U(i)\cap U(j)} r_{u,j}^2$	Each RS $s_{i,j}$
For RS	$S_i^{i,j} = \sum_{u\in U(i)\cap U(j)} r_{u,i}$		Each RS $s_{i,j}$
For RS	$S_j^{i,j} = \sum_{u\in U(i)\cap U(j)} r_{u,j}$		Each RS $s_{i,j}$
For RS	$n_{i,j} = \|U(i)\cap U(j)\|$		Each RS $s_{i,j}$
For RS	$\bar{r}_i$		Each item i
For RS	$n_i = \|R(i)\|$		Each item i
For LB	μ_r	μ_r	Only one
For LB	$\|R_U\|$	$\|R_U\|$	Only one
For LB	S_u	S_u	Each user u
For LB	$n_u = \|R(u)\|$	$n_u = \|R(u)\|$	Each user u
For LB	S_i	S_i	Each item i
For LB	$n_i = \|R(i)\|$[a]	$n_i = \|R(i)\|$	Each item i
For SS	$ss_{i,j} = \|U(i)\cap U(j)\|$[b]	$ss_{i,j} = \|U(i)\cap U(j)\|$	Each RS $s_{i,j}$
For SS	$\hat{\mu}_{ss}$	$\hat{\mu}_{ss}$	Only one
For SS	$\hat{\sigma}_{ss}^2$	$\hat{\sigma}_{ss}^2$	Only one
For SS	N	N	Only one

[a] In BI-KNN, n_i will be used to implement the incremental update of RS and LB, so only one copy for each is required.
[b] In BI-KNN, $ss_{i,j}$ is equal to $n_{i,j}$ for implementing the incremental update of RS, so only one copy for each is required.

4. Experiments

Tested models. For validating the effect of each integrated factor, we have first implemented the I-KNN model as the benchmark, and then integrated the GDC similarities, the SS factors, and the LB factors into this model by step. All involved models are given in Table 9.

Datasets. The experiments are conducted on two datasets, which are respectively the MovieLens 1 M (hereafter referred to as the ML1M) and the EachMovie (hereafter referred to as the EM2.8 M). ML1M is collected by the GroupLens Research Project at the University of Minnesota through the MovieLens web site, and contains 1 million ratings applied to 3900 movies by 6040 users with a scale on the [0, 5] interval. The data density of the ratings matrix in ML1M is 4.25%. EM2.8 M is collected through the EachMovie recommender by the DEC Systems Research Center, and contains over 2.8 million ratings ranging in the scale of [0.0, 1.0], entered by 72,916 users for 1628 different movies. The data density of E2.8 M is 2.37%.

Table 9
All tested models in the experiments.

Model	Comments
I-KNN	The benchmark which is implemented according to [28]
GDC	The incremental RS-KNN model with GDC
GDC2	The incremental RS-KNN model with GDC2
GDC-S	The incremental RS-KNN model with GDC and SS
GDC2-S	The incremental RS-KNN model with GDC2 and SS
BI-KNN	The BI-KNN model with GDC, SS and LB
BI-KNN2	The BI-KNN2 model with GDC2, SS and LB

During our experiment, for make the MAE on two datasets keep in the same order, we map the ratings in EM2.8 M into the interval of [0, 5]. Both datasets are split into 3 parts, respectively are (a) the static part, which is used to simulate the initially given data; (b) the incremental part, which is used to simulate the incremental variation of the rating matrix; and (c) the validation part, the data in which are used to check the performance of each model. The static part, incremental part and validation part respectively take up 10%, 40% and 50% ratings of the original dataset.

Evaluation metric. We mainly focus on testing the prediction accuracy of each model, so we employ MAE as the evaluation metric. We are also very interested in the stability of each model under the incremental circumstance. It is important the check that with the continuous input of incremental data whether or not each model can maintain the prediction accuracy steadily.

Experiment process. The experiments are conducted as follows,

(a) training the base model on the static part of ratings;
(b) tuning the optimal value of K for each tested model;
(c) gradually pushing the incremental ratings into the system, and updating the model accordingly.

Results. The experiment results are depicted in Fig. 2. Fig. 2(a) and (c) give the K-tuning process on ML1M and EM2.8 M respectively. Please note that with no incremental data, the I-KNN model and the other tested models can be regarded as static RS-KNN models with different rating similarity metrics and different prediction rules. According to the tuning results, we subsequently employ the optimal value of K and carried out the validation for the accuracy of each model under the incremental circumstances.

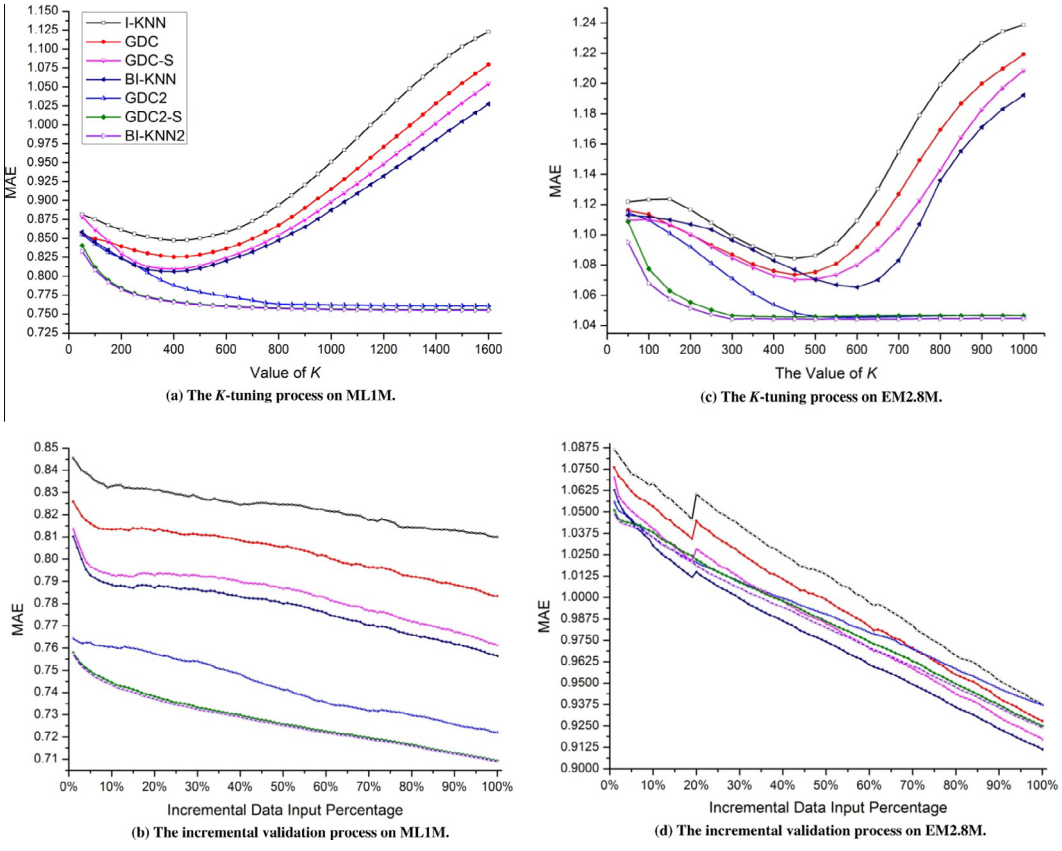

Fig. 2. The experiment results. All panels share the legend in Panel (a).

The results of this part are depicted in Fig. 2(b) and (d). From these results, we have several findings.

- With very few training instances, with the GDC2 similarity the recommender can generally obtain higher accuracy than those with PCC and GDC similarities. As shown in Fig. 2(a) and (b), during our *K*-tuning process, the tested models using GDC2 similarity always make predictions more accurate than those using PCC and GDC similarity. This might be caused by the ignorance of item-rating-average in GDC2 similarity. Without enough observed examples, this parameter may fail to capture the statistic characteristics in the rating pattern of each item; thus, the corresponding similarity metrics like PCC and GDC cannot play very well in measuring the item-item relationship.
- In general, by employing the GDC similarities and integrating SS and LB, the proposed incremental models can obtain more accurate recommendations according to rating data variations. On ML1M, the BI-KNN2 model is able to make the most accurate predictions throughout the whole testing process. On EM2.8 M, the situation is a little different. In the beginning, the BI-KNN2 model can make predictions more accurate than the other models; however, with the continuous input of incremental data, it is gradually outperformed by BI-KNN and GDC-S. Please note that BI-KNN and BI-KNN2 are only differed by the RS metrics, of which the difference lies in whether or not integrating the item-rating-average. Thus, this phenomenon again suggests that without enough training examples the rating similarity relying on rating statistics might fail to capture the characteristics of item-item relationship, and vice versa.
- Each integrated factor can improve the recommendation accuracy to some extent. During the *K*-tuning process, the GDC based models can always outperform the benchmark with PCC similarity, and are usually outperformed by the GDC2 based models. From Fig. 2(a) and (c), we can clearly recognize that when employing the GDC and GDC2 similarities, the integration of SS and LB can further boost the prediction accuracy respectively. Under the incremental settings, however, the situation is a bit different since on EM2.8 M where the GDC2 based models are outperformed by the GDC based models. Nevertheless, for models employing the same rating similarity metric, the prediction accuracy can still improve with the integration of SS and LB.
- In terms of the model stability, based on the experiment results we have located 3 factors useful for stabilizing the prediction accuracy in incremental circumstances, which are respectively the GDC2 similarity, the SS and the LB. This can be proved by the MAE changing tendency during the incremental performance validations on both datasets. As shown in Fig. 2(b), on ML1M the prediction accuracy of the benchmark fluctuates at several points (e.g., at 10%, 30% and 72%) with the incremental input. By replacing the PCC similarity with the GDC similarity, the recommender can obtain higher accuracy; however, the prediction accuracy also fluctuates at several points. And then, with the integration of SS and LB, the GDC-S and BI-KNN models can consecutively narrow the range of fluctuation at these points. This can be seen more clearly from Fig. 2(d). At the point of 21%, the MAE of I-KNN increases sharply, which is probably caused by some noises. With the same input, the recommender based on GDC confronts with the same problem. However, at the same point, the GDC-S and BI-KNN obviously narrow the range of fluctuations by integrating SS and LB.

Moreover, by employing the GDC2 similarity, the GDC2-S and BI-KNN2 can obtain better stability during the incremental updating processes. As shown in Fig. 2(b), compared to the other models, GDC2-S and BI-KNN2 can obtain rather smooth MAE-variation-curves without obvious fluctuations. This phenomenon also remains in the experiments on the EM2.8 M. At the point of 21% where I-KNN and BI-KNN confront a sharp rise of MAE, the GDC2 based models can still maintain the prediction accuracy. Thus, we can infer that the employment of GDC2 along with SS and LB has positive effect in stabilizing the prediction accuracy.

To summarize, based on the experiment results we see that the two proposed models BI-KNN and BI-KNN2 can obtain obvious advantage in prediction accuracy in incremental circumstances.

4.1. BI-KNN and I-RMF

As mentioned in Section 1, the NBM and LFM possess different principles; these two CF branches can be employed under different circumstances according to specified requirements. However, since the BI-KNN, BI-KNN2 and I-RMF all focus on modeling the increment brought by rating variations to implement quick responses to newly obtained user behaviors, it is interesting to make a simple comparison between these two models.

We have conducted a simple experiment to compare the prediction accuracy of BI-KNN, BI-KNN2 and I-RMF on ML1M and EM2.8 M. In this experiment, I-KNN is also included as the benchmark. The settings of BI-KNN and BI-KNN2 are the same as in the previous experiment. Following the instructions in [27], the learning rate η and the regularizing parameter λ of I-RMF are set at 7×10^{-4} and 0.02 on ML1M, and 2×10^{-4} and 0.02 on EM2.8 M, respectively. The dimension of the latent factor space is set at 20.

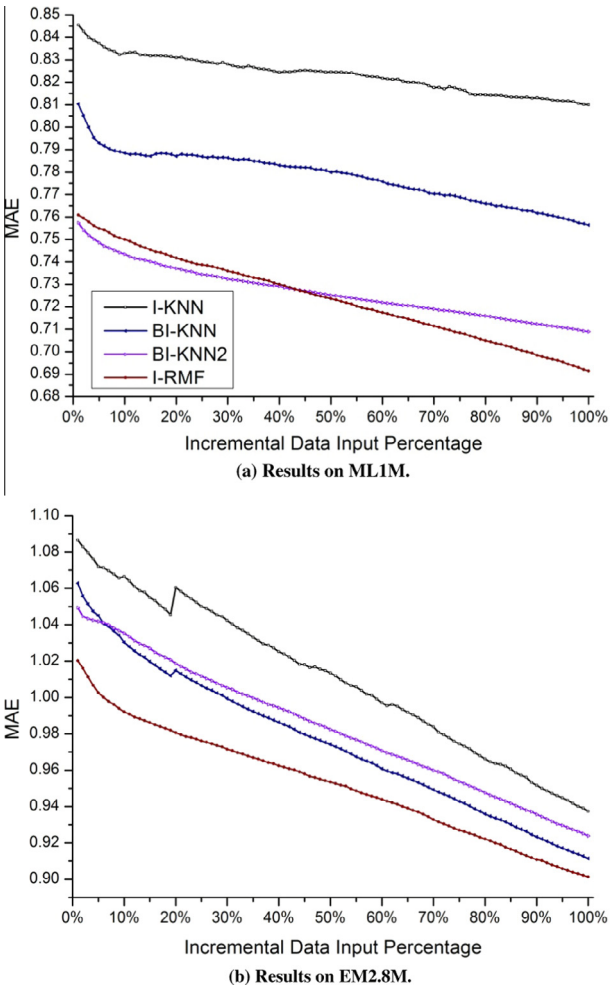

Fig. 3. MAE comparison between BI-KNN and I-RMF. Both the panels share the same legend in panel (a).

Table 10
Average time (ms) for integrating 1000 incremental ratings by BI-KNN and I-RMF on both datasets.

Model	ML1M	EM2.8 M
BI-KNN	1219.61	545.73
BI-KNN2	763.86	331.52
I-RMF	3777.90	7973.17

The detailed results of this experiment are depicted in Fig. 3. During this experiment, we have also recorded the average time to for integrating 1000 incremental ratings into all three incremental models, for measuring the computational efficiency. These records are shown in Table 10. Please note that all experiments have been implemented by Java and conducted on a PC with a 2.5 GHz CPU and 8 GB memory.

From Fig. 3, we can find that I-RMF can still outperform BI-KNN in prediction accuracy; however, after employing the boosting strategies, the accuracy difference between BIKNN and I-RMF is much less than that between the original I-KNN and I-RMF. As shown in Fig. 3(a), after integrating all incremental ratings, the MAE of I-KNN, BI-KNN, BI-KNN2 and I-RMF are at 0.8100, 0.7564, 0.7089 and 0.6914 respectively. The similar situation can be found from the experiment results on EM2.8 M, where the final MAE of I-KNN, BI-KNN, BI-KNN2 and I-RMF are 0.9374, 0.9113, 0.9239 and 0.9013 respectively. Thus, we can summarize that the proposed models BI-KNN and BI-KNN2 can provide predictions nearly as accurate as those by I-RMF.

In the meanwhile, on both datasets, BI-KNN and BI-KNN2 have shown higher computational efficiency in integrating incremental ratings than I-RMF, as shown in Table 10. This phenomenon can be interpreted by the principle of these 3 models. Since all 3 incremental models work by modeling the increment brought by rating variations, the total computational complexity relies on the cost of integrating each new rating. From Section 3, we can find that for each incremental rating $r_{u,i}$, BI-KNN and BI-KNN2 is required to check each rating similarity $\{s_{i,j}|j \in I, j \neq i\}$; thus, the corresponding computational cost is bound by $O(|I|)$. Further, from Eqs. (5) and (9) we see that BI-KNN requires updating more parameters in correspondence to an incremental rating than BI-KNN2; so BI-KNN2 has higher computational efficiency, as shown in Table 10. With regard to I-RMF, as discussed in [27], for each incremental rating $r_{u,i}$, the update cost is bound by $O\left(\left(\frac{|R_K|}{|U|}+\frac{|R_K|}{|I|}\right)\times D\times f\right)$, where D and f respectively denote the count of training iterations and the dimension of the latent factor space. Since the condition $|I| \ll |U|$ is usually satisfied in practical recommender systems, this bound can be approximated by $O\left(\frac{|R_K|}{|I|}\times D\times f\right)$. So, we can obtain that

$$O\left(\frac{C_{BI-KNN}}{C_{I-RMF}}\right) \approx O\left(\frac{|I|^2}{|R_K|\times D\times f}\right), \tag{20}$$

where T_{BI-KNN} and T_{I-RMF} respectively represent for the update cost function of BI-KNN/BI-KNN2 and I-RMF. Thus, with the common condition that $|I| \ll |U|$, the update cost of I-RMF is usually higher than that of BI-KNN/BI-KNN2, as shown in Table 10. Besides, as discussed in Section 3, BI-KNN/BI-KNN2 can deal with rating changes along with new ratings; in contrast, I-RMF can only deal with new ratings. So, in real world applications BI-KNN/BI-KNN2 and I-RMF can be employed according to the specified requirements.

5. Conclusion

In this work we aim at boosting the performance of the RS-KNN based incremental CF. We mainly focus on two issues, which are respectively lowering the storage complexity and improving the prediction accuracy. By investigating the I-KNN model, we find that the storage complexity mainly relies on the complicate form of PCC. In order to decrease this storage complexity while maintaining the prediction accuracy, we replace PCC with GDC similarities. With simpler formulas, the two employed GDC similarities can obviously reduce the memory cost. And from the experiment results we see that the prediction accuracy can be slightly improved by employing these two similarities. After that, we further integrate the SS and LB for boosting and stabilizing the prediction accuracy under incremental circumstances. The positive effects of our strategies are supported by the experiment results. Thus, we can arrive at the conclusion that to boost the RS-KNN based incremental CF through employing the GDC similarities and integrating SS and LB is feasible.

One interesting phenomenon found in our research is that without the reliance on the rating statistics such as the item rating average, the GDC2 similarity based recommender can outperform the rating-statistic-relying similarity (including GDC and PCC) based models when given few training examples. However, with more given instances the situation may change as in our experiment on the EM2.8 M. This point will be covered by our future work.

Table A.1
Acronyms used in the paper body.

Acronym	Meaning
CF	Collaborative Filtering
NBM	Neighborhood Based Model
LFM	Latent Factor Model
RS	Rating Similarity
KNN	K-Nearest-Neighborhood
RS-KNN	Rating Similarity based K-Nearest Neighborhood
PCC	Pearson Correlation Coefficient
SS	Similarity Support
GDC	Generalized Dice Coefficient
LB	Linear Biases

Acknowledgements

The Authors are very grateful to the anonymous reviewers for their insightful comments. This research is supported by the National Natural Science Foundation of China with Grant No. 61202347, the National Natural Science Foundation of China with Grant No. 61103036, the Fundamental Research Funds for the Central Universities with Project No. CDJZR12180012, and the Natural Science Foundation Project of CQ CSTC with Grant No. cstc2012jjA40016.

Appendix A

In this section we provide the table listing the acronyms which are used in the paper body for improving the readability. Please refer to the Appendix Table A.1 as follows.

References

[1] D. Goldberg, D. Nichols, B.M. Oki, D. Terry, Using collaborative filtering to weave an information tapestry, Communications of the ACM 35 (1992) 61–70.
[2] J. Herlocker, J. Konstan, A. Borchers, J. Riedl, An algorithmic framework for performing collaborative filtering, in: 22nd ACM SIGIR, ACM, Berkeley, California, United States, 1999, pp. 230–237.
[3] J. Herlocker, J. Konstan, J. Riedl, An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms, Information Retrieval 5 (2002) 287–310.
[4] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Transactions on Knowledge and Data Engineering 17 (2005) 734–749.
[5] J. Bobadilla, F. Serradilla, A. Hernando, Collaborative filtering adapted to recommender systems of e-learning, Knowledge-Based Systems 22 (2009) 261–265.

[6] Q. Shambour, J. Lu, A trust-semantic fusion-based recommendation approach for e-business applications, Decision Support Systems 54 (1) (2012) 768–780.
[7] B. Sarwar, G. Karypis, J. Konstan, J. Reidl, Application of dimensionality reduction in recommender systems—a case study, in: Proceedings of the ACM WebKDD 2000, ACM, 2000, pp. 285–295.
[8] T. Hofmann, Latent semantic models for collaborative filtering, ACM Transactions On Information Systems 22 (2004) 89–115.
[9] N. Srebro, J.D.M. Rennie, T.S. Jaakola, Maximum-margin matrix factorization, Advances in Neural Information Processing Systems 17 (2005) 1329–1336.
[10] M. Kurucz, A. Benczúr, K. Csalogány, Methods for large scale SVD with missing values, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, California, United States, 2007, pp. 7–14.
[11] A. Paterek, Improving regularized singular value decomposition for collaborative filtering, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, California, United States, 2007, pp. 39–42.
[12] G. Takács, I. Pilászy, Németh Bottyán, D. Tikky, Scalable collaborative filtering approaches for large recommender systems, Journal of Machine Learning Research 10 (2009) 623–656.
[13] R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, Advances in Neural Information Processing Systems 20 (2008) 1257–1264.
[14] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, IEEE Computer 42 (2009) 30–37.
[15] Y. Koren, R. Bell, Advances in collaborative filtering, in: F. Ricci, L. Rokach, B. Shapira, P.B. Kantor (Eds.), Recommender Systems Handbook, Springer US, New York, United States, 2011, pp. 145–186.
[16] J. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in: 14th International Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, United States, 1998, pp. 43–52.
[17] B. Sarwar, G. Karypis, J. Konstan, J. Reidl, Item based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, ACM, Hong Kong, China, 2001, pp. 285–295.
[18] M. Deshpande, G. Karypis, Item-based top-N recommendation algorithms, ACM Transactions On Information Systems 22 (2004) 143–177.
[19] M. Komkhaoa, J. Lu, Z. Lia, W.A. Halanga, Incremental collaborative filtering based on mahalanobis distance and fuzzy membership for recommender systems, International Journal of General Systems 42 (2013) 41–66.
[20] R. Bell, Y. Koren, Improved neighborhood-based collaborative filtering, in: KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, California, United States, 2007, pp. 7–14.
[21] Z. Zhang, H. Lin, K. Liu, D. Wu, G. Zhang, J. Lu, A hybrid fuzzy-based personalized recommender system for telecom products/services, Information Sciences 235 (2013) 117–129.
[22] J. Bobadilla, F. Ortega, A. Hernando, J. Bernal, Generalization of recommender systems: collaborative filtering extended to groups of users and restricted to groups of items, Expert Systems with Applications 39 (2012) 172–186.
[23] Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering model, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, Neveda, United States, 2008, pp. 426–434.
[24] D. Agarwal, B.-C. Chen, P. Elango, Fast online learning through offline initialization for time-sensitive recommendation, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, Washington. D.C., United States, 2010, pp. 703–712.
[25] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Incremental singular value decomposition algorithms for highly scalable recommender systems, in: Proceedings of the 5th International Conference on Computer and Information Science, 2002, pp. 27–28.
[26] S. Rendle, L. Schmidt-Thieme, Online-updating regularized kernel matrix factorization models for large-scale recommender systems, in: Proceedings of the 2nd ACM Conference on Recommender Systems, ACM, Lausanne, Switzerland, 2008, pp. 251–258.
[27] X. Luo, Y.-N. Xia, Q.-S. Zhu, Incremental collaborative filtering recommender based on regularized matrix factorization, Knowledge-Based Systems 27 (2012) 271–280.
[28] M. Papagelis, R. Ioannis, D. Plexousakis, E. Theoharopoulos, Incremental collaborative filtering for highly-scalable recommendation algorithms, in: Proceedings of the 15th International Symposium on Methodologies for Intelligent Systems, Springer Verlag Berlin, New York, United States, 2005, pp. 553–561.
[29] C. Miranda, A. Jorge, Item-based and user-based incremental collaborative filtering for web recommendations, in: Proceedings of the 14th Portuguese Conference on Artificial Intelligence, Springer Verlag Berlin, Aveiro, Portugal, 2009, pp. 673–684.
[30] X. Luo, Y.-X. Ouyang, Z. Xiong, Improving K-Nearest-Neighborhood based collaborative filtering via similarity support, International Journal of Digital Content Technology and its Applications 5 (2011) 248–256.
[31] C. Cornelis, J. Lu, X. Guo, G. Zhang, One-and-only item recommendation with fuzzy logic techniques, Information Sciences 177 (2007) 4906–4921.
[32] H.-N. Kim, A.-T. Ji, I. Ha, G.-S. Jo, Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation, Electronic Commerce Research and Applications 9 (2010) 73–83.
[33] V. Moscato, A. Picariello, A.M. Rinaldi, Towards a user based recommendation strategy for digital ecosystems, Knowledge-Based Systems 37 (2013) 165–175.
[34] L.R. Dice, Measures of the amount of ecologic association between species, Ecology 26 (1945) 297–302.
[35] J. Herlocker, J. Konstan, L. Terveen, J. Riedl, Evaluating collaborative filtering recommender systems, ACM Transactions On Information Systems 22 (2004) 5–53.
[36] G. Shani, A. Gunawardana, Evaluating recommendation systems, in: F. Ricci, L. Rokach, B. Shapira, P.B. Kantor (Eds.), Recommender Systems Handbook, Springer US, New York, United States, 2011, pp. 257–297.