Information Sciences 229 (2013) 29–39

Contents lists available at SciVerse ScienceDirect

Information Sciences

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

Unifying rating-oriented and ranking-oriented collaborative filtering for improved recommendation

Yue Shi *, Martha Larson, Alan Hanjalic

Multimedia Information Retrieval Lab, Department of Intelligent Systems, Delft University of Technology, Netherlands

ARTICLE INFO

Article history:
Received 23 February 2012
Received in revised form 1 November 2012
Accepted 2 December 2012
Available online 28 December 2012

Keywords:
Collaborative filtering
Matrix factorization
Ranking
Recommender systems
Unified recommendation model

ABSTRACT

We propose a novel unified recommendation model, URM, which combines a rating-oriented collaborative filtering (CF) approach, i.e., probabilistic matrix factorization (PMF), and a ranking-oriented CF approach, i.e., list-wise learning-to-rank with matrix factorization (ListRank). The URM benefits from the rating-oriented perspective and the ranking-oriented perspective by sharing common latent features of users and items in PMF and ListRank. We present an efficient learning algorithm to solve the optimization problem for URM. The computational complexity of the algorithm is shown to be scalable, i.e., to be linear with the number of observed ratings in a given user-item rating matrix. The experimental evaluation is conducted on three public datasets with different scales, allowing validation of the scalability of the proposed URM. Our experiments show the proposed URM significantly outperforms other state-of-the-art recommendation approaches across different datasets and different conditions of user profiles. We also demonstrate that the primary contribution to improve recommendation performance is contributed by the ranking-oriented component, while the rating-oriented component is responsible for a significant enhancement.

1. Introduction

Recommender systems attract research attention because they are able to connect users directly with consumable items, supporting them in handling the unprecedentedly large amounts of content, e.g., movies, music and books currently available online by providing personalized recommendations [1,6]. Collaborative filtering (CF) is widely acknowledged as one of the most successful recommender techniques. Compared to content-based approaches, CF enjoys the advantage of being content-agnostic. In other words, it can recommend items without the additional computational expense or copyright issues involved with processing items directly. One of two different types of approaches can be taken by a recommender system in order to generate recommendation lists for users. Under one approach, the system predicts ratings for individual items first and then generates the ranked recommendation list. We refer to this type of CF-based recommendation as rating-oriented [7,12,22]. Under the other approach, the system predicts rank scores, that are not necessarily related to ratings, but rather used directly to generate the recommendation list. We refer to this type of approach as ranking-oriented [14,15,27,30,31].

To illustrate the difference between rating- and ranking-oriented CF, we consider two specific toy examples. The first example involves the ratings of a user on items i and j. We assume that the user has rated item i with a 4 and item j with a 3; these are the reference values that we use to judge the quality of the predictions of the recommender system. If two recommendation approaches give rating predictions of (3,4), and (5,2) on items (i,j), the rating prediction error, e.g.,

* Corresponding author. Address: HB.11.070, Mekelweg 4, 2628CD Delft, Netherlands. Tel.: +31 152785812; fax: +31 152781843.
E-mail address: y.shi@tudelft.nl (Y. Shi).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.12.002

measured by mean absolute error or root mean square error [8] will be the same for both approaches. However, only the ranking-oriented perspective identifies the second approach as faithfully reflecting the users relatively higher preference for item i over item j. This example should not lead to the conclusion that working with absolute ratings is detrimental to recommendation performance. Quite to the contrary, successful recommender systems do use a rating-oriented approach to generate recommendation lists for users, e.g., MovieLens [7] and Netflix [12]. Our second example illustrates the usefulness of absolute ratings in capturing users preference strength. If user u and v have ratings $(5,3)$ and $(4,3)$ on items (i,j), user u is more explicit about his preference for item i over item j than user v. This information holds the potential to help resolve possible ambiguities in generating a ranked item list for the user u. Further, predicted ratings can provide the user with additional information used to inform the decision of whether or not view, purchase or download the item. Taken together, these examples serve to motivate our standpoint that ranking-oriented approaches have high potential and that combining rating-oriented and ranking-oriented approaches holds promise for designing more successful recommendation algorithms.

Another source of motivation derives from the recent recommender system literature, which demonstrates a growing awareness that under ranking-oriented recommendation, the ability of the system to predict ratings is also important. This awareness is based on the insight that although users find it important to receive a high quality ranked list from the recommender system, the list will be less useful or less acceptable to the user if the ratings assigned by the system to the items fail to approximate those that the user would have assigned. The increasing emphasis on providing the user with both a high quality ranked list and accurate ratings is reflected in the recent adoption of the Normalized Discounted Cumulative Gain (NDCG) evaluation metric [14,15,30,31]. As discussed in more detail in Section 4.2, NDCG simultaneously takes into account both the rank ordering of a list as well as the graded relevance, i.e., the magnitude of the scores of the items in the list. Somewhat unexpectedly, although recommender system research is increasingly taking both rank and rating prediction into account for evaluation, up until this point, no concerted research effort has been devoted to developing algorithms that produce recommendation lists that simultaneously optimize both rank and ratings of the recommended items. The contribution of this paper is to combine the two types of recommendation, ranking-oriented and rating-oriented, in order to arrive at a system that generates recommendations that are more completely suited to satisfy user needs.

We accomplish the goal of generating recommendations optimized not only for ranking, but also for rating by proposing a novel unified recommendation model (URM) that enhances ranking-oriented recommendation using a rating-oriented approach. The model combines probabilistic matrix factorization (PMF) [22], i.e., rating-oriented CF, and ListRank [27], i.e., ranking-oriented CF, by exploiting common latent features shared by both PMF and ListRank. In fact, by incorporating PMF we enable ListRank to benefit from rating predictions, which contributes another basis for generating the recommendation list. We demonstrate experimentally that the URM achieves significant improvement of recommendation performance over the state-of-the-art CF approaches on various data sets. Furthermore, we analyze and empirically demonstrate that URM maintains linear complexity with the number of observed ratings in the given user-item matrix, which means that it can scale up with the increasing amount of data.

The approach presented in this paper builds on and expands the basic finding of the effectiveness of list-wise learning-to-rank, demonstrated in [27], where we first introduced ListRank, a ranking-oriented matrix factorization approach. The expansions that are presented here extend along two dimensions. First, we combine the advantages of ranking-oriented and rating-oriented recommendation by combining ListRank with a rating-oriented component, resulting in URM, a new recommendation model. Second, we conduct experimental evaluations on multiple datasets of various scales to validate the usefulness of the proposed URM approach, and demonstrate its specific contributions to the state of the art.

The remainder of the paper is structured as follows. In the next section, we summarize related work and position our approach with respect to it. Then, we present the URM and validate it experimentally. Finally, we sum up the key aspects of URM and address possible directions for future work.

2. Related work

Our work builds on the foundation of the large body of work that has been carried out on CF. CF approaches are generally considered to fall into one of two categories, i.e., memory-based CF and model-based CF [1,6]. In general, memory-based CF uses similarities between users (user-based CF) or similarities between items (item-based CF) to make recommendations. User-based CF [7,21] recommends items to a user on the basis of how well similar users like those items. Item-based CF [5,13,23] recommends items to a user based on the similarity between the user's favored items and the items to be recommended. Recently, various studies have been devoted to the modification and enhancement of memory-based CF, e.g., to specifically improve user-based CF [26,33], to specifically improve item-based CF [32], and to combine user-based CF and item-based CF [16,29]. Although substantial improvements have been achieved, memory-based CF approaches still suffer from high computational complexity, i.e., computing similarities among the typically enormous number of users or items in recommender system applications is expensive.

In comparison, model-based CF approaches first fit prediction models based on training data and then use the model to predict users' preferences on items. These models include latent semantic models [9], mixture models [11,28] and fuzzy linguistic models [17,19]. Matrix factorization (MF) [12,22] has been recognized as one of the most successful model-based CF approaches, due to its superior accuracy and scalability. Generally, MF models learn low-rank representations (latent features) of users and items from the observed ratings in the user-item matrix, which are further used to predict unobserved

ratings. MF can also be formulated from a probabilistic perspective, i.e., PMF [22], which models the conditional probability of latent features given the observed ratings, and factors for complexity regularization encoding prior information on user and item ratings. In this paper, we adopt PMF as the rating-oriented CF component of our proposed URM.

Compared to the large volume of research on rating-oriented CF, the research on ranking-oriented CF is limited. The first mature ranking-oriented CF approach is *CofiRank* [30,31], which introduces structured ranking losses and various other extensions to MF. Further studies mainly focus on exploiting pair-wise preference between items for users, e.g., EigenRank [14], probabilistic latent preference analysis [15] and Bayesian personalized ranking [20]. Although empirical study has shown the great value of pair-wise comparisons other than individual ratings for representing users' preferences [10], all these existing pair-wise approaches [14,15,20] require deriving pair-wise training examples from individual ratings, thus, in general all suffer from high computational complexity of pair-wise comparisons, which scale quadratically to the number of rated items in a given data collection. In contrast, ListRank [27] is designed to incorporate a list-wise learning-to-rank concept with MF, which is characterized by a low complexity, i.e., complexity is linear with the number of the observed ratings in a given user-item rating matrix. Preliminary experiments [27] also show ListRank to be competitive for recommendation in comparison to other state-of-the-art approaches, represented by CofiRank. One of the latest contributions on exploiting other learning-to-rank methods for CF [2] shares the same motivation of ListRank, and also envisioned the potential of list-wise approach for CF, which is represented by ListRank. The established performance and value of ListRank makes it a natural choice as our ranking-oriented approach, to be extended within the proposed URM.

Our work in this paper unifies a rating-oriented CF, i.e., PMF and a ranking-oriented CF, i.e., ListRank in terms of the same latent features shared by PMF and ListRank. In view of the comparison of ranking-oriented and rating-oriented CF in the previous section, and also considering the target of generating a ranked list of recommendations for the user, we chose the ranking-oriented approach as the basis of our unified recommendation model (URM) and deploy rating-oriented PMF to expand it.

3. Unified recommendation model

In this section, we first briefly present the basic formulation of PMF and ListRank. Then, we combine PMF and ListRank by means of the URM and, finally, we present an efficient learning algorithm for solving the optimization problem in the URM and analyze the complexity of the algorithm.

3.1. PMF: matrix factorization for rating

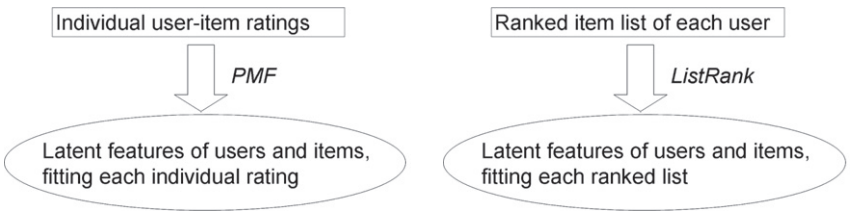
If we denote by R a user-item rating matrix consisting of M users ratings on N items, PMF [22] seeks to represent the matrix R by two low-rank matrices, U and V. A d-dimensional set of latent features is used to represent both users (in U) and items (in V). Note that we use U_i to denote a d-dimensional column feature vector of user i, V_j to denote a d-dimensional column feature vector of item j, and R_{ij} to denote the user i's rating on item j. Usually, the rating scale is different from one dataset (application scenario) to another. To achieve generality, the ratings are normalized to the range from [0,1]. The objective of PMF is now to fit each rating R_{ij} with the corresponding inner product $U_i^T V_j$, which can be formulated as follows:

$$U, V = \arg\min_{U,V} \left\{ \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{N} I_{ij} \left(R_{ij} - g(U_i^T V_j) \right)^2 + \frac{\lambda_U}{2} \|U\|_F^2 + \frac{\lambda_V}{2} \|V\|_F^2 \right\} \tag{1}$$

Here, I_{ij} is an indicator function that is equal to 1 when $R_{ij} > 0$, and 0 otherwise. The parameters λ_U and λ_V are regularization coefficients used to reduce over-fitting, while $\|U\|_F$ and $\|V\|_F$ are the Frobenius norms of the matrices U and V. For simplicity, we set $\lambda_U = \lambda_V = \lambda$. The $g(x)$ is a logistic function serving to bound the range of $U_i^T V_j$ to be also in the range [0, 1], i.e., $g(x) = 1/(1 + e^{-x})$. The input–output diagram of PMF is illustrated in Fig. 1.

3.2. ListRank: matrix factorization for ranking

In order to model the user's preference from her ranked list of rated items, we need to transform the user's ratings on different items to ranking scores, which are required to maintain two properties. First, for a given user, the ranking score



(a) PMF (b) ListRank

Fig. 1. The input–output diagrams of PMF and ListRank.

of item i should be higher than (or lower than, or equal to) item j, if she rates item i higher than (or lower than, or equally to) item j. Second, the ranking scores of all the users should share the same scale/space. For this reason, we exploit the top one probability [18] for the transformation from ratings of each user to ranking scores. From the probabilistic point, the top one probability indicates the probability of a graded item being ranked in the top position from all the graded items. Note that top one probability and its similar variants are usually used to map graded scores into a probability space in the literature [3,4]. Specifically, the top one probability (the ranking score) for item j that is rated R_{ij} by user i can be expressed as:

$$p(R_{ij}) = \frac{\exp(R_{ij})}{\sum_{k=1}^{N} \exp(R_{ik})} \tag{2}$$

in which $\exp(x)$ denotes the exponential function of x.

As opposed to PMF that aims at reproducing and extrapolating the ratings from R, the ListRank [27] has the objective to fit each user's ranked list of items with a factorization model. A regularized loss function that models the cross-entropy of top-one probabilities of the items in the training ranked item lists and the lists from the factorization model can be formulated as follows:

$$\begin{aligned} L(U,V) &= \sum_{i=1}^{M} \left\{ -\sum_{j=1}^{N} I_{ij} p(R_{ij}) \log p\left(g(U_i^T V_j)\right) \right\} + \frac{\lambda}{2}\left(\|U\|_F^2 + \|V\|_F^2\right) \\ &= \sum_{i=1}^{M} \left\{ -\sum_{j=1}^{N} I_{ij} \frac{\exp(R_{ij})}{\sum_{k=1}^{N} I_{ik} \exp(R_{ik})} \log \frac{\exp\left(g(U_i^T V_j)\right)}{\sum_{k=1}^{N} I_{ik} \exp\left(g(U_i^T V_j)\right)} \right\} + \frac{\lambda}{2}\left(\|U\|_F^2 + \|V\|_F^2\right) \end{aligned} \tag{3}$$

Note that in ListRank we also adopt the same simplification strategy as used in PMF (see Section 3.1), i.e., setting an equal regularization parameter λ for penalizing the magnitudes of both U and V. While the training lists are derived from the profiles of the users, the loss function reflects the uncertainty in predicting the output lists from the factorization model using the training lists. Note that minimizing the regularized loss function 3 results in a factorization model, i.e., U and V that is not optimized for rating prediction, but for ranking positions of items in the users lists. This key difference between ListRank and PMF is also shown in Fig. 1.

3.3. Combining PMF and ListRank

As introduced above, PMF and ListRank learn the latent features of users and items by taking different views on the known data, i.e., PMF exploiting the individual ratings, and ListRank exploiting the ranked lists. Our motivation of URM is then straightforward so that the two different views can be exploited simultaneously, by which the knowledge encoded in individual ratings is expected to improve the latent features of users and items from ListRank to achieve better ranking performance, as the example mentioned in Section 1. The illustration diagram of URM is shown in Fig. 2. Since both the PMF and the ListRank are based on matrix factorization, we link the two by imposing common latent features for both models. Then, the URM can be formulated by means of a new regularized loss function $F(U,V)$ as follows:

$$\begin{aligned} F(U,V) = \alpha \times \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(I_{ij} \left(R_{ij} - g(U_i^T V_j) \right)^2 \right) + (1-\alpha) \times \sum_{i=1}^{M} \left\{ -\sum_{j=1}^{N} I_{ij} \frac{\exp(R_{ij})}{\sum_{k=1}^{N} I_{ik} \exp(R_{ik})} \log \frac{\exp\left(g(U_i^T V_j)\right)}{\sum_{k=1}^{N} I_{ik} \exp\left(g(U_i^T V_j)\right)} \right\} \\ + \frac{\lambda}{2}\left(\|U\|_F^2 + \|V\|_F^2\right) \end{aligned} \tag{4}$$

The trade-off parameter α is used to control the relative contribution from PMF and ListRank. As stated in the introduction, we bias the loss function towards ranking. Consequently, the value of α should be relatively small. We justify this choice in Section 4.3, where we experimentally investigate the impact of α on the recommendation performance. Minimizing the loss function Eq. (4) results in the matrices U and V that are not only optimized for item ranking, but also enhanced by the information used to predict each item's rating. This result can be used to produce a ranked recommended item list for each user i

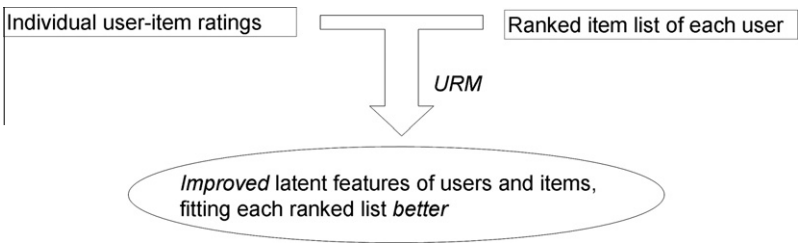


Fig. 2. The input–output diagram of URM.

and is generated by ordering items in the collection in the descending order according to the value $U_i^T V$. Note that items already rated by a user will be removed from the list.

3.4. Learning algorithm and complexity analysis

Since the loss function (4) is not convex jointly over U and V, we choose to deploy a gradient descent method by alternatively fixing U and V and searching for local minima. The gradients of $F(U,V)$ with respect to U and V can be computed as:

$$\frac{\partial F}{\partial U_i} = \alpha \sum_{j=1}^{N} I_{ij}\left(g(U_i^T V_j) - R_{ij}\right) g'(U_i^T V_j) V_j + (1-\alpha)\sum_{j=1}^{N} I_{ij}\left(\frac{\exp\left(g(U_i^T V_j)\right)}{\sum_{k=1}^{N} I_{ik}\exp\left(g(U_i^T V_j)\right)} - \frac{\exp(R_{ij})}{\sum_{k=1}^{N} I_{ik}\exp(R_{ik})}\right) g'(U_i^T V_j) V_j + \lambda U_i \tag{5}$$

$$\frac{\partial F}{\partial V_j} = \alpha \sum_{i=1}^{M} I_{ij}\left(g(U_i^T V_j) - R_{ij}\right) g'(U_i^T V_j) U_i + (1-\alpha)\sum_{i=1}^{M} I_{ij}\left(\frac{\exp\left(g(U_i^T V_j)\right)}{\sum_{k=1}^{N} I_{ik}\exp\left(g(U_i^T V_j)\right)} - \frac{\exp(R_{ij})}{\sum_{k=1}^{N} I_{ik}\exp(R_{ik})}\right) g'(U_i^T V_j) U_i + \lambda V_j \tag{6}$$

where $g'(x)$ denotes the derivative of $g(x)$. An overview of the algorithm deploying Eq. (5) and (6) for solving the minimization problem in the URM is given in Algorithm 1. The stopping parameter ϵ is used to indicate the desired level of the convergence of the algorithm. In our experiments, the value of ϵ is set to 0.01. Our experiments showed that the algorithm usually converges after no more than 200 iterations. Unlike the constant learning step size η as used for ListRank [27], we allow η in the URM to be as large as possible (maximally 1) in each iteration, as long as it leads to a decrease in the loss funcion Eq. (4). Setting η in this flexible way helps to speed up the convergence of the algorithm.

Algorithm 1. Learning algorithm for URM

Input: training data R, tradeoff parameter α, regularization parameter λ, stopping threshold ϵ.
Output: Complete user-item relevance matrix $\hat{R}$.
Initialize $U^{(0)}, V^{(0)}$ with random values;
Initialize f_1 with a large value and f_2 a small value;
$t = 0$;
repeat
  $f_1 = F(U^{(t)}, V^{(t)})$;
  $\eta = 1$;
  Compute $\frac{\partial F}{\partial U^{(t)}}, \frac{\partial F}{\partial V^{(t)}}$ as in Eq. (5) and (6);
repeat
    $\eta = \eta/2$
until $F(U^{(t)} - \eta\frac{\partial F}{\partial U^{(t)}}, V^{(t)} - \eta\frac{\partial F}{\partial V^{(t)}}) < f_1$;
  $U^{(t+1)} = U^{(t)} - \eta\frac{\partial F}{\partial U^{(t)}}, V^{(t+1)} = V^{(t)} - \eta\frac{\partial F}{\partial V^{(t)}}$;
  $f_2 = F(U^{(t+1)}, V^{(t+1)})$;
  $t = t + 1$;
until $f_1 - f_2 \leqslant \epsilon$;
$\hat{R} = U^{(t)T} V^{(t)}$;

It can be easily shown that the complexity of the loss function for URM is in the order of $O(dS + d(M+N))$, where S denotes the number of observed ratings in a given user-item matrix and where d is the dimensionality of latent features. The complexity of the gradients in Eq. (5) and (6) is of the order $O(dS + dM)$ and $O(dS + pdS + dN)$, respectively, where p denotes the average number of items rated per user and usually is substantially smaller than S. Considering we also often have $S \gg M, N$, the total complexity in one iteration has the order of $O(dS)$, which is linear with the number of observed ratings in the matrix. This analysis indicates the computational efficiency and scalability of URM. This will also be illustrated quantitatively in Section 4.4.

4. Experiments and evaluation

In this section we present a series of experiments that evaluate the proposed URM. We first give a detailed description of the setup of our experiments. Then, we investigate the impact of tradeoff parameters in URM and demonstrate the

effectiveness and efficiency of URM. Finally, we compare the recommendation performance of URM with some other baseline and state-of-the-art approaches.

We designed the experiments in order to be able to answer the following research questions:

1. Could URM as a combination of a rating-oriented and a ranking-oriented CF approach outperform each of the individual approaches? (Section 4.3 and 4.5)
2. Does the recommendation performance increase with the minimization of the loss function Eq. (4)? (Section 4.4)
3. How efficient and scalable is URM? (Section 4.4)
4. How does URM compare to alternative state-of-the-art approaches across different data sets and across users with different profiles? (Section 4.5)

4.1. Datasets

Our experiments are conducted on three publicly available datasets, i.e., two datasets from MovieLens,[1] and the EachMovie[2] dataset. All of them are widely used in the field of recommender systems. The first MovieLens dataset [7], denoted as ML1, contains 100 K ratings (scale 1–5) from 943 users on 1682 movies. The second MovieLens data set, denoted as ML2, contains 1 M ratings (scale 1–5) from ca. 6 K users on ca. 3.7 K movies. Each user in both ML1 and ML2 has rated at least 20 movies. The EachMovie dataset contains ca. 2.8 M ratings (scale 1–6) from ca. 61 K users on ca. 1.6 K movies. Note that in all of the used datasets we excluded the items (i.e., movies) that are never rated. Thus, the aforementioned statistics of the datasets may be slightly different from those in other literature. Some detailed statistics of the datasets are summarized in Table 1.

4.2. Experimental setup and evaluation metrics

We choose to conduct our experiments following a standard protocol as widely used in related work [27,30,31]. Note that our experimental protocol is designed to demonstrate the effectiveness of URM under different conditions of user profiles. We create variants of the datasets in order to test experimental conditions involving three different user profile lengths (UPLs), i.e., 10, 20 and 50. For example, in the case of UPL = 10, we randomly select 10 rated items for each user for training, and use the remaining user ratings for testing. Per UPL, users with less than 20, 30, or 60 rated items are removed in order to ensure we can evaluate on at least 10 rated items per user. For each UPL, we create 10 different versions of the dataset by sampling the user profiles to arrive at the targeted number of items in the training set. Note that in the case of UPL = 50 for each dataset, we create an additional version that is used as a validation set to tune the tradeoff parameter and investigate the impact of this parameter as shown in Section 4.3. The data from the validation sets have not been used for the test runs, which are used to evaluate the algorithm. We report the average performance attained across all users and 10 test runs in Section 4.5.

Following the standard evaluation strategy applied to recommender systems [14,15,30,31], we measure the recommendation performance only based on the rated items from each user. We consider the performance of a recommender algorithm to be good if it ranks items with high ratings in the test set to higher positions in the ranked list than those having low ratings. The algorithm should also emphasize the accuracy of highly ranked items, since users usually expect highly relevant items to be recommended as early as possible. The evaluation metric Normalized Discounted Cumulative Gain (NDCG) satisfies the two requirements and is widely used in recommender systems research [14,15,30,31]. Note that since we are not interested in rating prediction performance, metrics, such as mean average error (MAE), root mean square error (RMSE), are not considered. Also notice that since the datasets in our experiments contain graded relevance, NDCG should be more appropriate than other metrics, such as precision, recall, mean average precision (MAP), for which artificial thresholds need to be assumed to convert graded relevance to binary case. For those reasons, NDCG could be the best choice among all the metrics for our experimental evaluation. The definition of NDCG at the top-K ranked items for a user u can be given as:

$$NDCG_u@K = Z_u \sum_{k=1}^{K} \frac{2^{Y_u(k)} - 1}{\log_2(1+k)} \tag{7}$$

Here, $Y_u(k)$ denotes the grade of relevance of the item that is ranked in the kth position for user u. Note that in this setting the rating is regarded as the grade of relevance. Z_u is a normalization factor securing that the perfect ranking list will have $NDCG_u@K$ equal to 1. In other words, $1/Z_u$ is equal to $NDCG_u@K$ when the ranked list is created by sorting the ground truth items of the users in the test set in descending order by their ratings. In this paper, we report the recommendation performance by NDCG@5 and NDCG@10, which are averaged across all users.

We did not formally tune the dimensionality d of latent features and the regularization parameter λ for the URM in the experiments. The dimensionality d is set independent of the user-item matrix, and usually a small value of d is sufficient for acceptable recommendation performance [31]. In this work, we fix d as 10, which we adopted from the recently proposed CofiRank approach [31]. The regularization parameter λ is usually set large enough to avoid over-fitting, as demonstrated in

[1] http://www.grouplens.org/node/73.
[2] http://kumpf.org/eachtoeach/eachmovie.html.

Table 1
Statistics of datasets used in the experiments.

	# Users	# Items	# Ratings	Sparseness (%)	Scale	Ave. # ratings/user	Ave. rating
ML1	943	1682	100,000	93.7	1–5	106.0	3.53
ML2	6040	3706	1,000,209	95.5	1–5	165.6	3.58
EM	61,265	1623	2,811,718	97.2	1–6	45.9	4.04

ListRank [27]. We fix λ as 0.1 for all the experiments on different data sets, a setting from which we did not observe overfitting.

4.3. Impact of tradeoff parameter

In this subsection we investigate the impact of tradeoff parameter α on the performance of the proposed URM. For each dataset, we conduct an experiment on a validation set under the condition of UPL = 50, i.e., in the validation set we randomly select 50 rated items for each user for training and use the remaining rated items for testing. By varying the tradeoff parameter in the URM, we can evaluate its influence on the recommendation performance, i.e., NDCG@10 here, as shown in Fig. 3. Note that the URM is equivalent to ListRank if the tradeoff parameter $\alpha = 0$ and to PMF if $\alpha = 1$. The diagrams in Fig. 3 indicate that different optimal values of tradeoff parameters can be selected for different datasets. Selecting this optimal value per dataset leads to an improvement in the recommendation performance compared to either ListRank or PMF taken individually. This observation suggests the promise of combining rating-oriented and ranking-oriented approaches, providing initial evidence that our first research question can be answered positively. Additional experiments in Section 4.5 make further contribution to this issue. Furthermore, it can also be observed that the optimal tradeoff parameter in each dataset is below 0.5, which means that the major contribution to the recommendation performance comes from the ranking-oriented CF. This observation confirms the achievements by the recent progress in ranking-oriented CF approaches, e.g., [14,15,30], which usually outperform rating-oriented CF approaches, and also justifies our choice to bias the URM towards ranking prediction, as stated in Section 3.3. The optimal tradeoff parameters obtained from analyzing the validation sets are used subsequently on the three datasets for the test runs in all test cases as reported in Section 4.5.

4.4. Effectiveness and efficiency

In this subsection, we investigate whether minimizing the loss function of URM in Eq. (4) indeed leads to an increase in recommendation performance, and whether the proposed URM is empirically an efficient algorithm. These experiments

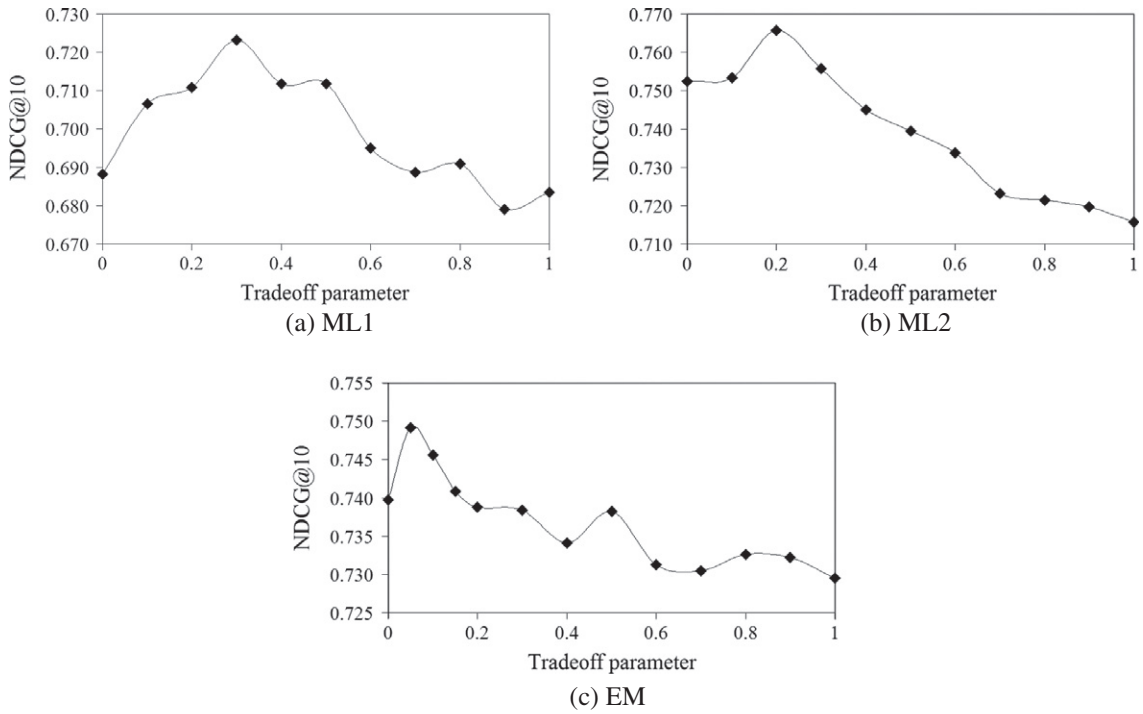


Fig. 3. Impact of the tradeoff parameter in URM on NDCG@10.

were also conducted on the validation sets. We adopt the optimal tradeoff parameters obtained from previous subsection for this investigation. The diagrams in Fig. 4 demonstrate the development of the loss function and NDCG@10 during the iterations of the minimization process. Here, we normalized loss for the demonstration purposes. We can see that the NDCG@10 grows steadily and converges in all of the datasets in parallel with the loss function being minimized. These observations indicate that the approach proposed in this paper is effective in achieving improved recommendation performance, thus addressing our second research question and allowing us to give it a positive answer. Additionally, they provide evidence that URM is indeed a model of the phenomenon that it was designed to capture.

Furthermore, we can also observe in Fig. 4 that the NDCG@10 already becomes close to optimal after 10 iterations on ML1 dataset, after 50 iterations on ML2 dataset and EM dataset. This observation indicates that the proposed URM is efficient in reaching convergence, even for a large scale data set.

We also demonstrate the relationship of average iteration time against the scale (i.e., the number of ratings) of each data set, as shown in Fig. 5. Note that for the smallest dataset ML1 one iteration only takes around 1 s, and for the largest data set (EM) one iteration only takes around 20 s in our MATLAB implementation on a PC with 1.59 GHz CPU and 2.93 GB memory. Moreover, the runtime of a single iteration increases almost linearly with the increase of the data scale, which empirically verifies that the URM can easily scale up to address datasets of any size. The conclusions from this section lead to an answer to our third research question, namely, they demonstrate the efficiency and scalability of URM.

4.5. Performance comparison

In this subsection, we compare the performance of URM and a number of representative alternative CF approaches that we list and briefly describe below. Our selection of alternative approaches covers various aspects in recommendation area, including a non-personalized approach, a widely-used memory-based approach and three state-of-the-art model-based approaches.

- **ItemAvRat**: This is a naive non-personalized recommendation approach, that recommends items to users according to the average item rating. In other words, the item that has the highest average rating in the training data will be the top recommended item for every user. Using this method, every user will get offered the same recommendation list.
- **ItemCF**: This is a traditional and widely used item-based CF approach [5,13,23]. Our implementation of ItemCF is based on [5].
- **PMF**: This is a state-of-the-art rating-oriented CF approach [22], which is equivalent to the proposed URM when α is set to 1. Note that we use the same dimensionality of latent features and regularization parameter as used in URM.

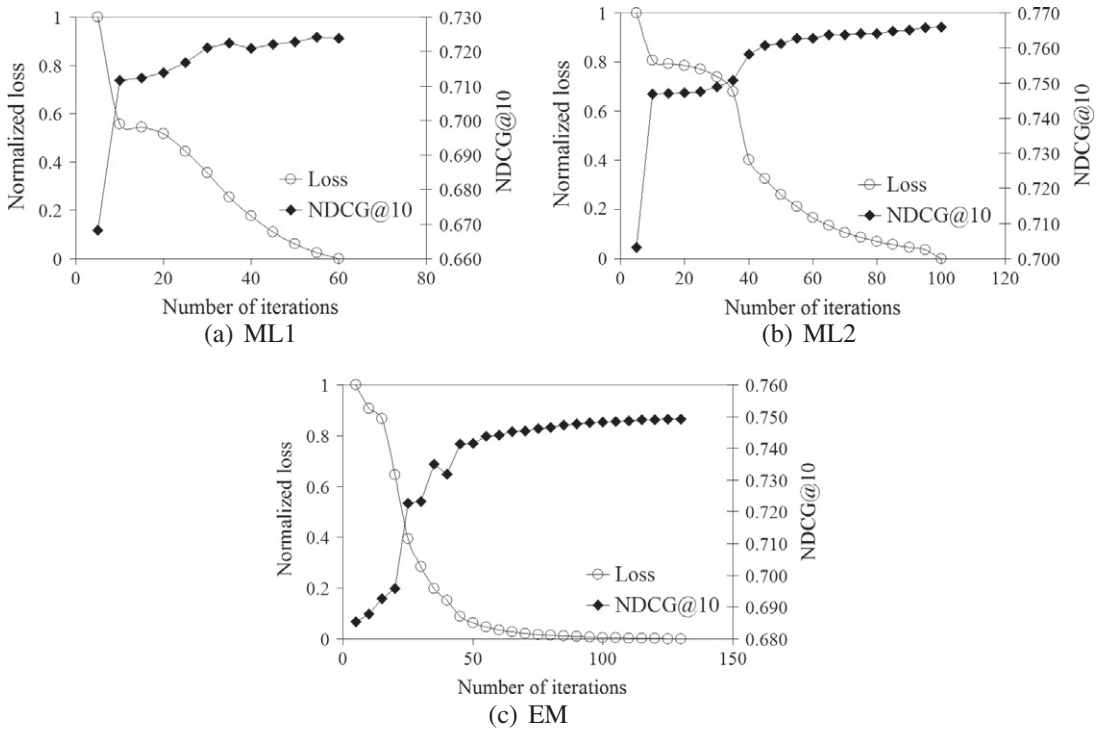


Fig. 4. The variation of NDCG@10 and the loss in URM during the minimization.

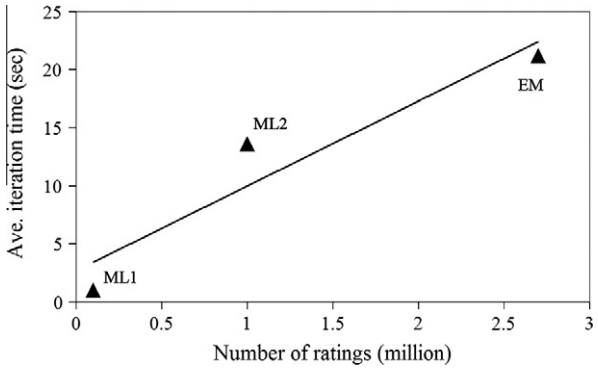


Fig. 5. The relationship between the average iteration time in the URM and the scale of the data.

Table 2
Performance comparison in terms of NDCG between URM and other recommendation approaches on ML1 dataset. According to Wilcoxon signed rank significance test with $p < 0.05$, we use † to denote the significant improvement over ListRank, and $*$ to denote the significant improvement over all the other approaches except ListRank.

	UPL = 10		UPL = 20		UPL = 50	
	NDCG@5	NDCG@10	NDCG@5	NDCG@10	NDCG@5	NDCG@10
ItemAvRat	0.345 ± 0.005	0.400 ± 0.004	0.313 ± 0.008	0.357 ± 0.006	0.274 ± 0.004	0.309 ± 0.002
ItemCF	0.552 ± 0.003	0.578 ± 0.002	0.556 ± 0.004	0.580 ± 0.003	0.546 ± 0.008	0.571 ± 0.007
PMF	0.603 ± 0.007	0.630 ± 0.004	0.588 ± 0.013	0.610 ± 0.010	0.597 ± 0.014	0.616 ± 0.011
CofiRank	0.600 ± 0.007	0.678 ± 0.000	0.633 ± 0.005	0.681 ± 0.000	0.664 ± 0.007	0.701 ± 0.000
ListRank	0.672 ± 0.005	0.693 ± 0.005	0.682 ± 0.006	0.691 ± 0.004	0.687 ± 0.008	0.684 ± 0.006
URM	**0.673 ± 0.003***	**0.694 ± 0.003***	**0.699 ± 0.005***†	**0.708 ± 0.003***†	**0.717 ± 0.009***†	**0.718 ± 0.007***†

- **ListRank**: This is a state-of-the-art ranking-oriented CF approach [27], which is equivalent to the proposed URM when α is set to 0. Note that we also use the same dimensionality of latent features and regularization parameter as used in URM.
- **CofiRank**: This is another state-of-the-art ranking-oriented CF approach. We implemented it using publicly available software.[3] Regarding the parameter setting, we adopted the optimal values of most parameters from [31], and we tuned the rest of them for optimal performance using the same validation sets as for URM. Since our experimental setting is exactly same as the work of CofiRank and its extensions [31], we can compare our results directly to the best of theirs among various parameter settings if available, i.e., the CofiRank performance of NDCG@10 on ML1 and EM data sets in Tables 2 and 4 are directly adopted from [31].

The performance of different approaches with respect to different user profile length (UPL) is shown in Tables 2–4. For each dataset and each UPL we repeat experiments 10 times, i.e., with 10 random splits of training and testing data as described in Section 4.2. As can be seen from Table 2, URM outperforms other approaches significantly in most of the cases on ML1 dataset, according to Wilcoxon signed rank significance test with $p < 0.05$. Note that we use † to denote the significant improvement over ListRank, and * to denote the significant improvement over all the other approaches except ListRank. For the results directly available from CofiRank [31], we did not conduct the significance test for the comparison with the corresponding results from URM, since we do not have the results of CofiRank in each run. The URM achieves large amount of improvement (ca. 20%) over the naive approach ItemAvRat and the traditional CF approach ItemCF, and over 10% improvement over PMF. Compared to the state-of-the-art CofiRank, it also achieves ca. 3–10% improvement. Note that these improvements are consistent across different user profiles, i.e., different conditions of UPL. We can also observe that URM significantly improves upon ListRank by ca. 2–5% in the cases of UPL as 20 and 50. Although the improvement over ListRank in the case of UPL as 10 is not statistically significant, we emphasize that the tradeoff parameter used in the testing runs is based on the validation set, which is formed in the condition of UPL = 50. In practice, we could tune the tradeoff parameter more tightly by considering the targeting user profile length in order to attain further performance gain. In this paper, we only tune tradeoff parameter based on a certain condition of UPL, which allows us to show that the tuned tradeoff parameter could be robust enough to be applied to other conditions of UPL.

For the performance of the URM on ML2 and EM datasets, which are much larger than ML1, similar observations can be found, as shown in Table 3 and Table 4. Note that on these datasets URM achieves significant improvement over all the other

[3] http://www.cofirank.org/downloads.

Table 3
Performance comparison in terms of NDCG between URM and other recommendation approaches on ML2 dataset

	UPL = 10		UPL = 20		UPL = 50	
	NDCG@5	NDCG@10	NDCG@5	NDCG@10	NDCG@5	NDCG@10
ItemAvRat	0.297 ± 0.002	0.342 ± 0.001	0.280 ± 0.002	0.322 ± 0.001	0.255 ± 0.001	0.293 ± 0.001
ItemCF	0.594 ± 0.002	0.589 ± 0.002	0.603 ± 0.002	0.616 ± 0.002	0.589 ± 0.002	0.607 ± 0.001
PMF	0.645 ± 0.001	0.653 ± 0.001	0.644 ± 0.004	0.653 ± 0.003	0.680 ± 0.005	0.686 ± 0.004
CofiRank	0.671 ± 0.002	0.668 ± 0.002	0.694 ± 0.002	0.689 ± 0.002	0.693 ± 0.002	0.692 ± 0.002
ListRank	0.647 ± 0.002	0.654 ± 0.002	0.683 ± 0.003	0.688 ± 0.003	0.751 ± 0.002	0.751 ± 0.002
URM	**0.732 ± 0.004***†	**0.735 ± 0.002***†	**0.748 ± 0.003***†	**0.747 ± 0.002***†	**0.764 ± 0.009***†	**0.760 ± 0.009***†

Table 4
Performance comparison in terms of NDCG between URM and other recommendation approaches on EM dataset.

	UPL = 10		UPL = 20		UPL = 50	
	NDCG@5	NDCG@10	NDCG@5	NDCG@10	NDCG@5	NDCG@10
ItemAvRat	0.236 ± 0.000	0.307 ± 0.000	0.222 ± 0.001	0.291 ± 0.000	0.194 ± 0.000	0.255 ± 0.000
ItemCF	0.534 ± 0.001	0.579 ± 0.001	0.545 ± 0.001	0.592 ± 0.001	0.552 ± 0.002	0.598 ± 0.001
PMF	0.608 ± 0.004	0.643 ± 0.003	0.606 ± 0.003	0.646 ± 0.002	0.690 ± 0.003	0.714 ± 0.002
CofiRank	0.639 ± 0.003	0.646 ± 0.000	0.671 ± 0.002	0.653 ± 0.000	0.641 ± 0.001	0.647 ± 0.000
ListRank	0.567 ± 0.002	0.607 ± 0.002	0.642 ± 0.003	0.674 ± 0.003	0.721 ± 0.002	0.740 ± 0.002
URM	**0.668 ± 0.003***†	**0.695 ± 0.002***†	**0.707 ± 0.003***†	**0.726 ± 0.003***†	**0.735 ± 0.001***†	**0.747 ± 0.001***†

approaches in all the conditions of UPL. Compared to the second best approach in each case, the improvement attained by the URM is of ca. 2–10%. These results allow us to give a positive answer to our first research question, namely, they show that URM could improve recommendation performance over state-of-the-art approaches across different datasets and for users with different profiles. They also make it possible to give a positive answer to our fourth and final research question: Regarding the comparison of URM with other state-of-the-art approaches, the performance of URM is clearly and consistently superior.

5. Conclusion and future work

In this paper, we present a novel recommendation approach URM, which is capable of unifying a ranking-oriented CF approach ListRank and a rating-oriented CF approach PMF by exploiting common latent features of users and items. We qualitatively and quantitatively demonstrate that the complexity of URM is linear with the number of observed ratings in a given user-item matrix, indicating that URM can be deployed in large-scale use cases. We also experimentally verify that the recommendation performance of URM mainly derives from the ranking-oriented component, i.e., ListRank, while the rating-oriented component, i.e., PMF, contributes significant enhancement. Our experimental results indicate that URM substantially outperforms both component approaches, i.e., ListRank and PMF, and other traditional and state-of-the-art recommendation approaches. Performance improvements achieved by URM are also shown to be consistent with respect to various datasets and users with various profile lengths.

Moving forward, future work in this area will explore three interesting directions. First, we are interested in investigating other options of item-list representation, which might influence the performance of the ranking-oriented recommendation approach, thus, improve the performance of URM. Second, in this paper we established that the latent space can mediate between the rating-oriented approach and the ranking-oriented approach. We are interested in exploring the shared latent space to integrate in the framework of URM with other types of information, e.g., item content features, contextual information of users and items. Third, we are also interested in investigating the relationship between URM and the latest proposed recommendation models that directly optimizes ranking measures, such as mean average precision [24] and mean reciprocal rank [25].

References

[1] 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 (6) (2005) 734–749.
[2] S. Balakrishnan, S. Chopra, Collaborative ranking, in: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, WSDM '12, ACM, New York, NY, USA, 2012, pp. 143–152.
[3] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, H. Li, Learning to Rank: From Pairwise Approach to Listwise Approach, Tech. Rep. MSR-TR-2007-40, Microsoft Research, 2007.
[4] O. Chapelle, D. Metlzer, Y. Zhang, P. Grinspan, Expected reciprocal rank for graded relevance, in: Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM '09, ACM, New York, NY, USA, 2009, pp. 621–630.
[5] M. Deshpande, G. Karypis, Item-based top-n recommendation algorithms, ACM Transactions on Information Systems 22 (2004) 143–177.

[6] M.D. Ekstrand, J.T. Riedl, J.A. Konstan, Collaborative filtering recommender systems, Foundations and Trends in Human–Computer Interaction 4 (2) (2011) 81–173.
[7] J.L. Herlocker, J.A. Konstan, A. Borchers, J. Riedl, An algorithmic framework for performing collaborative filtering, in: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '99, ACM, New York, NY, USA, 1999, pp. 230–237.
[8] J.L. Herlocker, J.A. Konstan, L.G. Terveen, J.T. Riedl, Evaluating collaborative filtering recommender systems, ACM Transactions on Information Systems 22 (2004) 5–53.
[9] T. Hofmann, Latent semantic models for collaborative filtering, ACM Transactions on Information Systems 22 (2004) 89–115.
[10] N. Jones, A. Brun, A. Boyer, Comparisons instead of ratings: towards more stable preferences, in: Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, WI-IAT '11, vol. 01, IEEE Computer Society, Washington, DC, USA, 2011, pp. 451–456.
[11] J. Kleinberg, M. Sandler, Using mixture models for collaborative filtering, Journal of Computer and System Sciences 74 (2008) 49–69.
[12] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (2009) 30–37.
[13] G. Linden, B. Smith, J. York, Amazon.com recommendations: item-to-item collaborative filtering, IEEE Internet Computing 7 (2003) 76–80.
[14] N.N. Liu, Q. Yang, Eigenrank: a ranking-oriented approach to collaborative filtering, in: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '08, ACM, New York, NY, USA, 2008, pp. 83–90.
[15] N.N. Liu, M. Zhao, Q. Yang, Probabilistic latent preference analysis for collaborative filtering, in: Proceeding of the 18th ACM Conference on Information and Knowledge Management, CIKM '09, ACM, New York, NY, USA, 2009, pp. 759–766.
[16] H. Ma, I. King, M.R. Lyu, Effective missing data prediction for collaborative filtering, in: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '07, ACM, New York, NY, USA, 2007, pp. 39–46.
[17] J.M. Morales-Del-Castillo, E. Peis, E. Herrera-Viedma, A filtering and recommender system prototype for scholarly users of digital libraries, in: Proceedings of the 2nd World Summit on the Knowledge Society: Visioning and Engineering the Knowledge Society, A Web Science Perspective, WSKS '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 108–117.
[18] R.L. Plackett, The analysis of permutations, Applied Statistics 24 (2) (1975) 193–202.
[19] C. Porcel, E. Herrera-Viedma, A fuzzy linguistic recommender system to disseminate the own academic resources in universities, Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, WI-IAT '09, vol. 03, IEEE Computer Society, Washington, DC, USA, 2009, pp. 179–182.
[20] S. Rendle, C. Freudenthaler, Z. Gantner, S.-T. Lars, Bpr: Bayesian personalized ranking from implicit feedback, in: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI '09, AUAI Press, Arlington, VA, United States, 2009, pp. 452–461.
[21] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, Grouplens: an open architecture for collaborative filtering of netnews, in: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, CSCW '94, ACM, New York, NY, USA, 1994, pp. 175–186.
[22] R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, in: Advances in Neural Information Processing Systems, vol. 20, 2008.
[23] 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, WWW '01, ACM, New York, NY, USA, 2001, pp. 285–295.
[24] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, N. Oliver, TFMAP: optimizing map for top-n context-aware recommendation, in: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '12, ACM, New York, NY, USA, 2012, pp. 155–164.
[25] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, A. Hanjalic, CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering, in: Proceedings of the Sixth ACM Conference on Recommender Systems, RecSys '12, ACM, New York, NY, USA, 2012, pp. 139–146.
[26] Y. Shi, M. Larson, A. Hanjalic, Exploiting user similarity based on rated-item pools for improved user-based collaborative filtering, in: Proceedings of the Third ACM Conference on Recommender Systems, RecSys '09, ACM, New York, NY, USA, 2009, pp. 125–132.
[27] Y. Shi, M. Larson, A. Hanjalic, List-wise learning to rank with matrix factorization for collaborative filtering, in: Proceedings of the Fourth ACM Conference on Recommender Systems, RecSys '10, ACM, New York, NY, USA, 2010, pp. 269–272.
[28] L. Si, R. Jin, Flexible mixture model for collaborative filtering, in: ICML, 2003, pp. 704–711.
[29] J. Wang, A.P. deVries, M.J.T. Reinders, Unifying user-based and item-based collaborative filtering approaches by similarity fusion, in: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '06, ACM, New York, NY, USA, 2006, pp. 501–508.
[30] M. Weimer, A. Karatzoglou, Q. Le, A. Smola, Cofirank – maximum margin matrix factorization for collaborative ranking, in: NIPS'07, Advances in Neural Information Processing Systems, 2007, pp. 1593–1600.
[31] M. Weimer, A. Karatzoglou, A. Smola, Improving maximum margin matrix factorization, Machine Learning 72 (2008) 263–276.
[32] H. Yildirim, M.S. Krishnamoorthy, A random walk method for alleviating the sparsity problem in collaborative filtering, in: Proceedings of the 2008 ACM Conference on Recommender Systems, RecSys '08, ACM, New York, NY, USA, 2008, pp. 131–138.
[33] J. Zhang, P. Pu, A recursive prediction algorithm for collaborative filtering recommender systems, in: Proceedings of the 2007 ACM Conference on Recommender Systems, RecSys '07, ACM, New York, NY, USA, 2007, pp. 57–64.