Transfer Learning for Behavior Ranking

WEIKE PAN, Shenzhen University
QIANG YANG, Hong Kong University of Science and Technology
YUCHAO DUAN, Shenzhen University
BEN TAN, Hong Kong University of Science and Technology
ZHONG MING, Shenzhen University

Intelligent recommendation has been well recognized as one of the major approaches to address the information overload problem in the big data era. A typical intelligent recommendation engine usually consists of three major components, that is, data as the main input, algorithms for preference learning, and system for user interaction and high-performance computation. We observe that the data (e.g., users' behavior) are usually in different forms, such as examinations (e.g., browse and collection) and ratings, where the former are often much more abundant than the latter. Although the data are in different representations, they are both related to users' true preferences and are also deemed complementary to each other for preference learning. However, very few ranking or recommendation algorithms have been developed to exploit such two types of user behavior.

In this article, we focus on jointly modeling the examination behavior and rating behavior and develop a novel and efficient ranking-oriented recommendation algorithm accordingly. First, we formally define a new recommendation problem termed *behavior ranking*, which aims to build a ranking-oriented model by exploiting both the examination behavior and rating behavior. Second, we develop a simple and generic *transfer to rank* (ToR) algorithm for behavior ranking, which transfers knowledge of candidate items from a global preference learning task to a local preference learning task. Compared with the previous work on integrating heterogeneous user behavior, our ToR algorithm is the first ranking-oriented solution, which can effectively generate recommendations in a more direct manner than those regression-oriented methods. Extensive empirical studies show that our ToR algorithm performs significantly more accurately than the state-of-the-art methods in most cases. Furthermore, our ToR algorithm is very efficient in terms of the time complexity, which is similar to those for homogeneous user behavior alone.

CCS Concepts: • **Information systems → Personalization**; • **Human-centered computing → Collaborative filtering**; • **Computing methodologies → Transfer learning;**

Additional Key Words and Phrases: Transfer to rank, behavior ranking, preference learning, collaborative recommendation

Weike Pan, Yuchao Duan, and Zhong Ming thank the National Natural Science Foundation of China (NSFC) (Nos. 61502307 and 61672358) and Natural Science Foundation of Guangdong Province (Nos. 2014A030310268 and 2016A030313038) for support. Qiang Yang and Ben Tan thank the support of China National 973 project No. 2014CB340304 and Hong Kong CERG projects Nos. 16211214, 16209715, and 16244616.
Authors' addresses: W. Pan, Y. Duan, and Z. Ming, College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China; emails: panweike@szu.edu.cn, duanyuchao@email.szu.edu.cn, mingz@szu.edu.cn; Q. Yang and B. Tan, Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China; emails: {qyang, btan}@cse.ust.hk. Q. Yang and Z. Ming are corresponding authors.

DOI: http://dx.doi.org/10.1145/3057732

ACM Reference Format:
Weike Pan, Qiang Yang, Yuchao Duan, Ben Tan, and Zhong Ming. 2017. Transfer learning for behavior ranking. ACM Trans. Intell. Syst. Technol. 8, 5, Article 65 (June 2017), 23 pages.
DOI: http://dx.doi.org/10.1145/3057732

1. INTRODUCTION

In the big data era, intelligent recommendation technology has been well recognized as one of the major approaches to the information overload problem. Recently, some research lines have switched from the classical problem of collaborative filtering with numerical ratings to a scenario with more than one type of user behavior, such as examinations and ratings [Jawaheer et al. 2014]. A typical examination behavior denotes some kind of interest of a user to an item, though the preference may not be clear and could be of some uncertainty. For example, a user may examine an item for different reasons, such as doing a comparison about the properties and price of the item with that of some other items or recording it in a memorandum for future purchase. On the contrary, a rating behavior clearly indicates a user's preference to an item, where a numerical score is usually used. Another difference is that the volume of the examination behavior is usually larger than that of the rating behavior, because the former is easier to be collected by a deployed system considering the users' little involvement. Despite the aforementioned differences between the examinations and ratings, the two types of user behavior are known to be complementary to each other [Pan et al. 2016b], because both reflect users' preferences to some extent.

For exploiting the heterogeneous user behavior in a recommendation system, there are some attempts in recent work, including singular value decomposition plus (SVD++) [Koren 2008], factorization machine (FM) [Rendle 2012], and self-transfer learning (sTL) [Pan et al. 2016b]. SVD++ integrates a user's examination behavior as an additional term when modeling an assigned rating, which is based on the assumption that users with similar examination behavior are likely to have similar tastes. FM is a generic framework that can be used to integrate different types of data, including heterogeneous user behavior, where the main technique is feature engineering that represents a user's examination behavior and a rating assigned by the user in a unified vector. The very recent algorithm sTL turns to iteratively identify some examination behavior with high probability of being preferred by a user and then integrates them in a similar way to that of SVD++. Empirically, sTL, FM, and SVD++ perform well in integrating such heterogeneous behavior, though they are of high time complexity due to the expanded prediction rule or the iterative procedure. What's more, SVD++, FM, and sTL are all regression-oriented recommendation methods, that is, the goal is for rating prediction (of already-examined items) instead of item recommendation, which thus may not be optimal for a recommendation scenario, because a user cares more about the items' orders instead of the predicted scores. This phenomenon is also observed in our empirical studies. Another notice is that predicting the ratings is a more difficult task than predicting the orders, where the latter is closer to a real situation. In this article, we call the problem of exploiting those two types of user behavior for ranking-oriented personalized recommendation *behavior ranking* (BR) and concentrate our effort on building a novel ranking-oriented algorithm for this problem.

In order to build a ranking-oriented algorithm, we first propose a new preference assumption, that is, dependent preference assumption. Specifically, we assume that a user's rating behavior is dependent on his/her examination behavior, and there is a sequential relationship. Based on this assumption, we develop a novel and generic preference learning algorithm, that is, *transfer to rank* (ToR), which contains a global preference learning task for both rating behavior and examination behavior and a local preference learning task for rating behavior only. Those two dependent preference

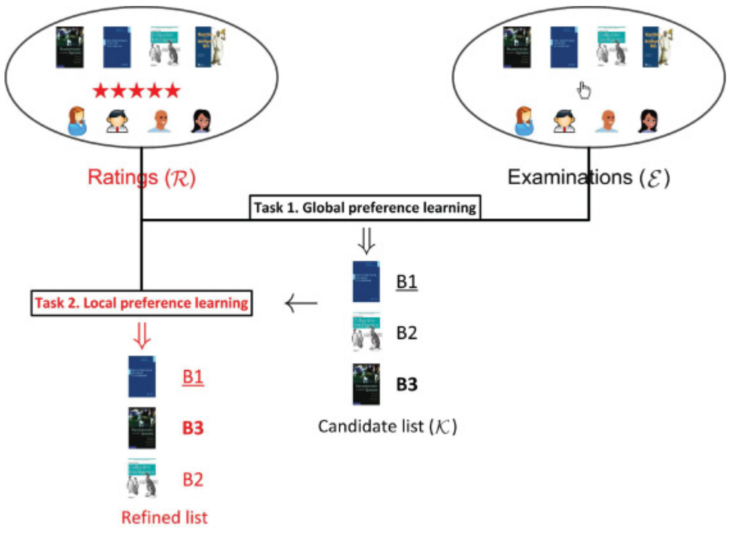
Ratings ($\mathcal{R}$)
Examinations ($\mathcal{E}$)
Task 1. Global preference learning
Task 2. Local preference learning
B1
B2
B3
Candidate list ($\mathcal{K}$)
B1
B3
B2
Refined list

Fig. 1. Illustration of BR with both rating behavior and examination behavior, and ToR with two tasks of preference learning.

learning tasks share knowledge of candidate lists of items. Our ToR algorithm is very generic and can embody different base learners as components for the two preference learning tasks. Empirical results show that our ToR algorithm is able to achieve very competitive recommendation performance as compared with the state-of-the-art methods, which also has the merit of high efficiency and the potential to be used in real systems.

We summarize the main contributions of our work from three different perspectives, that is, problem setting, technique, and empirical studies, as follows.

(1) We study a new ranking-oriented recommendation problem, that is, BR, which aims to provide a personalized ranked list of items for each user by exploiting different types of user behavior.
(2) To solve the BR problem, we develop a novel and generic preference learning algorithm, that is, ToR, which consists of two dependent preference learning tasks of global preference learning and local preference learning.
(3) We conduct extensive empirical studies from different perspectives to thoroughly validate the proposed preference assumption and learning algorithm.

We organize the remainder of the article as follows. In Section 2, we define the problem, discuss the challenges, and introduce some preliminaries that serve as a basis for our solution. In Section 3, we describe our proposed dependent preference assumption, our designed generic preference learning algorithm, and two specific learning tasks in detail. In Section 4, we conduct a series of empirical studies on four large datasets to validate the proposed learning algorithm. In Section 5, we discuss some closely related work from the perspective of different problem settings. Finally, in Section 6, we conclude the work with some remarks and future directions.

2. BACKGROUND AND PRELIMINARIES

2.1. Problem Definition

We illustrate the studied problem in Figure 1. Specifically, in the studied problem of BR, there are two types of user behavior, that is, examinations $\mathcal{E}$ and ratings $\mathcal{R}$, where

Table I. Some Notations and Explanations

$\mathcal{U}$	user set, $u \in \mathcal{U}$
$\mathcal{I}$	item set, $i, i', j \in \mathcal{I}$
$\mathcal{E}$	examination behavior (training data)
$\mathcal{E}_u = \{i \vert (u, i) \in \mathcal{E}\}$	examined items by user u
$\mathcal{R} = \{(u, i, r_{ui})\}$	rating behavior (training data)
$\mathcal{R}_u = \{i \vert (u, i, r_{ui}) \in \mathcal{R}\}$	rated items by user u
$\mathcal{P}$	purchases (test data)
$\mathcal{P}_u = \{i \vert (u, i) \in \mathcal{P}\}$	purchased items by user u
Θ, Φ, Ψ	model parameters
$\mathcal{K}$	shared knowledge of candidate lists

the examinations $\mathcal{E}$ are in the form of (user, item) pairs and the ratings $\mathcal{R}$ are in the form of (user, item, rating) triples. We aim to exploit those user behaviors to generate a personalized ranked list of items for each user u from the set of items that user u has not seen before, that is, $\mathcal{I}\backslash(\mathcal{R}_u \cup \mathcal{E}_u)$, where $\mathcal{E}_u$ and $\mathcal{R}_u$ denotes the set of examined items and the set of rated items of user u, respectively. Notice that we will use ranking-oriented metrics such as normalized discounted cumulative gain (NDCG) and precision for performance evaluation.

We put some math notations and their explanations in Table I.

2.2. Challenges

In order to solve the behavior ranking problem effectively and efficiently, we have to address the following two challenges.

(1) The *sparsity* challenge. Users' rating behavior are usually few due to the fact of users' unwillingness to provide such feedback to a system, which makes the the learning problem rather difficult.
(2) The *scalability* challenge. Users' examination behavior are usually of large volume as they are recorded implicitly by a deployed system without users' proactive involvement, which requires to be modeled in an efficient way.

As a response to the posed challenges, we design a novel solution termed ToR, which consists of two sequential preference learning tasks, that is, a global preference learning task and a local preference learning task with candidate lists of items as shared knowledge.

(1) For the *sparsity* challenge, we first conduct global preference learning based on the abundant examination behavior to identify candidate lists of items and then conduct local preference learning using the sparse rating information to refine the candidate lists. Hence, we address the sparsity challenge in a sequential manner.
(2) For the *scalability* challenge, we instantiate our generic learning algorithm ToR and choose efficient pairwise and pointwise recommendation algorithms for modeling the examination behavior and rating behavior, respectively. In this way, we address the scalability issue of modeling users' examination and rating behavior separately.

2.3. Bayesian Personalized Ranking

In this subsection, we describe a seminal recommendation algorithm, that is, Bayesian personalized ranking (BPR) [Rendle et al. 2009], which is based on a pairwise preference assumption for examination behavior. BPR will be used as a base learner in our global preference learning task, which is thus served as a background model and technical preliminaries of our *transfer to rank* algorithm.

In BPR [Rendle et al. 2009], the users' examination behaviors are modeled based on a so-called pairwise preference assumption, that is, a user u prefers an examined item

i to an unexamined one j. Mathematically, it can be represented as follows [Rendle et al. 2009]:

$$\mathrm{Pref}(u, i|\Phi) > \mathrm{Pref}(u, j|\Phi),\ (u, i) \in \mathcal{E}, (u, j) \notin \mathcal{E}, \tag{1}$$

where Pref(u, i) and Pref(u, j) denote the unobserved preference scores of user u on item i and item j, respectively, and Φ is the model parameters to be learned. With such a pairwise assumption defined on two (user, item) pairs, we can construct an optimization problem that maximizes the overall likelihood that satisfies the pairwise requirements [Rendle et al. 2009],

$$\prod_{(u,i,j);(u,i)\in\mathcal{E},(u,j)\notin\mathcal{E}} \mathrm{Likelihood}(\mathrm{Pref}(u, i|\Phi) > \mathrm{Pref}(u, j|\Phi)). \tag{2}$$

Finally, we can simplify the term Likelihood(Pref($u, i|\Phi$) $>$ Pref($u, j|\Phi$)) via some approximation such as $1/(1 + \exp(-(\mathrm{Pref}(u, i|\Phi) - \mathrm{Pref}(u, j|\Phi))))$ and learn the model parameters Φ via likelihood maximization. In practice, we usually use random sampling to generate two (user, item) pairs in each iteration and apply stochastic gradient descent method for parameter update [Rendle et al. 2009]. Empirically, BPR is a very effective and efficient recommendation algorithm for positive-only feedback such as users' examinations in recommender systems and has been recognized as one of the top-performance algorithms in various academic work and industry-sponsored competitions since its publication in 2009.

2.4. Probabilistic Matrix Factorization

In this subsection, we introduce another seminal recommendation algorithm, that is, probabilistic matrix factorization (PMF) [Salakhutdinov and Mnih 2008], which is based on a pointwise preference assumption for rating behavior. Similarly, PMF will be used as a base learner in the local preference learning task in our *transfer to rank* algorithm.

In PMF [Salakhutdinov and Mnih 2008], the authors propose to exploit the users' rating behavior in a pointwise manner, that is, treat one rating at a time and approximate the rating score by some latent variables. Mathematically, the approximation can be formulated as follows [Salakhutdinov and Mnih 2008]:

$$\mathrm{Loss}(\mathrm{Rating}(u, i) - \mathrm{Pref}(u, i|\Psi)),\ (u, i) \in \mathcal{R}, \tag{3}$$

where Rating(u, i) and Pref($u, i|\Psi$) are the observed user u's rating assigned to item i and the predicted preference score with model parameters Ψ, respectively. In order to approximate the observed score, we usually use a square loss, that is, $\mathrm{Loss}(\mathrm{Rating}(u, i) - \mathrm{Pref}(u, i|\Psi)) = (\mathrm{Rating}(u, i) - \mathrm{Pref}(u, i|\Psi))^2$. By accumulating the loss on each rating, we reach an overall loss function to be minimized [Salakhutdinov and Mnih 2008],

$$\sum_{(u,i)\in\mathcal{R}} (\mathrm{Rating}(u, i) - \mathrm{Pref}(u, i|\Psi))^2. \tag{4}$$

Similarly to that of BPR, with proper regularization of the model parameters Ψ, we can then solve the optimization problem using a stochastic gradient descent method, that is, update the model parameters for each randomly sampled (user, item, rating) triple. PMF is one of the few best base models in the champion's ensemble solution for the Netflix contest and is also the state-of-the-art method for rating prediction in various reported studies.

3. TRANSFER TO RANK

In this section, we first define an overall objective function to be maximized for behavior ranking with heterogeneous user behavior and then introduce a new preference assumption to simplify the optimization problem. Finally, we describe our generic *transfer to rank* algorithm consisting of two preference learning tasks in detail.

The behavior ranking problem with both examinations $\mathcal{E}$ and ratings $\mathcal{R}$ can be formulated as maximizing the following objective function:

$$\text{Prob}(\mathcal{E}, \mathcal{R}|\Theta), \tag{5}$$

where Θ denotes the model parameters that govern the generation of the two types of user behavior. The problem is difficult to solve due to the correlation of the two types of user behavior. As a response, we propose a new assumption, called *dependent preference assumption*, that is, a user's rating behavior follows a user's examination behavior. In other words, a user's rated items are from the set of items that have been examined by the user, which means that a user's rating behavior is dependent on a user's examination behavior. Notice that the proposed assumption is more general than the missing not at random (MNAR) assumption [Marlin et al. 2007; Rec; Steck 2010; Ling et al. 2012], where the latter usually require some specific rating value distribution of the observed data. Furthermore, our assumption targets to rank the unobserved (user, item) pairs while the MNAR assumption mainly focus on approximating the observed numerical ratings of (user, item, rating) triples.

It is actually a common practice adopted by most e-commerce and entertainment sites that a user can rate an item only if he/she has examined the item. However, such dependency information has not been fully exploited by a recommendation algorithm yet. Mathematically, we can represent the dependency between heterogeneous user behavior as follows:

$$\text{Prob}(examination|(user, item)) \cdot \text{Prob}(rating|examination, (user, item)), \tag{6}$$

which means that the probability of an assigned rating to a (user, item) pair is dependent not only on the preference of the user to the item but also on whether the item will be examined by the user. The dependence relationship in Equation (6) looks similar to but differs from the examination hypothesis [Richardson et al. 2007; Craswell et al. 2008; Allen Zhu et al. 2010] that is inspired from the well-known eye-tracking experiment [Granka et al. 2004]. Specifically, the probability that a user clicks a Uniform Resource Locator (URL) after submitting a query can be written as follows [Allen Zhu et al. 2010],

$$\text{Prob}(examination|position) \cdot \text{Prob}(click|examination, (query, url)), \tag{7}$$

where the examination probability, that is, $\text{Prob}(examination|position)$, is solely dependent on the position of the corresponding url returned by a search engine. Notice that most click models [Craswell et al. 2008; Allen Zhu et al. 2010] exploit or extend the examination hypothesis in Equation (7) to capture and explain the *position bias* well, while we do not have such a position-bias issue in Equation (6). Furthermore, to estimate the click-after-examination conditional probability, that is, $\text{Prob}(click|examination, (query, url))$, click models require users' click behavior on the displayed urls, which makes them not applicable to inference $\text{Prob}(rating|examination, (user, item))$ in Equation (6), because the candidate items are not associated with any behavior by the corresponding user.

The dependent preference assumption in Equation (6) also helps alleviate the sparsity problem of rating behavior, because the term $\text{Prob}(rating|examination, (user, item))$ denotes that an observed rating is dependent not only on a (user, item) pair but also on the abundant examination information. Our further derivations and development in the subsequent sections will rely on this assumption.

Input: Users' examination behavior $\mathcal{E}$ and rating behavior $\mathcal{R}$.

Output: A personalized ranked list of items for each user.

Task 1. Conduct global preference learning (i.e., BPR), and recommend a candidate list.

Task 2. Conduct local preference learning (i.e., PMF), and refine the candidate list.

Fig. 2. The algorithm of ToR for BR.

3.1. A Generic Preference Learning Algorithm

With the dependent preference assumption as shown in Equation (6), we develop a generic preference learning algorithm, that is, ToR. In order to obtain an efficient preference learning solution, we simplify the preference assumption in Equation (6) and convert it to two sequential and dependent tasks inspired by a user's online behavior of rating after examination,

$$\text{Prob}(examination|(user, item)) \rightarrow \text{Prob}(rating|examination, (user, item)), \qquad (8)$$

where the notation "$\rightarrow$" means that there is a sequential ordering between the left part and right part. The left part denotes a *global* collaborative filtering with a focus on whether an item will be examined by a user, and the right part denotes a *local* collaborative filtering aiming to estimate the preference score a user will assign to an item if he/she has examined it. Notice that the condition of examination of a user to an item in the right part of Equation (8) denotes the connection and dependency between the global preference learning and local preference learning, which also acts as a knowledge bridge for the two tasks of preference learning in a transfer learning perspective [Pan and Yang 2010]. For this reason, we call our proposed solution *transfer to rank*.

The simplification from Equation (6) to Equation (8) avoids the joint probability (or likelihood) maximization in learning of model parameters that is usually inefficient. We thus have two separate preference learning tasks, including global preference learning for $\text{Prob}(examination|(user, item))$ and local preference learning for $\text{Prob}(rating|examination, (user, item))$. We depict the proposed generic preference learning algorithm in Figure 2. We can see that the global preference learning task and local preference learning task are combined in a sequential manner, which also achieves a balance between the relevance preference and quality preference.

3.2. Global Preference Learning

The goal of the global preference learning task is to identify a candidate list of items for each user u that he/she will be interested in. We thus replace the continuous examination probability $\text{Prob}(examination|(user, item))$ with a discrete one, $\delta(examination|(user, item)) \in \{0, 1\}$, where $\delta(x)$ is an indicator function. In order to find some candidate items from $\mathcal{I}\backslash(\mathcal{E}_u \cup \mathcal{R}_u)$, we propose to combine the examined items and rated items of each user, that is, $\mathcal{E}_u \cup \mathcal{R}_u$, to learn the user's preferences. This problem is usually called one-class collaborative filtering (OCCF) [Pan et al. 2008] or recommendation with implicit feedback [Oard and Kim 1998; Rendle et al. 2009].

Mathematically, we can instantiate the probability $\text{Prob}(examination|(user, item))$ or $\delta(examination|(user, item))$ as follows,

$$\text{Prob}(\mathcal{E} \cup \mathcal{E}_{\mathcal{R}}|\Phi) \Rightarrow \mathcal{K}, \qquad (9)$$

where $\mathcal{E}_{\mathcal{R}}$ denotes the (user, item) pairs converted from the rating behavior by removing the rating scores, and Φ is the set of model parameters that govern the generation of the combined examination behavior $\mathcal{E} \cup \mathcal{E}_{\mathcal{R}}$. Notice that the notation "$\Rightarrow$" denotes the generation procedure of some candidate lists of items (i.e., $\mathcal{K} = \{item|\delta(examination|(user, item)) = 1\}$) with the learned model of the global preference learning task. And as shown in the middle of Figure 1, the candidate lists of items will be used by the local preference learning task. Specifically, those items are taken as the ones that are likely to be examined by the corresponding users.

For the task of maximizing the probability in Equation (9) and learning the model parameters Φ, we may freely use some off-the-shelf OCCF tools [Rendle et al. 2009; Hu et al. 2008; Kabbur et al. 2013]. Considering the training and prediction efficiency and recommendation accuracy, we choose BPR [Rendle et al. 2009] for this task of preference learning. BPR has been extensively studied and verified to be very competitive regarding both efficiency and effectiveness.

Once we have built a global preference learning model, that is, BPR [Rendle et al. 2009] in our empirical studies, we can make predictions of user u's preference scores to items from $\mathcal{I}\backslash(\mathcal{E}_u \cup \mathcal{R}_u)$. Finally, we can rank the items w.r.t. the predicted scores and generate a candidate list of items (e.g., 15 items in our experiments) with highest scores for the user.

The candidate list of items is generated using both the examined items and rated items, that is, the information from both examination behavior and rating behavior. For this reason, we call this task *global* preference learning.

3.3. Local Preference Learning

The goal of the local preference learning task is to refine the candidate list of items for each user u and then produce a final recommendation. For this purpose, we propose to use the rating behavior $\mathcal{R}$ to further estimate the preference scores of the items that have passed the first task of filtering, that is, the items on the candidate list. Notice that our focus has switched to select some items from the candidate list, and thus estimating the preference scores via the rating behavior $\mathcal{R}$ is a natural choice. This problem is usually called collaborative regression or collaborative filtering, and it has a long history [Goldberg et al. 1992; Salakhutdinov and Mnih 2008; Koren 2008; Rendle 2012].

Mathematically, we can instantiate the notation Prob($rating|examination$, ($user$, $item$)) as follows:

$$\text{Prob}(\mathcal{R}|\mathcal{E}, \Psi) \approx \text{Prob}(\mathcal{R}|\mathcal{K}, \Psi), \tag{10}$$

where the likely-to-be-examined candidate lists of items (i.e., $\mathcal{K}$) from the global preference learning task are used to replace the examination behavior $\mathcal{E}$, and Ψ is the set of model parameters used to govern the generation of the rating behavior $\mathcal{R}$. Notice that the candidate lists $\mathcal{K}$ denote the transferred knowledge from the global preference learning task to the local preference learning task in the transfer learning view [Pan and Yang 2010].

For the task of learning the model parameters Ψ in Equation (10), we chose a well-known state-of-the-art collaborative filtering algorithm, that is, PMF [Salakhutdinov and Mnih 2008]. PMF is an efficient and accurate recommendation model, which has been used in various contests, including the Netflix \$1 Million Grand Prize.

Once we have trained the PMF model, we can estimate a user's preferences to the items on the candidate list and adjust the positions of the items accordingly and then take a few top-ranked items for final recommendation.

In the second task of preference learning, the final recommendation list of items is locally refined using the rating behavior only, which is thus called *local* preference learning.

3.4. Learning the ToR

With the above descriptions and discussions about the global preference learning and local preference learning tasks, our *transfer to rank* algorithm in Figure 2 can thus be instantiated with two efficient and effective base learners for examination behavior and rating behavior, respectively, that is, BPR($\mathcal{E}_{\mathcal{R}} \cup \mathcal{E}$) and PMF($\mathcal{R}$).

Specifically, we first take BPR($\mathcal{E}_{\mathcal{R}} \cup \mathcal{E}$) as a background model for global preference learning, which is a ranking-oriented recommendation algorithm aiming to answer the following question:

"whether a user will examine an item."

We then take PMF($\mathcal{R}$) as a refinement model for local preference learning, which is a regression-oriented recommendation algorithm targeted for the following question:

"how will a user like an item if he/she has examined it."

The above two preference learning tasks are described in a whole algorithm in Figure 2. Notice that the seemly very similar click models in information retrieval such as the cascade model [Craswell et al. 2008], position model [Chapelle and Zhang 2009], and general click model [Allen Zhu et al. 2010] actually differ: (i) they do not answer the above first question, because they require the displayed url list as input, and (ii) they cannot answer the above second question, because there is no examination or rating behavior to the items on the candidate list by the corresponding user, which is required by click models for reasoning and prediction.

The time complexity of the proposed ToR algorithm is roughly the summation of that of the two preference learning tasks. In our empirical studies, we chose the very efficient stochastic gradient descent method to learn the model parameters of BPR and PMF. Notice that in each of the two preference learning tasks, that is, BPR($\mathcal{E}_{\mathcal{R}} \cup \mathcal{E}$) and PMF($\mathcal{R}$), we do not have to model different types of user behavior as done in SVD++ [Koren 2008] and sTL [Pan et al. 2016b], which usually makes the learning method very inefficient. Specifically, the time complexities of BPR, PMF, ToR, SVD++, and sTL are $O(dT|\mathcal{E}|)$, $O(dT|\mathcal{R}|)$, $O(dT(2|\mathcal{E}|+|\mathcal{R}|))$, $O(dT|\mathcal{R}|(1+|\bar{\mathcal{E}}_u|))$, and $O(dT|\mathcal{R}|(1+\bar{\mathcal{E}}_u)L)$, respectively, where d is the number of latent dimensions, T and L are the iteration numbers, and $|\bar{\mathcal{E}}_u|$ is the average number of examined items of a user. We can see that the time complexity of our ToR is close to that of BPR and PMF and is much lower than that of SVD++ and sTL.

4. EXPERIMENTAL RESULTS

In this section, we conduct extensive empirical studies on four large datasets to investigate our proposed *transfer to rank* algorithm. In particular, we design the experiments to verify the following hypotheses:

(1) We believe that the proposed *transfer to rank* algorithm can share knowledge between examination behavior and rating behavior via global and local preference learning in a sequential manner and thus addresses the *sparsity* challenge associated with rating behavior and provide accurate recommendations to users.
(2) We believe that the proposed solution can be very efficient with proper global preference learning and local preference learning algorithms, which addresses the *scalability* challenge.

Table II. Statistics of the Datasets Used in the Empirical Studies, Including the Number of Users n, Items m, Rating Behavior $|\mathcal{R}|$, Examination Behavior $|\mathcal{E}|$, Purchase Records (in Validation Data) $|\mathcal{P}(va.)|$ and Purchase Records (in Test Data) $|\mathcal{P}(te.)|$

Dataset	u	m	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{P}(va.)\rvert$	$\lvert\mathcal{P}(te.)\rvert$	$\lvert\mathcal{R}\rvert/nm$	$\lvert\mathcal{R}\rvert/n$
ML10M	71567	10681	2000010	4000024	308673	308702	0.26%	27.9
ML20M	138493	26744	4000052	8000107	579741	580093	0.11%	28.9
Flixter	147612	48794	1639215	3278430	318353	318671	0.02%	11.1
Netflix	480189	17770	19814422	39628846	4556347	4558506	0.23%	41.3

We also include the density (i.e., $|\mathcal{R}|/nm$) and the number of ratings per user (i.e., $|\mathcal{R}|/n$) of each dataset in the last column.

Specifically, we verify the first hypothesis in Sections 4.3–4.6 and the second hypothesis in Section 4.7.

4.1. Datasets and Evaluation Metrics

In our empirical studies, we use four large public datasets, including MovieLens 10M and 20M,[1] Flixter,[2] and Netflix.[3] Specifically, MovieLens 10M (ML10M) contains $71{,}567$ users, $10{,}681$ items, and $10{,}000{,}000$ ratings; MovieLens 20M (ML20M) contains $138{,}493$ users, $26{,}744$ items, and $20{,}000{,}263$ ratings; Flixter contains $147{,}612$ users, $48{,}794$ items, and $8{,}196{,}077$ ratings; and Netflix contains $480{,}189$ users, $17{,}770$ items, and $99{,}072{,}112$ ratings.

In order to simulate the studied problem setting of heterogeneous user behavior as shown in Figure 1, we process the data as follows. For each of the above four datasets, we first split the data into five sets with equal size and then take one set as the training data of rating behavior $\mathcal{R}$, two sets as the training data of examination behavior $\mathcal{E}$ by keeping all the (user, item) pairs [Liu et al. 2010], one set as the validation data of purchases for users' likes by keeping the (user, item) pairs with rating values of 5, and one set as the test data of purchases by keeping the (user, item) pairs with rating values of 5.

We repeat the above procedure 3 times and then obtain three copies of data for each dataset. We list the statistics of one copy of data for each dataset in Table II.

As shown in the problem definition, the goal of behavior ranking is for item recommendation (or item ranking) instead of rating prediction in the traditional collaborative filtering [Salakhutdinov and Mnih 2008] or the recent semi-supervised collaborative filtering [Pan et al. 2016b] problems. Hence, we use some popular ranking-oriented evaluation metrics, including precision, recall, F1, NDCG, and 1-call. For each of the evaluation metrics, we use its top-K version, because we believe that a certain user is more interested in those few (e.g., $K = 5$) items ranked in the first positions. Notice that we keep a bit more, that is, 15, items per user with highest predicted scores as the shared candidate list in our ToR algorithm.

4.2. Baselines and Parameter Settings

In our studied problem, we have two different types of user behavior, that is, examinations and ratings. Hence, we include competitive baselines for examination behavior, for rating behavior, and also for heterogeneous behavior. Specifically, we compare our ToR algorithm with the following five baselines, where PopRank, item-based one-class collaborative filtering (ICF) [Deshpande and Karypis 2004], and BPR [Rendle et al.

[1]http://grouplens.org/datasets/movielens/.
[2]http://www.cs.ubc.ca/jamalim/datasets/.
[3]http://www.netflix.com/.

2009] are for examination behavior; PMF [Salakhutdinov and Mnih 2008] is for rating behavior; and SVD++ [Koren 2008] is for both rating and examination behavior.

—PopRank (popularity-based ranking) ranks and recommends items based their popularity according to the expanded set of examination behavior. Specifically, we first convert the rating behavior to examination behavior by removing the rating values and then merge the original examinations and the converted examinations into one set, that is, $\mathcal{E} \cup \mathcal{E}_{\mathcal{R}}$, on which the item popularity is estimated.
—User-based random walk (URW) is adapted from Díez et al. [2010] to be applied to our studied problem. Specifically, URW uses users' nearest neighbors to simulate social connections (denoted as $\mathbf{A}$) and keeps 1,000 items with highest average numerical rating values (the resulted one-class preference matrix for rating values of 5 is denoted as $\mathbf{S}$) and, finally, predicts the preference via matrix multiplication $\mathbf{SA}$ [Díez et al. 2010].
—ICF [Deshpande and Karypis 2004] recommends a user some items that are similar to his/her examined or rated ones. Similarly, we use the expanded set of examination behavior $\mathcal{E} \cup \mathcal{E}_{\mathcal{R}}$ to calculate the Jaccard index to find the most nearest κ friends with similar tastes.
—BPR [Rendle et al. 2009] is a seminal method for one-class user behavior based on pairwise preference assumption, that is, a user prefers an examined item to an unexamined item. Similarly, we use $\mathcal{E} \cup \mathcal{E}_{\mathcal{R}}$ to learn the model parameters.
—PMF [Salakhutdinov and Mnih 2008] is a well-known solution for rating prediction in collaborative filtering, which learns the latent factors of users and items from the rating behavior $\mathcal{R}$.
—SVD++ [Koren 2008] is the state-of-the-art solution for heterogeneous user behavior such as examinations $\mathcal{E}$ and ratings $\mathcal{R}$, where the examination behavior are used as an augmented term to the rating prediction rule defined on rating behavior in PMF.
—Pairwise collaborative ranking (PCR) is adapted from BPR [Rendle et al. 2009] and collaborative ranking (CR) [Balakrishnan and Chopra 2012], where a factorization-based method (instead of neural networks [Balakrishnan and Chopra 2012]) is used to compare with other factorization-based methods more directly. Specifically, PCR constructs pairwise preferences from the rating values of users' rated items and updates the model parameters in the same way to that of BPR [Rendle et al. 2009].

Notice that we do not include some other regression-oriented modeling techniques such as FM [Rendle 2012] and sTL [Pan et al. 2016b], although they can be used for heterogeneous user behavior. The reason are threefold: high similarity between FM and SVD++, higher time cost of sTL as compared with FM and SVD++, and poor performance of regression-oriented models for the task of ranking-oriented item recommendation in our studied problem.

For the model-based recommendation methods BPR, PMF, SVD++, PCR, and ToR, we fix the learning rate as $\gamma = 0.01$, the number of latent dimensions as $d = 100$, and then search the tradeoff parameter $\alpha \in \{0.001, 0.01, 0.1\}$ and iteration number $T \in \{100, 500, 1000\}$ via the performance of NDCG@15 on the corresponding validation data. The reason we use NDCG@15 for parameter searching is that its result is dependent on the ordering of the items on the candidate list and is also recognized as a more important metric as compared with other metrics such as precision and recall. Notice that we have to do parameter searching 2 times using the same validation data for the tasks of global and local preference learning in our ToR. To study the effect of using different numbers of latent dimensions, we later change it to $d \in \{50, 100, 150\}$, in which the best tradeoff parameter and iteration number are searched again in the same way. For ICF, we first fix the number of nearest neighbors as $\kappa = 100$, and then change it to $\kappa \in \{50, 100, 150\}$ to do a comparative study to that of model-based methods. For URW, we also use Jaccard index as the similarity measurement and fix the number of nearest neighbors for social connections as $\max(20, \frac{\kappa m}{n})$ to have comparable neighborhood size to that of ICF.

4.3. Main Results

We show the main quantitative results in Table III, from which we can make the following observations:

—The proposed ToR algorithm performs significantly best in almost all cases except NDCG@5 on Flixter. Such promising results clearly show the effectiveness of our ToR preference learning algorithm in addressing the sparsity challenge of rating behavior by combing two different types of user behavior.

—The overall relative performance of different types of methods is very clear, that is, PopRank, URW, ICF, and BPR are usually much better than the regression-oriented methods (i.e., PMF and SVD++) and the collaborative ranking method with explicit feedback (i.e., PCR), which shows the critical importance of selecting a matched type of recommendation method for a specific problem.

—For PopRank, URW, ICF, and BPR, the performance ordering is usually BPR > ICF, URW > PopRank, which shows the effectiveness of model-based (i.e., BPR) and memory-based (i.e., ICF and URW) recommendation methods. This observation is also consistent with that of modeling homogeneous user behavior such as rating prediction or item ranking. Notice that URW does not perform well when the rating behaviors are few such as in the Flixter data, which is caused by the fact that the resulted matrix $\mathbf{A}$ is too sparse to be used for prediction.

—For the two regression-oriented methods of PMF and SVD++, we can see that SVD++ is always better than PMF, which shows that those two different types of user behavior are complementary to each other. However, they are ineffective as compared with the ranking-oriented method of PCR in most cases. The reason is that both PMF and SVD++ are specifically devised for rating prediction of already rated (user, item) pairs similar to that of our local preference learning task, rather than for rating prediction of all (user, item) pairs. Notice that running SVD++ is also of high time cost (see the results in the sequel). The poor performance on ranking-oriented metrics of PMF and SVD++ also echoes the observation in Cremonesi et al. [2010] that good rating prediction performance does not ensure good recommendation performance. Furthermore, for the ranking-oriented method with explicit feedback (i.e., PCR), we can see that it performs worse than the counterpart with combined implicit feedback (i.e., BPR), which again shows the difficulty of the sparsity challenge of the rating behavior.

—From the perspective of modeling techniques, we mix ranking-oriented modeling and regression-oriented modeling sequentially in ToR and obtain significantly better results than either ranking-oriented modeling or regression-oriented modeling alone. This observation is of importance in theoretical studies and practical deployment, because we may not focus on selecting one single best modeling technique but can turn to mix them in a proper way. The effectiveness of our sequential mixing manner in ToR also verifies the effectiveness of the proposed dependent preference assumption between those two different types of user behavior.

4.4. Improvement after Local Preference Learning

To have some deep understanding of the improvement after local preference learning as shown in Table III, we check the change between the lists as generated by global preference learning and that refined after local preference learning. We select two example users and put the corresponding lists in Table IV. The items preferred by

Table III. Recommendation Performance of ToR and Other Methods on ML10M, ML20M, Flixter, and Netflix Using Five Ranking-Oriented Evaluation Metrics

Dataset	Method	Prec@5	Rec@5	F1@5	NDCG@5	1-call@5
ML10M	PopRank	0.0645$_{\pm0.0003}$	0.0882$_{\pm0.0007}$	0.0617$_{\pm0.0003}$	0.0923$_{\pm0.0008}$	0.2608$_{\pm0.0015}$
	URW	0.0784$_{\pm0.0009}$	0.1014$_{\pm0.0017}$	0.0730$_{\pm0.0009}$	0.1142$_{\pm0.0020}$	0.3164$_{\pm0.0030}$
	ICF	0.0815$_{\pm0.0005}$	0.1006$_{\pm0.0006}$	0.0744$_{\pm0.0004}$	0.1084$_{\pm0.0005}$	0.3195$_{\pm0.0013}$
	BPR	0.0901$_{\pm0.0020}$	0.1194$_{\pm0.0034}$	0.0856$_{\pm0.0020}$	0.1234$_{\pm0.0038}$	0.3538$_{\pm0.0051}$
	PMF	0.0005$_{\pm0.0001}$	0.0002$_{\pm0.0001}$	0.0003$_{\pm0.0001}$	0.0007$_{\pm0.0002}$	0.0022$_{\pm0.0006}$
	SVD++	0.0152$_{\pm0.0010}$	0.0107$_{\pm0.0007}$	0.0103$_{\pm0.0007}$	0.0198$_{\pm0.0016}$	0.0588$_{\pm0.0033}$
	PCR	0.0275$_{\pm0.0005}$	0.0350$_{\pm0.0005}$	0.0252$_{\pm0.0005}$	0.0385$_{\pm0.0006}$	0.1258$_{\pm0.0027}$
	ToR	**0.1169**$_{\pm0.0001}$	**0.1561**$_{\pm0.0002}$	**0.1112**$_{\pm0.0001}$	**0.1653**$_{\pm0.0001}$	**0.4346**$_{\pm0.0012}$
ML20M	PopRank	0.0626$_{\pm0.0004}$	0.0828$_{\pm0.0002}$	0.0587$_{\pm0.0003}$	0.0893$_{\pm0.0004}$	0.2557$_{\pm0.0012}$
	URW	0.0690$_{\pm0.0022}$	0.0861$_{\pm0.0032}$	0.0630$_{\pm0.0023}$	0.1017$_{\pm0.0031}$	0.2806$_{\pm0.0079}$
	ICF	0.0772$_{\pm0.0006}$	0.0962$_{\pm0.0010}$	0.0709$_{\pm0.0005}$	0.1036$_{\pm0.0012}$	0.3045$_{\pm0.0016}$
	BPR	0.0893$_{\pm0.0012}$	0.1140$_{\pm0.0034}$	0.0832$_{\pm0.0016}$	0.1213$_{\pm0.0028}$	0.3462$_{\pm0.0048}$
	PMF	0.0022$_{\pm0.0002}$	0.0012$_{\pm0.0002}$	0.0013$_{\pm0.0002}$	0.0027$_{\pm0.0004}$	0.0094$_{\pm0.0010}$
	SVD++	0.0136$_{\pm0.0014}$	0.0101$_{\pm0.0013}$	0.0094$_{\pm0.0011}$	0.0177$_{\pm0.0021}$	0.0544$_{\pm0.0054}$
	PCR	0.0244$_{\pm0.0002}$	0.0340$_{\pm0.0007}$	0.0233$_{\pm0.0003}$	0.0360$_{\pm0.0009}$	0.1129$_{\pm0.0009}$
	ToR	**0.1101**$_{\pm0.0010}$	**0.1436**$_{\pm0.0022}$	**0.1032**$_{\pm0.0012}$	**0.1545**$_{\pm0.0019}$	**0.4101**$_{\pm0.0036}$
Flixter	PopRank	0.0476$_{\pm0.0002}$	0.0754$_{\pm0.0004}$	0.0426$_{\pm0.0001}$	0.0778$_{\pm0.0001}$	0.1966$_{\pm0.0001}$
	URW	3.0E-5$_{\pm4.3E-6}$	2.0E-5$_{\pm1.1E-5}$	1.7E-5$_{\pm8.6E-6}$	4.9E-5$_{\pm8.8E-6}$	1.4E-4$_{\pm2.5E-5}$
	ICF	0.0578$_{\pm0.0002}$	0.1017$_{\pm0.0003}$	0.0550$_{\pm0.0002}$	0.1000$_{\pm0.0002}$	0.2326$_{\pm0.0004}$
	BPR	0.0623$_{\pm0.0003}$	0.1000$_{\pm0.0007}$	0.0564$_{\pm0.0005}$	**0.1070**$_{\pm0.0005}$	0.2518$_{\pm0.0020}$
	PMF	0.0002$_{\pm0.0000}$	0.0001$_{\pm0.0000}$	0.0001$_{\pm0.0000}$	0.0002$_{\pm0.0001}$	0.0008$_{\pm0.0002}$
	SVD++	0.0035$_{\pm0.0001}$	0.0015$_{\pm0.0001}$	0.0017$_{\pm0.0001}$	0.0041$_{\pm0.0001}$	0.0133$_{\pm0.0003}$
	PCR	0.0151$_{\pm0.0023}$	0.0169$_{\pm0.0056}$	0.0116$_{\pm0.0027}$	0.0199$_{\pm0.0037}$	0.0693$_{\pm0.0109}$
	ToR	**0.0654**$_{\pm0.0008}$	**0.1043**$_{\pm0.0017}$	**0.0584**$_{\pm0.0008}$	0.0980$_{\pm0.0011}$	**0.2619**$_{\pm0.0022}$
Netflix	PopRank	0.0484$_{\pm0.0002}$	0.0323$_{\pm0.0003}$	0.0295$_{\pm0.0002}$	0.0521$_{\pm0.0003}$	0.1965$_{\pm0.0007}$
	URW	–	–	–	–	–
	ICF	0.1006$_{\pm0.0002}$	0.0681$_{\pm0.0005}$	0.0626$_{\pm0.0003}$	0.1165$_{\pm0.0004}$	0.3610$_{\pm0.0016}$
	BPR	0.1087$_{\pm0.0003}$	0.0755$_{\pm0.0006}$	0.0686$_{\pm0.0003}$	0.1262$_{\pm0.0005}$	0.3880$_{\pm0.0007}$
	PMF	0.0304$_{\pm0.0019}$	0.0133$_{\pm0.0014}$	0.0151$_{\pm0.0013}$	0.0322$_{\pm0.0021}$	0.1284$_{\pm0.0077}$
	SVD++	–	–	–	–	–
	PCR	0.0237$_{\pm0.0006}$	0.0125$_{\pm0.0005}$	0.0132$_{\pm0.0005}$	0.0255$_{\pm0.0009}$	0.1008$_{\pm0.0027}$
	ToR	**0.1326**$_{\pm0.0005}$	**0.0895**$_{\pm0.0003}$	**0.0826**$_{\pm0.0003}$	**0.1559**$_{\pm0.0005}$	**0.4462**$_{\pm0.0015}$

The number of latent dimensions and nearest neighbors are fixed as $d = \kappa = 100$. The significantly best results are marked in bold (p value < 0.01). We use "–" to denote the cases that do not produce results within 168 hours.

the users according to the test data are marked in bold. We can have the following observations:

—For each of those two users, the ordering of the two lists differ considerably, for example, there are only two items (marked with an underline) that do not change the ranking positions, which means that the local preference learning has a significant impact to the candidate list.
—For the preferred items (marked in bold), the changes of their rankings will usually improve the performance, for example, there are three increases and one decrease for example user 1 and four increases and one decrease for example user 2.

The improvement contributed by the second task of local preference learning also shows that using BPR alone is not sufficient for modeling the heterogeneous user behavior, and our ToR solution is a promising solution.

Table IV. The Lists of Items after Global Preference Learning and Local Preference Learning

Ranking	Example user 1			Example user 2		
Position	Global	Local	Change	Global	Local	Change
1	1024	**536**	↑	2561	**1271**	↑
2	1562	**497**	↑	2564	1251	
3	**5**	1024		2467	**1226**	↑
4	1537	**1446**	↑	1316	**1189**	↑
5	**536**	1562		1251	2564	
6	<u>2341</u>	<u>2341</u>		1899	1078	
7	**1446**	838		2570	**1076**	↑
8	<u>581</u>	<u>581</u>		1078	2561	
9	2348	1337		**1274**	2467	
10	776	2348		**1189**	1617	
11	838	**5**	↓	1617	**1274**	↓
12	1828	2252		**1076**	1911	
13	**497**	776		**1226**	2570	
14	2252	1537		**1271**	1899	
15	1337	1828		1911	1316	

The items preferred by the users according to the test data are marked in bold. We use ↑ and ↓ to denote an increase and a decrease of the position of a purchased item after local preference learning, respectively.

4.5. Performance with Different *K*

In this subsection, we study the performance of top-K evaluation metrics with different values of $K \in \{1, 2, 3, \ldots, 15\}$. We show the Prec@K and NDCG@K results of the three most competitive methods, that is, ICF, BPR, and ToR, in Figure 3. Notice that the results on other metrics are similar and are thus not included. From the results in Figure 3, we can see the following:

—The relative performance ordering on different Prec@K and NDCG@K of ICF, BPR, and ToR over ML10M, ML20M, and Netflix are very clear, which shows that our ToR algorithm is much better than the other two methods in most cases. This again shows the effectiveness of our proposed ToR algorithm.

—The results of ToR on Flixter looks poorer than that of BPR, which means that the second task of local preference learning is not able to further refine and improve the candidate list. This is caused by the PMF model that is not sufficiently trained with the extremely sparse data of rating behavior, that is, 0.02% of density and 11.1 ratings per user (see the sparsity of the datasets in the rightmost columns of Table II). Notice that we have a validation data for the tradeoff parameter and iteration number selection, which can also be used to decide whether the regression-based modeling (i.e., local preference learning) in ToR is needed. Hence, that situation can actually be avoided. We include such results to emphasize that the combination of ranking-oriented modeling and regression-oriented modeling may degrade the performance sometimes especially when the data of rating behavior is extremely sparse. In practice, we can use the validation data to check the learning procedure and avoid such a phenomenon.

—The results of Prec@15 of ToR and BPR are exactly the same, because the prediction and refinement after the task of local preference learning will only adjust the ordering but not the set of the top-15 items. Notice that NDCG@15 is defined on the orderings of the items and thus will be changed after the second task of local preference learning.

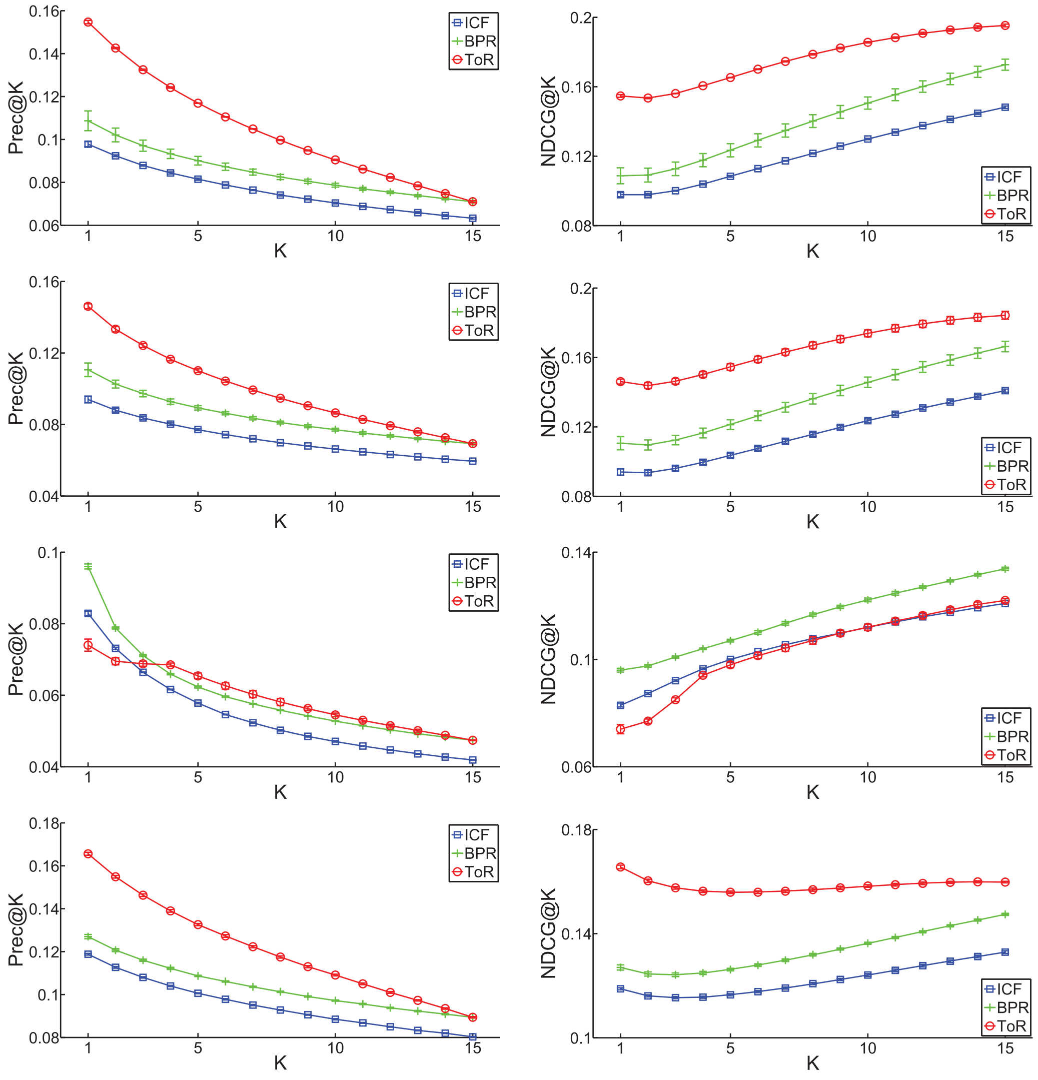

Fig. 3. Recommendation performance of ICF, BPR, and ToR on Prec@K and NDCG@K with different values of K. The number of latent dimensions and nearest neighbors are fixed as $d = \kappa = 100$. From the top row to the bottom row are the results of ML10M, ML20M, Flixter, and Netflix, respectively.

The results in Figure 3 again clearly show the effectiveness of our ToR preference learning algorithm, in particular of the items at the top of a recommendation list.

4.6. Performance Using Different Dimensions

In this subsection, we study the impact of the number of latent dimensions of our proposed ToR algorithm. For comparison, we also include ICF with the same numbers of nearest neighbors. The results of the most competitive three methods are reported in Figure 4. We can have the following observations:

—The overall relative performance ordering of the three methods with different numbers of latent dimensions or nearest neighbors are consistent, that is,

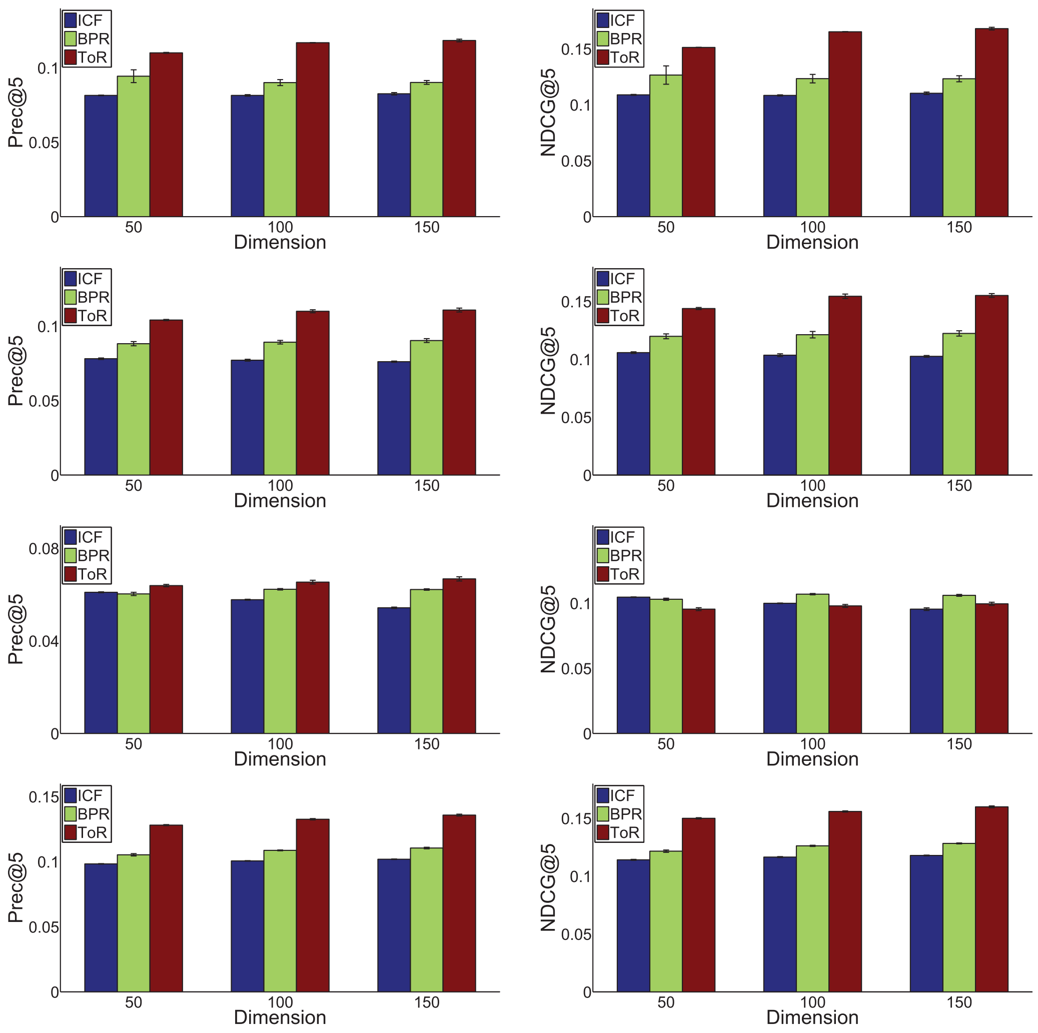

Fig. 4. Recommendation performance with different numbers of latent dimensions in BPR and ToR, and different numbers of nearest neighbors in ICF. From the top row to the bottom row are the results of ML10M, ML20M, Flixter, and Netflix, respectively.

ToR > BPR > ICF, which again shows the superiority of our proposed ToR algorithm in terms of recommendation accuracy.

—The recommendation performance of each method usually increases when the models are associated with more parameters, that is, larger number of dimensions or nearest neighbors. This property and trend is helpful in real deployment, especially for a balance between recommendation accuracy and model complexity.

Overall, the results in Figure 4 are expected and reasonable, which again shows the effectiveness of our preference assumption and learning algorithm.

4.7. Time and Memory Usage

In this subsection, we study the time and space cost of training the models. We run the learning algorithms of SVD++, PMF, and BPR on Windows Server 2008 with

Table V. Time and Memory Usage of the Model-Based Methods, That Is, SVD++, PMF, BPR, and our ToR, on Different Datasets

Dataset	Metric	SVD++	PMF	BPR	ToR
ML10M	CPU time per iteration (second)	158.9	1.9	9.0	11.0
	Memory usage (MB)	3000.3	2498.6	2878.1	2878.1
ML20M	CPU time per iteration (second)	358.8	4.7	18.5	23.2
	Memory usage (MB)	4546.2	2918.4	4215.4	4215.4
Flixter	CPU time per iteration (second)	300.9	1.6	8.0	9.6
	Memory usage (MB)	1884.2	1679.4	1822.7	1822.7
Netflix	CPU time per iteration (second)	3423.4	21.6	104.3	125.9
	Memory usage (MB)	6615.0	4003.8	6553.6	6553.6

Notice that the CPU time is for one iteration, and the number of latent dimensions is fixed as $d = 100$.

Intel(R) Core(TM) i7-3770 CPU @ 3.40 GHz (1-CPU/4-core)/16GB Random-Access Memory (RAM). We report the Central Processing Unit (CPU) time for one iteration of each algorithm, and also the memory consumption when the algorithm is running. For the CPU time of ToR, we use the summation of that of BPR and PMF, and for the memory usage of ToR, we use that of BPR since its memory cost is higher than that of PMF. We show the quantitative results in Table V and can have the following observations:

—The time cost of SVD++ is much higher than that of PMF, BPR, and ToR, which is caused by the update rules related to each examined item of a corresponding user u for each randomly sampled (u, i, r_{ui}) triple. Because the time cost of the proposed ToR algorithm can be roughly estimated as the summation of that of BPR and PMF, we can see that ToR is a very efficient solution for heterogeneous user behavior as we discussed before.

—The memory usage of SVD++ is again higher than that of PMF, BPR, and ToR, which is caused by the additional set of model parameters for examined items. This observation again shows the practical merits of the proposed ToR algorithm.

The results in Table V show that our proposed ToR algorithm is a very efficient approach for heterogeneous user behavior, which has a big potential for real deployment.

5. RELATED WORK

We put our work in the context of recommendation with homogeneous and heterogeneous user behavior and auxiliary data and thus discuss the related work in following three directions, including recommendation with homogeneous user behavior, recommendation with heterogeneous user behavior, and recommendation with user behavior and auxiliary data.

5.1. Recommendation with Homogeneous User Behavior

In this section, we discuss some related work on exploiting homogeneous user behavior in two types of modeling fashions, including ranking-oriented modeling and regression-oriented modeling. Specifically, we will describe some work on collaborative regression (or collaborative filtering) with rating behavior, collaborative ranking with rating behavior, and one-class collaborative filtering with examination behavior.

Collaborative regression (or collaborative filtering) [Goldberg et al. 1992] aims to exploit users' numerical rating scores and learn users' preferences for personalization services, which has been an important research topic for more than two decades. Mathematically, the problem is often formulated as a missing value prediction problem or a matrix completion problem, that is, predict the missing scores in a user-item rating matrix. Technically, different machine-learning and data-mining approaches have been invented, including memory-based methods, model-based methods, and hybrid methods. First, a memory-based method such as user-oriented collaborative filtering makes a rating prediction of a target user to a target item with three basic steps, including (i) calculating similarities between the target user and other users, (ii) sorting the users w.r.t. the similarities and constructing a neighborhood with a few most similar users, and (iii) estimating the preference score by aggregating the assigned ratings of the users in the neighborhood to the target item. Second, a model-based method such as probabilistic matrix factorization and other matrix completion methods predict a missing entry of a matrix with two steps of matrix decomposition and matrix reconstruction, that is, decompose the rating matrix into two or three latent low-rank matrices with the constraint of approximating the observed rating scores (i.e., matrix decomposition), and then predict the ratings by multiplying the latent feature matrices (i.e., matrix reconstruction). Third, a hybrid method combines more than one memory-based and/or model-based methods in some mechanisms such as voting and averaging, which usually performs better.

Collaborative ranking focuses on item ranking instead of rating prediction in collaborative filtering, where the latter is usually deemed as an intermediate procedure for a recommendation purpose. The main justification to switch from rating prediction to item ranking is often supported by the fact that the final goal of recommendation is to deliver a ranked list of items rather than to provide some estimated preference scores. Mathematically, the item ranking problem is usually formulated as a learning-to-rank problem in information retrieval but without content information such as textual features of queries and documents. Technically, existing approaches are mostly model-based methods such as maximum-margin matrix factorization (MMMF) [Srebro et al. 2004], CoFiRank [Weimer et al. 2007], and CR [Balakrishnan and Chopra 2012]. In MMMF [Srebro et al. 2004], a maximum-margin style loss function defined on pairs of observed ratings is used to construct an optimization problem, where low-rank latent feature matrices are taken as model parameters to be learned. MMMF is empirically better than PMF [Salakhutdinov and Mnih 2008] due to the merit of modeling the ratings in a ranking-oriented manner, which is closer to the final evaluation metrics. In CoFiRank [Weimer et al. 2007], the authors design a loss function that directly optimizes the NDCG-based evaluation metric, that is, the learned model parameters are optimal in terms of NDCG. As expected, CoFiRank performs even better than MMMF. However, it is also of high time complexity due to the sophisticated modeling and optimization issues. In CR [Balakrishnan and Chopra 2012], the authors turn to address the main obstacle when making use of various existing learning-to-rank techniques in information retrieval, that is, the lack of textual features. As a response, a two-step solution is designed, including (i) applying basic matrix factorization to obtain latent feature matrices as "virtual" textual features and (ii) building a learning-to-rank model with the latent features in the same way with that in information retrieval. Empirically, the CR approach is more efficient and is also more accurate.

One-class collaborative filtering [Pan et al. 2008] makes use of one-class user behavior such as examinations only. The main challenge of modeling one-class behavior is its "positive-only" property, that is, there is no explicit negative feedback like the lowest preference scores in numerical ratings. Notice that, due to the lack of numerical preference scores, most works build a ranking-oriented model instead of a regression-oriented model. Hence, it is mathematically formulated as a one-class learning-to-rank problem without content information. Technically, the developed models can roughly be divided into three categories parallel to that of learning-to-rank algorithms in information retrieval, that is, the pointwise method, pairwise method and listwise method. For a pointwise method, we usually randomly sample some missing entries from the user-item rating matrix as negative feedback and then apply some existing collaborative filtering methods or design some new methods aiming to minimize a pointwise square loss [Pan et al. 2008]. For a pairwise method, some preference assumptions based on (user, item) pairs will be made and some loss functions will then be derived to be optimized. One of the most representative work in this category is BPR [Rendle et al. 2009]. For a listwise method, some loss function defined on more than two (user, item) pairs is used to be optimized, which is usually designed to approximate some ranking-oriented evaluation metric [Shi et al. 2012]. Empirically, a listwise method performs better than a pairwise or pointwise method but is often of high time cost, while the pairwise method is a good balance in terms of accuracy and efficiency. For this reason, we choose the pairwise method BPR as a base learner in our global preference learning task.

5.2. Recommendation with Heterogeneous User Behavior

In this section, we focus on some recent work on modeling different types of user behavior in recommendation systems. Specifically, we will discuss the representative work on heterogeneous one-class behavior such as examinations and transactions, multi-class behavior (e.g., ratings) and one-class behavior (e.g., examinations), and multi-class behavior (e.g., ratings) and binary-class behavior (e.g., likes/dislikes).

Recommendation with heterogeneous one-class behavior (or transfer learning for behavior prediction, TLBP [Pan and Yang 2016]) is a recently proposed problem aiming to leverage users' examination behavior when modeling the transaction behavior. For TLBP [Pan and Yang 2016], there are mainly three types of solutions. First, in adaptive Bayesian personalized ranking, the authors focus on learning some confidence weight for each examination behavior so the uncertainty problem associated with the examinations can be alleviated, and then the weighted examination data and the transaction data are combined and used to train a BPR model in an adaptive fashion. Second, in transfer via joint similarity learning, the authors focus on jointly learning a similarity between a candidate item and a purchased item, and a similarity between a candidate item and an identified likely-to-prefer item (from examinations), so the sparsity challenge of transaction data and the uncertainty challenge of examination data can be addressed. Thirdly, in role-based Bayesian personalized ranking (RBPR), the authors design an examiner-based preference learning and a purchaser-based preference learning in a hybrid way. Empirically, RBPR is much more efficient with comparable recommendation accuracy.

Recommendation with multi-class and one-class behavior (or recommendation with explicit and implicit feedback) has been well recognized as an important recommendation problem [Jawaheer et al. 2014]. There have been three modeling styles proposed so far, including collective [Liu et al. 2010], integrative [Koren 2008; Rendle 2012], and iterative [Pan et al. 2016b]. First, in a collective modeling approach [Liu et al. 2010], the authors propose to jointly factorize a rating matrix and an examination matrix (with some randomly generated negative feedback), for which the well-known collective matrix factorization model [Singh and Gordon 2008] can be applied. Collective modeling can usually improve the recommendation accuracy over that of probabilistic matrix factorization on rating behavior only. Second, in two integrative modeling approaches, that is, SVD++ [Koren 2008] and FM [Rendle 2012], the main idea is to expand the prediction function for rating data to integrate some additional term(s) for examinations and/or the interactions between ratings and examinations. Such integrative modeling techniques are usually of higher recommendation accuracy but also of high time complexity in the learning procedure. Third, in a recent iterative modeling approach [Pan et al. 2016b], that is, sTL, the authors propose to identify some examinations and then incorporate them into a prediction function in an iterative fashion instead of the commonly used one-time data integration. Empirically, the iterative modeling approach usually performs better in terms of recommendation accuracy.

However, all these modeling techniques are regression-oriented, and the evaluation metrics are usually the *root mean square error* or *mean absolute error* instead of some ranking-oriented metrics. In this article, although the input data of the studied problem are the same, the modeling technique and evaluation metrics totally differ, that is, ranking-oriented modeling and ranking-oriented evaluation metrics. Notice that some conversational recommender systems [Zanker and Jessenitschnig 2009] and social recommender systems [Díez et al. 2010] also make use of explicit, implicit, and other data, but their problem settings differ from ours.

Recommendation with multi-class and binary-class behavior (or recommendation with heterogeneous explicit feedback) learns users' preferences from different types of explicit feedback, such as ratings and likes/dislikes. The main challenge is the heterogeneity or the different granularities of users' feedback. The basic idea of resolving such heterogeneity is to model the data-dependent effect and data-independent effect simultaneously. Besides this, we may also introduce interactions between two modeling tasks for two different types of user behaviors or even expand the prediction rule via integrating the binary feedback into the multi-class feedback [Pan et al. 2016a]. Empirically, sophisticated modeling with rich interactions and complex prediction rules can usually improve the recommendation accuracy.

5.3. Recommendation with User Behavior and Auxiliary Data

In this section, we discuss some related work on a relatively new research topic, that is transfer learning for collaborative recommendation with auxiliary data. Auxiliary data refers to the additional information that are relevant to users' behavior such as the four dimensions of content, context, network, and feedback as mentioned in a recent survey [Pan 2016]. Collaborative recommendation with auxiliary data is a very important problem, because a real recommendation system almost always consists of some users' behavior and some other information such as items' descriptions and users' profiles, and so on. Specifically, there are two problem setups, including recommendation with homogeneous user behavior and auxiliary data and recommendation with heterogeneous user behavior and auxiliary data.

Recommendation with homogeneous user behavior and auxiliary data can be instantiated to recommendation with rating behavior and auxiliary data or recommendation with examination behavior and auxiliary data. For recommendation with ratings and auxiliary data, many works have been published, including ratings and social networks [Tang et al. 2013], temporal and geometrical information [Koren 2010; Tan et al. 2014], user profiles and item descriptions [Zhen et al. 2009], and so on. There is a relatively complete survey on this direction [Pan 2016]. For recommendation with examinations and auxiliary data, only a few algorithms have been proposed [Elbadrawy and Karypis 2015], and the auxiliary data can again be social connections, contextual information, and entity descriptions.

Recommendation with heterogeneous user behavior and auxiliary data is a more difficult problem with very few work, which is also more closely related to the industry scenarios. For this problem, we have to address not only the heterogeneity, uncertainty, and sparsity challenges but also the scalability and real-time computation challenges. We believe that there will be more unified solutions proposed in this direction.

In this article, we study the problem of recommendation with heterogenous multi-class and one-class behavior but focus on building a ranking-oriented recommendation model, which thus differs from the problem settings of all the above work.

6. CONCLUSIONS AND FUTURE WORK

In this article, we have studied a new recommendation problem, that is, BR, which aims to construct a ranking-oriented recommendation model based on examination and rating behavior. In order to address the sparsity and scalability challenges of the studied problem, we first propose a new assumption, that is, dependent preference assumption, to digest the correlated heterogeneous behavior well, and then develop a simple and generic ToR algorithm for preference learning and item ranking. Our ToR algorithm is able to integrate heterogeneous behavior and share knowledge (i.e., candidate lists of items) in a sequential manner and learn the users' preferences in a rather efficient way, which thus addresses the aforementioned two challenges. Empirical results on four large datasets show that our ToR algorithm is very competitive as compared with the state-of-the-art methods regarding both recommendation accuracy and learning efficiency.

For future work, we are interested in deploying our ToR algorithm in real industry scenarios. We are also interested in generalizing the ToR recommendation algorithm to more types of user behavior and information such as the textual content [Mikolov et al. 2013] of users' searched queries and visual content [Zhao et al. 2016] in the items' advertisements and promotions and to more cross-domain and transfer learning paradigms [Zheng 2015].

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their expert and constructive comments and suggestions.

REFERENCES

Zeyuan Allen Zhu, Weizhu Chen, Tom Minka, Chenguang Zhu, and Zheng Chen. 2010. A novel click model and its applications to online advertising. In *Proceedings of the 3rd International Conference on Web Search and Web Data Mining (WSDM'10)*. 321–330.

Suhrid Balakrishnan and Sumit Chopra. 2012. Collaborative ranking. In *Proceedings of the 5th International Conference on Web Search and Web Data Mining*. 143–152.

Olivier Chapelle and Ya Zhang. 2009. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th International Conference on World Wide Web (WWW'09)*. 1–10.

Nick Craswell, Onno Zoeter, Michael J. Taylor, and Bill Ramsey. 2008. An experimental comparison of click position-bias models. In *Proceedings of the International Conference on Web Search and Web Data Mining (WWW'08)*. 87–94.

Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys'10)*. 39–46.

Mukund Deshpande and George Karypis. 2004. Item-based top-N recommendation algorithms. *ACM Trans. Inf. Syst.* 22, 1 (2004), 143–177.

Fernando Díez, J. Enrique Chavarriaga, Pedro G. Campos, and Alejandro Bellogín. 2010. Movie recommendations based in explicit and implicit features extracted from the filmtipset dataset. In *Proceedings of the Workshop on Context-Aware Movie Recommendation (CAMRa'10)*. 45–52.

Asmaa Elbadrawy and George Karypis. 2015. User-specific feature-based similarity models for top-n recommendation of new items. *ACM Trans. Intell. Syst. Technol.* 6, 3 (2015), 33:1–33:20.

David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. 1992. Using collaborative filtering to weave an information tapestry. *Commun. ACM* 35, 12 (1992), 61–70.

Laura A. Granka, Thorsten Joachims, and Geri Gay. 2004. Eye-tracking analysis of user behavior in WWW search. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'04)*. 478–479.

Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM'08)*. 263–272.

Gawesh Jawaheer, Peter Weller, and Patty Kostkova. 2014. Modeling user preferences in recommender systems: A classification framework for explicit and implicit user feedback. *ACM Trans. Interact. Intell. Syst.* 4, 2 (2014), 8:1–8:26.

Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: Factored item similarity models for top-N recommender systems. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'13)*. 659–667.

Yehuda Koren. 2008. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08)*. 426–434.

Yehuda Koren. 2010. Collaborative filtering with temporal dynamics. *Commun. ACM* 53, 4 (2010), 89–97.

Guang Ling, Haiqin Yang, Michael R. Lyu, and Irwin King. 2012. Response aware model-based collaborative filtering. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI'12)*. 501–510.

Nathan N. Liu, Evan W. Xiang, Min Zhao, and Qiang Yang. 2010. Unifying explicit and implicit feedback for collaborative filtering. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM'10)*. 1445–1448.

Benjamin M. Marlin, Richard S. Zemel, Sam T. Roweis, and Malcolm Slaney. 2007. Collaborative filtering and the missing at random assumption. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI'07)*. 267–275.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781 (2013).

Douglas Oard and Jinmook Kim. 1998. Implicit feedback for recommender systems. In *Proceedings of the AAAI Workshop on Recommender Systems*.

Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan M. Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM'08)*. 502–511.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* 22, 10 (2010), 1345–1359.

Weike Pan. 2016. A survey of transfer learning for collaborative recommendation with auxiliary data. *Neurocomputing* 177 (2016), 447–453.

Weike Pan, Shanchuan Xia, Zhuode Liu, Xiaogang Peng, and Zhong Ming. 2016a. Mixed factorization for collaborative recommendation with heterogeneous explicit feedbacks. *Inf. Sci.* 332 (2016), 84–93.

Weike Pan and Qiang Yang. 2016. Transfer learning for behavior prediction. *IEEE Intell. Syst.* 31, 2 (2016), 86–88.

Weike Pan, Qiang Yang, Yuchao Duan, and Zhong Ming. 2016b. Transfer learning for semi-supervised collaborative recommendation. *ACM Trans. Interact. Intell. Syst.* 6, 2 (2016), 10:1–10:21.

Steffen Rendle. 2012. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.* 3, 3 (2012), 57:1–57:22.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI'09)*. 452–461.

Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: Estimating the click-through rate for new ads. In *Proceedings of the 16th International Conference on World Wide Web (WWW'09)*. 521–530.

Ruslan Salakhutdinov and Andriy Mnih. 2008. Probabilistic matrix factorization. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS'08)*. 1257–1264.

Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. 2012. CLiMF: Learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the 6th ACM conference on Recommender Systems (RecSys'12)*. 139–146.

Ajit P. Singh and Geoffrey J. Gordon. 2008. Relational learning via collective matrix factorization. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08)*. 650–658.

Nathan Srebro, Jason D. M. Rennie, and Tommi S. Jaakkola. 2004. Maximum-margin matrix factorization. In *Annual Conference on Neural Information Processing Systems (NIPS'04)*. 1329–1336.

Harald Steck. 2010. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'10)*. 713–722.

Chang Tan, Qi Liu, Enhong Chen, Hui Xiong, and Xiang Wu. 2014. Object-oriented travel package recommendation. *ACM Trans. Intell. Syst. Technol.* 5, 3 (2014), 43:1–43:26.

Jiliang Tang, Xia Hu, and Huan Liu. 2013. Social recommendation: A review. *Social Netw. Anal. Min.* 3, 4 (2013), 1113–1133.

Markus Weimer, Alexandros Karatzoglou, Quoc V. Le, and Alexander J. Smola. 2007. COFI RANK - Maximum margin matrix factorization for collaborative ranking. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS'07)*. 1593–1600.

Markus Zanker and Markus Jessenitschnig. 2009. Collaborative feature-combination recommender exploiting explicit and implicit user feedback. In *Proceedings of the 2009 IEEE Conference on Commerce and Enterprise Computing (CEC'09)*. 49–56.

Lili Zhao, Zhongqi Lu, Sinno J. Pan, and Qiang Yang. 2016. Matrix factorization+ for movie recommendation. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI'16)*.

Yi Zhen, Wu-Jun Li, and Dit-Yan Yeung. 2009. TagiCoFi: Tag informed collaborative filtering. In *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys'09)*. 69–76.

Yu Zheng. 2015. Methodologies for cross-domain data fusion: An overview. *IEEE Trans. Big Data* 1, 1 (2015), 16–33.

Received August 2016; revised December 2016; accepted January 2017