RaPare: A Generic Strategy for Cold-Start Rating Prediction Problem

Jingwei Xu, Yuan Yao, Hanghang Tong, Xianping Tao, Jian Lu

Abstract—In recent years, recommender system is one of indispensable components in many e-commerce websites. One of the major challenges that largely remains open is the *cold-start* problem, which can be viewed as a barrier that keeps the cold-start users/items away from the existing ones. In this paper, we aim to break through this barrier for cold-start users/items by the assistance of existing ones. In particular, inspired by the classic Elo Rating System, which has been widely adopted in chess tournaments; we propose a novel rating comparison strategy (RaPare) to learn the latent profiles of cold-start users/items. The center-piece of our RaPare is to provide a fine-grained calibration on the latent profiles of cold-start users/items by exploring the differences between cold-start and existing users/items. As a generic strategy, our proposed strategy can be instantiated into existing methods in recommender systems. To reveal the capability of RaPare strategy, we instantiate our strategy on two prevalent methods in recommender systems, i.e., the matrix factorization based and neighborhood based collaborative filtering. Experimental evaluations on five real data sets validate the superiority of our approach over the existing methods in cold-start scenario.

Index Terms—Recommender systems, Cold-start problem, Rating comparison strategy.

1 Introduction

Since the concept of recommender systems was proposed in 1997 [1], both industry and academia have provided their contribution to the improvement of quality and efficiency for recommender systems. As one of the major components of e-commerce and social websites, recommender system has become an inalienable part of these websites. During the last decade, many mainstream e-commerce companies have reported significant profit growth by integrating recommender systems into their applications [2], [3], [4], [5], [6].

Despite the success of existing recommender systems all over the world, the *cold-start* problem [7], [8], i.e., how to make proper recommendations for cold-start users or cold-start items, largely remains a daunting dilemma. On one hand, cold-start users (e.g., who have rated no more than 10 items) and cold-start items (e.g., which have received no more than 10 ratings) occupy a large proportion in many real applications such as Netflix [9]. On the other hand, the effectiveness of the existing recommendation approaches (e.g., collaborative filtering) largely depends on the sufficient amount of historical ratings, and hence these approaches might quickly become ineffective for cold-start users/items that only have few ratings.

To date, many collaborative filtering methods have been proposed to mitigate the cold-start problem, and these efforts can be divided into three classes. In the first class, a well designed *interview process* is introduced for cold-start users [10]. During this interview process, a set of items are provided for the cold-start users to express their opinions. The main disadvantage of methods in this class is the additional burdens incurred by the interview process. Methods in the second class resort to *side information* such as the user/item attributes [11] and social relationships [12] for the cold-start problem. The advantage is that these methods could be applicable for a new user/item with not rating at all. However, they rely on the access of such side information. These methods are inapplicable when the information is not available due to some reasons (e.g., privacy issue, user's social network structure not existing [12]), and has a higher computational cost compared with its side information free counterpart. In the third class, the cold-start problem is tackled in a *dynamic manner*. The intuition is that, compared to existing users/items, ratings for cold-start users/items may be more valuable to improve the accuracy of recommendation for these cold-start users/items; consequently, methods in this class aim to provide fast recommendations for cold-start users/items specifically, and then dynamically and efficiently adjust their latent profiles as they give/receive new ratings. Existing methods in this class include the incremental singular value decomposition (iSVD) method [13] and the incremental matrix factorization method [14], [15], etc. However, methods in the third class cannot serve users with no rating in the recommender system.

Compared with methods in the first two classes, the methods with dynamic view of the cold-start problem do not incur additional interview burden or rely on the access of side information, and thus become the focus of this paper.

In particular, we make the following analogy, i.e., to view the cold-start problem as a barrier between the cold-start users/items and the existing ones, and such a barrier could be broken with the assistance of existing users/items. To this end, we propose a novel rating comparison strategy

- *Jingwei Xu, Yuan Yao, Xianping Tao and Jian Lu are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210046, China.*
  E-mail: jingwei.xu, yyao@smail.nju.edu.cn; txp, lj@nju.edu.cn.
- *Hanghang Tong is with the School of Computing, Informatics, Decision Systems Engineering, Arizona State University, Phoenix, AZ 85281 U.S.A.*
  E-mail: hanghang.tong@asu.edu

(RAPARE) which can calibrate the latent profiles for cold-start users/items. Take cold-start user as an example, when a cold-start user gives a rating on an item, we first compare this rating with the existing ratings (which are from existing users) on this item. Then, we adjust the profile of the cold-start user based on the outcomes of the comparisons. Our rating comparison strategy (RAPARE) is inspired by the Elo Rating System [16] which has been widely used to calculate players' ratings in many different types of match systems, such as chess tournaments, FIFA, ATP, MLB and even some online competition sites (e.g., TopCoder).

The main contributions of this paper are summarized as follows:

- We propose a novel and generic rating comparison strategy RAPARE to serve for the cold-start problem. We formulate the strategy as an optimization problem. The key idea of RAPARE is to exploit the knowledge from existing users/items to help calibrate the latent profiles of cold-start users/items.
- We instantiate the proposed generic RAPARE strategy on both matrix factorization based (RAPARE-MF) and neighborhood based (RAPARE-KNN) collaborative filtering, together with algorithms to solve them.
- We present the algorithm analysis for RAPARE strategy and its instantiations on aspects of effectiveness and efficiency.
- We conduct extensive experimental evaluations on five real data sets, showing that our approach (1) outperforms several benchmark collaborative filtering methods and online updating methods in terms of prediction accuracy for cold-start scenario; (2) earns better quality-speed balance while enjoying a linear scalability.

The rest of the paper is organized as follows. In Section 2, we provide the problem statement. In Section 3, we describe the proposed rating comparison strategy. In Section 4, we present the proposed models for cold-start scenario, give the comprehensive algorithm analyses. In Section 5, we present the experimental results. In Section 6, we review related work. Finally, we conclude the paper in Section 7.

2 PROBLEM STATEMENT

In this section, we present the notations and the problem statement of recommending items to cold-start users and recommending cold-start items to users.

We list the main notations that are used throughout the paper in Table 1. Suppose we have sets of users $\mathcal{U}$, items $\mathcal{I}$ and ratings $\mathcal{R}$. Let u, v represent the users, and i, j represent the items, respectively. Then, $r_{ui} \in \mathcal{R}$ is the rating of user u for item i accordingly. For the scenario of cold-start user problem, we call the users who have given more than a certain quantity of ratings (e.g., 10 ratings) as existing users and the rest as cold-start users. Similarly, we call the items that have received more than a certain quantity of ratings (e.g., 10 ratings) as existing items and the rest as cold-start items. Meanwhile, $\mathcal{R}_c$ and $\mathcal{R}_w$ denote the sets of ratings that belong to cold-start users and existing users, respectively. We use $\mathcal{R}_w^i$ to represent the set of ratings on item i from existing users.

Table 1
Symbols.

Symbol	Definition and Description
u, v	users
i, j	items
r_{ui}	real rating from user u to item i
$\hat{r}_{ui}$	the predicted rating from user u to item i
$\mathcal{U}, \mathcal{I}, \mathcal{R}$	the set of users, items, and ratings, respectively
$\mathcal{I}(u)$	the set of items that have been rated by user u
$\mathcal{U}(i)$	the set of users who have rated item i
$\mathcal{R}_c$	the set of ratings by cold-start users
$\mathcal{R}_w$	the set of ratings by existing users
$\mathcal{R}_w^i$	the set of ratings on item i from existing users
$\mathcal{T}$	the set of observed ratings
$\mathcal{E}$	the set of ratings for evaluation
p_u, q_i	the latent profiles for user u and item i, respectively
$\mathcal{P}_c$	the set of latent profiles for cold-start users
$\mathcal{Q}$	the set of latent profiles for items

Based on the notations, we have the following problem definition for recommending items to cold-start users. Similar notations and definition can be derived for the cold-start item problem, and thus are omitted for brevity.

Problem 1. *The Cold-Start User Problem*

Given: *(1) the existing ratings R_w from existing users, (2) a new rating r_{uj} from a cold-start user u to item j, and (3) an item i ($i \neq j$);*
Find: *the estimated rating $\hat{r}_{ui}$ from user u to item i.*

As we can see from the definition, the input of our problem includes the existing ratings from existing users, as well as the new ratings from cold-start users. No side information is needed. When a new rating from a cold-start user arrives, we aim to immediately update the estimated rating $\hat{r}_{ui}$ for any given item i. The estimated rating indicates to what extent the cold-start user u would prefer to an item i.

In recommender systems, it is relatively difficult to draw proper latent profiles for the cold-start users/items due to the lack of sufficient historical ratings from them. We pay special attention to the new coming ratings from these cold-start users. Intuitively, these new ratings are more important for cold-start users/items. However, many existing methods treat the existing users/items and cold-start users/items in the same way. Additionally, these methods are usually static, i.e., they need to retrain the model when new ratings arrive which may be computationally expensive. Therefore, special treatments are needed for the cold-start users/items to adjust their latent profiles dynamically and efficiently.

Preliminary #1. To date, matrix factorization based collaborative filtering has been one of the most dominate methods in recommender systems. In detail, matrix factorization (MF) [17] assumes that users' opinions to items are based on the latent profiles for both users and items. With this assumption, MF projects both users and items into a joint latent factor space. The latent factors in the latent space can be seen as the latent profiles for users/items. To predict the rating $\hat{r}_{ui}$ from user u to item i, we only need to compute $\hat{r}_{ui}$ as

$$\hat{r}_{ui} = \mathbf{p}_u^{\mathsf{T}} \cdot \mathbf{q}_i \tag{1}$$

where vectors $\mathbf{p}_u$ and $\mathbf{q}_i$ are latent factor vectors for user u and item i, respectively. To learn these latent vectors, the following optimization formulation is usually constructed

$$\min_{\mathbf{q}*,\mathbf{p}*} \sum_{r_{ui}\in\mathcal{T}} (r_{ui} - \mathbf{p}_u^T \cdot \mathbf{q}_i)^2 + \lambda(||\mathbf{q}_i||^2 + ||\mathbf{p}_u||^2) \quad (2)$$

where square loss is used as the loss function, and *L2* regularization is to avoid over-fitting.

In Eq. (2), the learned latent vectors may be less accurate for cold-start users/items due to the lack of sufficient ratings.

Preliminary #2. K-Nearest-Neighbors method (KNN) is one of popular approaches in neighborhood based collaborative filtering. The key of K-Nearest-Neighbors method is to calculate the similarities between users or items. There are two kinds of KNN (i.e., user-based KNN and item-based KNN) in recommender systems based on the type of similarity calculation.

User-based KNN: The key intuition of user-based KNN is that users with similar tastes may give the similar ratings to the same item. Calculating the similarity between each pair of the given users is the key part of this method. We choose the adjusted cosine similarity [18] from existing similarity measurements in recommender systems

$$w_{u,v} = \frac{\sum_{i\in\mathcal{I}}(r_{ui} - \overline{r}_u)(r_{vi} - \overline{r}_v)}{\sqrt{\sum_{i\in\mathcal{I}}(r_{ui} - \overline{r}_u)^2}\sqrt{\sum_{i\in\mathcal{I}}(r_{vi} - \overline{r}_v)^2}} \quad (3)$$

where $w_{u,v}$ is the similarity between user u and user v, and $\overline{r}_u$ and $\overline{r}_v$ are the mean values of ratings from user u and user v, respectively. Then, we have the following rating prediction method with mean-centering normalization [19]

$$\hat{r}_{ui} = \overline{r}_u + \frac{\sum_{v\in\mathcal{U}(i)\bigwedge v\neq u} w_{u,v}(r_{vi} - \overline{r}_v)}{\sum_{v\in\mathcal{U}(i)\bigwedge v\neq u} |w_{u,v}|} \quad (4)$$

Item-based KNN: The key of item-based KNN is on the similarity calculation of items. We also choose the adjusted cosine similarity to calculate similarities

$$w_{i,j} = \frac{\sum_{u\in\mathcal{U}}(r_{ui} - \overline{r}_i)(r_{uj} - \overline{r}_j)}{\sqrt{\sum_{u\in\mathcal{U}}(r_{ui} - \overline{r}_i)^2}\sqrt{\sum_{u\in\mathcal{U}}(r_{uj} - \overline{r}_j)^2}} \quad (5)$$

where $w_{i,j}$ is the similarity between item i and item j, $\mathcal{U}$ is the user set, and $\overline{r}_i$ and $\overline{r}_j$ are the mean values of ratings on item i and item j, respectively. Similarly, the rating prediction with mean-centering normalization is as follows

$$\hat{r}_{ui} = \overline{r}_i + \frac{\sum_{j\in\mathcal{I}(u)\bigwedge j\neq i} w_{i,j}(r_{uj} - \overline{r}_j)}{\sum_{j\in\mathcal{I}(u)\bigwedge j\neq i} |w_{i,j}|} \quad (6)$$

Preliminary #3. Elo Rating System, which is first adopted in chess tournament, can be used to measure the relative skill levels between players in a certain competition. The basic idea behind Elo Rating System is that a player's rating (skill level) is determined by the competition outcomes against her opponents and the ratings of these opponents. For example, a player's rating will be greatly changed if she wins an opponent whose rating is much higher or if she loses to an opponent who has a much lower rating. In other words, the system implicitly aims to minimize the difference between the expected and the actual outcome of competitions.

To this end, we pay special attention to the cold-start users/items, and aim to calibrate the latent profiles for cold-start users/items with the help of the existing users/items. We will describe how to calibrate the latent profiles for cold-start users/items in the following section.

3 The Rating Comparison Strategy

In this section, we first present the core idea of our rating comparison strategy RaPare, and then formulate the strategy as an optimization formulation. The probabilistic interpretation of RaPare presents in Appendix.

3.1 The Core Idea of RaPare Strategy

Our goal is to break through the barrier between cold-start users and existing users by the assistance of existing users. Specifically, we achieve this goal by borrowing the idea of rating comparison from Elo Rating System. That is, we use the difference between the *expected* result and the *actual* result from the rating comparison strategy to calibrate the latent profiles of cold-start users.

To start the calibration, we need to first create a *competition* between a cold-start user and a selected existing user over a given item. Suppose that u is a cold-start user who has just rated item i, and v is an existing user who rated item i in the past. Then, user u and user v have a competition in terms of item i. Next, we need to compare the expected result and the actual result of the competition. Still using the example above, the expected result of this competition can be calculated as the difference between $\hat{r}_{ui}$ and r_{vi}. Here, $\hat{r}_{ui}$ is estimated based on the latent profiles of user u (which is also the parameters that we aim to estimate). In the meanwhile, with the actual rating r_{ui}, we can have the actual result of the competition, which is the difference between r_{ui} and r_{vi}. Finally, based on the expected result and actual result of the competition, we may update the latent profiles of user u by following a similar strategy as in Elo Rating System. That is, the farther the expected result of competition deviates from its actual result, the larger the latent profiles of user u will be changed.

In reality, when a cold-start user gives a rating on an item, there could be multiple existing users who have rated the same item. We then need to create multiple competitions/comparisons and update the latent profiles of the cold-start user multiple times. From the perspective of optimization, the RaPare strategy is equivalent to minimize the difference between the expected result and actual result by learning/calibrating the the latent profiles of cold-start users. Therefore, we can minimize the following equation

$$\sum_{r_{ui}\in\mathcal{R}_c}\sum_{r_{vi}\in\mathcal{R}_w^i} (\underbrace{g(r_{ui}, r_{vi})}_{actual\ diff} - \underbrace{g(\hat{r}_{ui}, r_{vi})}_{expected\ diff})^2 \quad (7)$$

where g is the loss/difference function for a rating comparison (either actual difference or expected difference), and the square loss is used as the loss function for the comparison between expected result and actual result. As we can see from the above formulation, for a given cold-start user, we can employ the potentially large amount of existing ratings

from existing users to help calibrate the latent profiles of this cold-start user.

In this paper, we put our focus on the g function. We provide several candidates, including the *linear* difference, the *logistic* difference, and the *elo* difference. For the first two candidates, they are defined as

$$g_{linear}(r_{ui}, r_{vi}) = r_{ui} - r_{vi} \tag{8}$$

$$g_{logistic}(r_{ui}, r_{vi}) = \frac{1}{1 + e^{-(r_{ui} - r_{vi})}} \tag{9}$$

We use $g(r_{ui}, r_{vi})$ as an example in the above equations, and equations for $g(\hat{r}_{ui}, r_{vi})$ can be similarly obtained by substituting r_{ui} with $\hat{r}_{ui}$. For the *elo* candidate, two difference functions are used to compute $g(r_{ui}, r_{vi})$ and $g(\hat{r}_{ui}, r_{vi})$, which are directly borrowed from Elo Rating System

$$g_{elo_{actual}}(r_{ui}, r_{vi}) = \begin{cases} 1, & \text{if } r_{ui} > r_{vi} \\ 0.5, & \text{if } r_{ui} = r_{vi}, \ and \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

$$g_{elo_{expected}}(\hat{r}_{ui}, r_{vi}) = \frac{1}{1 + e^{-(\hat{r}_{ui} - r_{vi})}} \tag{11}$$

where $g_{elo_{actual}}$ is used for calculating the actual result of the competition, and $g_{elo_{expected}}$ is used for the expected result. Notice that $g_{elo_{expected}}$ is actually the same with $g_{logistic}$.

3.2 Optimization Formulation and Inference

In this part, we derive the optimization problem for RAPARE strategy, and propose an inference algorithm to learn the parameters. Formally, we have the following optimization problem for RAPARE

$$\arg\min_{\mathcal{P}_c} \text{RAPARE-OPT}(\mathcal{R}_w, \mathcal{R}_c, \mathcal{P}_c) \tag{12}$$

with

$$\text{RAPARE-OPT} = \sum_{r_{ui} \in \mathcal{R}_c} \sum_{r_{vi} \in \mathcal{R}_w^i} (\underbrace{g(r_{ui}, r_{vi})}_{actual\ diff} - \underbrace{g(\hat{r}_{ui}, r_{vi})}_{expected\ diff})^2 + \lambda \sum_{p_u \in \mathcal{P}_c} p_u^2$$

where $\mathcal{R}_w$ is the set of ratings from existing users, $\mathcal{R}_w^i$ is the set of ratings from existing users to item i, $\mathcal{R}_c$ is the set of ratings from cold-start users, and $\mathcal{P}_c$ is the set of latent profiles for cold-start users. To avoid the overfitting problem, we also add a regularization term which is controlled by λ in the formulation.

As we can see from Eq. (12), given the input of the existing ratings from existing users, the arrival ratings from cold-start users, and the initial latent profiles of cold-start users, the RAPARE strategy aims to minimize the loss function defined in Eq. (12) by adjusting $\mathcal{P}_c$. The reason we focus on calibrating $\mathcal{P}_c$ is based on the observation that a few number of arrival ratings from cold-start users would not change the latent profiles of the corresponding items dramatically, but could have a much bigger impact on the latent profiles of cold-start users. Once $\mathcal{P}_c$ is solved, we can estimate the rating from a cold-start user by some existing rating prediction functions (e.g., Eq. (1)).

The Alg. 1 shows the calibrating process for latent profiles of cold-start users. For a given rating r_{ui} (from the cold-start user u to item i), we can run the RAPARE strategy to calibrate her latent profile by a given existing rating (from one of the existing users to the item i). We treat this process as a basic update unit. Specifically, we have the following general updating rule for Eq. (12)

Algorithm 1: Learning RAPARE

Input: ratings from existing users $\mathcal{R}_w$, and ratings from cold-start users $\mathcal{R}_c$
Output: latent factors of cold-start users $\mathcal{P}_c$

```
initialize P_c
repeat
    for r_ui ∈ R_c do
        for r_vi ∈ R_w^i do
            update p_u ← p_u − α∇_{p_u} as defined in Eq. (14)
until convergence;
return P_c
```

$$p_u \leftarrow p_u - \alpha \nabla_{p_u} \tag{13}$$

where the basic update unit ∇_{p_u} for the *linear* and the *logistic* differences is

$$\nabla_{p_u} = \frac{\partial \text{RAPARE-OPT}}{\partial p_u} = -2 \cdot (g(r_{ui}, r) - g(\hat{r}_{ui}, r)) \cdot \frac{\partial g(\hat{r}_{ui}, r)}{\partial p_u} + 2 \cdot \lambda \cdot p_u \tag{14}$$

where

$$\frac{\partial g_{linear}}{\partial p_u} = \frac{\partial \hat{r}_{ui}}{\partial p_u} \tag{15}$$

for the *linear* difference function, and

$$\frac{\partial g_{logistic}}{\partial p_u} = \frac{e^{-(\hat{r}_{ui} - r_{vi})}}{(1 + e^{-(\hat{r}_{ui} - r_{vi})})^2} \cdot \frac{\partial \hat{r}_{ui}}{\partial p_u} \tag{16}$$

for the *logistic* difference function.

For the *elo* difference, we directly use the difference between expected result and actual result as the basic update unit. The equation is as follow

$$\nabla_{p_u} = -2 \cdot (g_{elo_{actual}} - g_{elo_{expected}}) \cdot \frac{\partial \hat{r}_{ui}}{\partial p_u} + 2 \cdot \lambda \cdot p_u \tag{17}$$

4 INSTANTIATIONS OF RAPARE STRATEGY

In this section, we instantiate RAPARE with two existing collaborative filtering methods (i.e., matrix factorization and neighborhood based methods) in cold-start scenario. To demonstrate the potential applicability of the proposed RAPARE strategy for classic rating prediction problem, we also propose RAPARE-UNIVERSAL by assembling RAPARE and matrix factorization. The discussion could be found in Appendix.

4.1 RaPare Instantiation with Matrix Factorization

To instantiate RAPARE with MF, we only need to specify the derivative computation of $\frac{\partial \hat{r}_{ui}}{\partial \mathbf{p_u}}$ as shown in Eq. (22) - (24). As described in Section 2, matrix factorization uses Eq. (1) to predict ratings. The derivative of each factor $p_{u,f}$ in latent profile $\mathbf{p_u}$ is

$$\frac{\partial \hat{r}_{ui}}{\partial p_{u,f}} = \frac{\partial \hat{r}_{ui}}{\partial \mathbf{p_u}} \cdot \frac{\partial \mathbf{p_u}}{\partial p_{u,f}} = q_{i,f} \tag{18}$$

where $q_{i,f}$ is the corresponding factor in item i's latent profile $\mathbf{q_i}$. In this paper, we call RAPARE instantiation with matrix factorization as RAPARE-MF.

Based on the above equation, we can directly apply matrix factorization via Alg. 1. Here, we further propose a fast learning algorithm for RAPARE-MF to solve the optimization problem in Eq. (12), which is based on the following two key observations of the inherent structure in the optimization formulation. First, there are usually a small set of possible ratings (e.g., 1–5 stars) for most recommender systems. Second, in Eq. (12), the contribution of the ratings from different existing users to item i is equal to each other if they share the same rating value on item i. Specifically, the optimization equation in Eq. (12) could be re-written as

$$\text{RAPARE-OPT*} = \sum_{r_{ui} \in R_c} \sum_{r=1}^{r_{max}} |\Omega_{\mathcal{R}_w^i, r}|(g(r_{ui}, r) - g(\hat{r}_{ui}, r))^2 + \lambda \sum_{\mathbf{p_u} \in \mathcal{P}_c} ||\mathbf{p_u}||_F^2 \quad (19)$$

where $\mathcal{P}_c$ is the set of latent profiles of cold-start users, r_{max} is the maximal rating scale (e.g., $r_{max} = 5$ in Netflix), and $|\Omega_{\mathcal{R}_w^i, r}|$ indicates the number of ratings in $\mathcal{R}_w^i$ with the value r.

For a given rating r_{ui} (from the cold-start user u to item i), we can now aggregate the existing ratings (from the existing users to item i) with the same value, to the basic update unit. Specially, we have the following general updating rule for Eq. (19)

$$p_{u,f} \leftarrow p_{u,f} - \alpha \nabla^*_{p_{u,f}} \quad (20)$$

where the basic update unit $\nabla^*_{p_{u,f}}$ for the *linear* and the *logistic* differences is

$$\nabla^*_{p_{u,f}} = \frac{\partial \text{RAPARE-OPT*}}{\partial p_{u,f}} = -2 \cdot |\Omega_{\mathcal{R}_w^i, r}| \cdot (g(r_{ui}, r) - g(\hat{r}_{ui}, r)) \cdot \frac{\partial g(\hat{r}_{ui}, r)}{\partial p_{u,f}} + 2 \cdot \lambda \cdot p_{u,f} \quad (21)$$

where

$$\frac{\partial}{\partial p_{u,f}} g_{linear} = q_{i,f} \quad (22)$$

for the *linear* difference function, and

$$\frac{\partial}{\partial p_{u,f}} g_{logistic} = \frac{e^{-(\hat{r}_{ui} - r)}}{(1 + e^{-(\hat{r}_{ui} - r)})^2} \cdot q_{i,f} \quad (23)$$

for the *logistic* difference function.

For the *elo* difference, the basic update unit is as follows

$$\nabla^*_{p_{u,f}} = -2 \cdot |\Omega_{\mathcal{R}_w^i, r}|(g_{elo_{actual}} - g_{elo_{expected}}) \cdot q_{i,f} + 2 \cdot \lambda \cdot p_{u,f} \quad (24)$$

We summarize our fast learning algorithm in Alg. 2, where we use mini-batch gradient descent method to learn the parameters.

Implementation Details: Initialization. Here, we discuss the initialization methods for $\mathcal{P}_c$ in RAPARE-MF. Although the latent profiles in $\mathcal{P}_c$ can be randomly initialized, a better initialization (e.g., by leveraging the small amount of ratings from the cold-start users) can yield optima efficiency. In this work, we consider the following three initialization methods:

Algorithm 2: Fast learning RAPARE-MF

Input: ratings from existing users $\mathcal{R}_w$, ratings from cold-start users $\mathcal{R}_c$, item latent profiles $\mathcal{Q}$, and the maximal rating scale r_{max}
Output: latent profiles of cold-start users $\mathcal{P}_c$

```
initialize P_c
repeat
    for r_ui ∈ R_c do
        for r ← 1, ..., r_max do
            for f ← 1, ..., k do
                update p_{u,f} ← p_{u,f} − α∇*_{p_{u,f}} as defined in Eq. (20)
until convergence;
return P_c
```

- *Random initialization*. The simplest initialization of RAPARE-MF is to randomly initialize each $\mathbf{p_u}$ in $\mathcal{P}_c$.
- *Average initialization*. For a cold-start user u and an item i that u has rated, this average method first finds the set of existing users who have given the same rating to item i as u does, and then computes the average $\overline{\mathbf{p}}_\mathbf{u}(i)$ over these existing users

$$\overline{\mathbf{p}}_\mathbf{u}(i) = \frac{\sum_{v \in \mathcal{U}(i) \bigwedge v \neq u} \delta^i_{u,v} \mathbf{p_v}}{\sum_{v \in \mathcal{U}(i) \bigwedge v \neq u} \delta^i_{u,v}} \quad (25)$$

  where $\mathcal{U}(i)$ is the set of existing users who have rated item i, and $\delta^i_{u,v}$ is an indicator to indicate whether user u and user v have the same rating to item i. Then, this average method further averages the computed $\overline{\mathbf{p}}_\mathbf{u}(i)$ over all the items that user u has rated, which gives the initial latent profile for the cold-start user u

$$\mathbf{p_u} = \frac{\sum_{i \in \mathcal{I}(u)} \overline{\mathbf{p}}_\mathbf{u}(i)}{|\mathcal{I}(u)|} \quad (26)$$

  where $\mathcal{I}(u)$ is the set of items that user u has already rated.
- *Clustering-based initialization*. In this method, we first cluster the existing users into several groups (e.g., by k-means algorithm), each of which is accompanied with a representative latent profile. During initialization, this method chooses the best representative latent profile: if the cold-start user u has rated the item i, this method will test all the representative latent profiles from each cluster, and choose the one with the lowest error as $\mathbf{p_u}$.

Connections to existing methods. Notice that the RAPARE-MF model with *linear* difference function is closely connected to the traditional MF model. The expanded form of RAPARE-MF model with *linear* difference can be written as

$$\begin{aligned} &\text{RAPARE-OPT*} \\ &= \sum_{r_{ui} \in \mathcal{R}_c} \sum_{r_{vi} \in \mathcal{R}_w^i} ((r_{ui} - r_{vi}) - (\hat{r}_{ui} - r_{vi}))^2 + \lambda \sum_{\mathbf{p_u} \in \mathcal{P}_c} ||\mathbf{p_u}||_F^2 \\ &= \sum_{r_{ui} \in \mathcal{R}_c} |\mathcal{R}_w^i|(r_{ui} - \hat{r}_{ui})^2 + \lambda \sum_{\mathbf{p_u} \in \mathcal{P}_c} ||\mathbf{p_u}||_F^2 \end{aligned}$$

Algorithm 3: Learning user-based RAPARE-KNN

Input: set of existing users $\mathcal{U}_w$, ratings from existing users $\mathcal{R}_w$, ratings from cold-start users $\mathcal{R}_c$
Output: Similarity matrix $\mathbf{W_c}$ that Cold-start users to existing users

initialize $\mathbf{W_c}$
repeat
  for $r_{ui} \in \mathcal{R}_c$ **do**
    for $v \in \mathcal{U}_w(i)$ **do**
      update $w_{u,v} \leftarrow w_{u,v} - \alpha \nabla_{w_{u,v}}$ as defined in Eq. (27)
until *convergence*;
return $\mathbf{W_c}$

Algorithm 4: Learning item-based RAPARE-KNN

Input: set of items $\mathcal{I}$, ratings from items $\mathcal{R}_i$, ratings from cold-start users $\mathcal{R}_c$
Output: Item-item similarity matrix $\mathbf{W_i}$

initialize $\mathbf{W_i}$
repeat
  for $r_{ui} \in \mathcal{R}_c$ **do**
    for $j \in \mathcal{I}(u)$ **do**
      update $w_{i,j} \leftarrow w_{i,j} - \alpha \nabla_{w_{i,j}}$ as defined in Eq. (28)
until *convergence*;
return $\mathbf{W_i}$

As we can see, similar to the traditional MF method [15], the above formulation also aims to optimize over the square loss between the actual rating and the predicted rating. In other words, our RAPARE-MF with *linear* difference function can be viewed as a weighted matrix factorization method for cold-start users.

4.2 RaPare Instantiation with KNN

In this part, we propose a new vision to get the proper similarity by learning it instead of calculating it, so that RAPARE strategy could be directly instantiated with KNN as RAPARE-KNN. Specifically, we treat similarities between users/items as a special kind of latent profiles. For example, the latent profile $p_{u,v}$ in user-based RAPARE-KNN represents the similarity between user u and user v. So, as the derivatives of Eq. (4) and Eq. (6), the Eq. (27) and Eq. (28) are used for updating the latent profiles of users and items (i.e., user-user and item-item similarity matrices).

$$\frac{\partial \hat{r}_{ui}}{\partial w_{u,v}} = \frac{(r_{vi} - \overline{r}_v)\sum_k |w_{u,k}| - \sum_k (w_{u,k}(r_{ki} - \overline{r}_k))}{(\sum_k |w_{u,k}|)^2} \quad (27)$$

$$\frac{\partial \hat{r}_{ui}}{\partial w_{i,j}} = \frac{(r_{uj} - \overline{r}_j)\sum_k |w_{i,k}| - \sum_k (w_{i,k}(r_{uk} - \overline{r}_k))}{(\sum_k |w_{i,k}|)^2} \quad (28)$$

We can then use Eq. (27)/Eq. (28) to update the latent profiles of users/items (i.e., user-user/item-item similarity matrices). We use stochastic gradient descent method to learn the parameters in RAPARE-KNN, and the algorithms are summarized in Alg. 3 and Alg. 4.

4.3 Algorithm Analysis

Here, we analyze the effectiveness and efficiency of our algorithms.

The effectiveness of proposed RAPARE is summarized in Lemma 4.1. Following the proof of the convergence for gradient descent algorithm [20], this Lemma shows that RAPARE finds the global minimum for the proposed model.

Lemma 4.1 (Effectiveness of RAPARE). *Fixing the ratings in $\mathcal{R}_w$ and $\mathcal{R}_c$,* RAPARE *finds the global minimum for the optimization problem in Eq.* (12) *with fixed step size satisfying $\alpha \leq \frac{1}{L}$.*

Proof. Eq. (12) is convex and denoted by $f(x)$, where x represents $\mathcal{P}_c$. $\nabla f(x)$ is Lipschitz continuous with constant L [1]. We have

$$f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{L}{2}||y - x||^2, \forall x, y$$

Let $x^+ = x - \alpha \nabla f(x)$, take $0 < \alpha \leq \frac{1}{L}$, and we have x^* such that $f(x^*) = \min f(x)$

$$\begin{aligned} f(x^+) &\leq f(x^*) + \nabla f(x)^T (x - x^*) + \frac{\alpha}{2}||\nabla f(x)||^2 \\ &= f(x^*) + \frac{1}{2\alpha}(||x - x^*||^2 - ||x^+ - x^*||^2) \end{aligned}$$

Summing over iterations

$$\sum_{i=1}^{k} (f(x^{(i)}) - f(x^*)) \leq \frac{1}{2\alpha}||x^{(0)} - x^*||^2$$

Since $f(x^{(k)})$ is non-increasing

$$f(x^{(k)}) - f(x^*) \leq \frac{1}{k}\sum_{i=1}^{k}(f(x^{(i)}) - f(x^*)) \leq \frac{1}{2\alpha k}||x^{(0)} - x^*||^2$$

This implies that $\lim_{k \to \infty} f(x^{(k)}) - f(x^*) = 0$

□

As we discussed in section 4.1, the parameters of RAPARE-MF could be learned in a fast way with the modified optimization problem in Eq. (19). The equivalence of this modified optimization problem to the original optimization problem in Eq. (12) is summarized in Lemma 4.2. This Lemma shows that our proposed fast inference algorithm (Eq. (20)) could find the same solution as the straightforward gradient descent (Eq. (13)).

Lemma 4.2 (Equivalence of Fast Inference of RAPARE-MF). *For* RAPARE-MF*, the fast inference algorithm (Eq.* (20)*) could find the same solution as the straightforward gradient descent method (Eq.* (13)*).*

Proof. First, the goal of RAPARE-MF is to calibrate latent profiles of cold-start users while keeping the latent profiles of item unchanged. Thus, Eq. (19) is convex due to the latent profiles of items are fixed. In Alg. 2, step 3-6 defines the gradient descent process for each iteration that would never increase the cost of Eq. (19). Thus, the process could converge when the global minimum is reached.

Next, we only need to prove that the modified optimization equation (Eq. (19)) could reach the same global

1. http://moon.nju.edu.cn/people/jingweixu/static/proof.pdf

minimum solution as the original optimization equation (Eq. (12)). For this purpose, we have

$$
\begin{aligned}
&\text{RAPARE-OPT}\\
&= \sum_{r_{ui}\in\mathcal{R}_c}\sum_{r_{vi}\in\mathcal{R}_w^i}(g(r_{ui},r_{vi})-g(\hat{r}_{ui},r_{vi}))^2+\lambda\sum_{\mathbf{p_u}\in\mathcal{P}_c}||\mathbf{p_u}||_F^2\\
&= \sum_{r_{ui}\in\mathcal{R}_c}\sum_{r=1}^{r_{max}}\sum_{r_{vi}\in\mathcal{R}_w^i}\delta(r-r_{ui})(g(r_{ui},r_{vi})-g(\hat{r}_{ui},r_{vi}))^2\\
&\quad+\lambda\sum_{\mathbf{p_u}\in\mathcal{P}_c}||\mathbf{p_u}||_F^2\\
&= \sum_{r_{ui}\in\mathcal{R}_c}\sum_{r_{vi}\in\mathcal{R}_w^i}\delta(1-r_{ui})(g(r_{ui},r_{vi})-g(\hat{r}_{ui},r_{vi}))^2\\
&\quad\vdots\\
&\quad\sum_{r_{ui}\in\mathcal{R}_c}\sum_{r_{vi}\in\mathcal{R}_w^i}\delta(r_{max}-r_{ui})(g(r_{ui},r_{vi})-g(\hat{r}_{ui},r_{vi}))^2\\
&\quad+\lambda\sum_{\mathbf{p_u}\in\mathcal{P}_c}||\mathbf{p_u}||_F^2\\
&= \sum_{r_{ui}\in\mathcal{R}_c}|\Omega_{\mathcal{R}_w^i,1}|(g(r_{ui},r_{vi})-g(\hat{r}_{ui},r_{vi}))^2\\
&\quad\vdots\\
&\quad\sum_{r_{ui}\in\mathcal{R}_c}|\Omega_{\mathcal{R}_w^i,r_{max}}|(g(r_{ui},r_{vi})-g(\hat{r}_{ui},r_{vi}))^2\\
&\quad+\lambda\sum_{\mathbf{p_u}\in\mathcal{P}_c}||\mathbf{p_u}||_F^2\\
&= \sum_{r_{ui}\in R_c}\sum_{r=1}^{r_{max}}|\Omega_{\mathcal{R}_w^i,r}|(g(r_{ui},r)-g(\hat{r}_{ui},r))^2+\lambda\sum_{\mathbf{p_u}\in\mathcal{P}_c}||\mathbf{p_u}||_F^2\\
&=\text{RAPARE-OPT*}
\end{aligned}
$$

where $\delta(x)$ is the Dirac's delta function that returns 1 when x is 0, and returns 0 when x is not 0. Therefore, RAPARE-OPT is equivalent to RAPARE-OPT*, which completes the proof. □

The time complexity of the proposed RAPARE with two instantiations is summarized in Lemma 4.3. This Lemma shows that RAPARE-MF requires *linear time* for online calibrating latent profiles of cold-start users/items (e.g., step 3-6 in Alg. 2); RAPARE-KNN requires *quadratic time* for online calibrating latent profiles of cold-start users/items (e.g., step 3-5 in Alg. 3 and Alg. 4).

Lemma 4.3 (Time Complexity of RAPARE). *Fixing $\mathcal{R}_w$, $\mathcal{R}_c$, $\mathbf{Q}$ and r_{max}, (P1)* RAPARE-MF *requires $O(|\mathcal{R}_c|)$ time in Alg. 2; (P2)* RAPARE-KNN *requires $O(|\mathcal{R}_c|\cdot|\overline{\mathcal{U}_w(i)}|)$ and $O(|\mathcal{R}_c|\cdot|\overline{\mathcal{I}(u)}|)$ time for each iteration in Alg. 3 and Alg. 4).*

Proof. First, we prove (P1) for calibrating latent profiles of cold-start users via RAPARE-MF. For each iteration in Alg. 2, we need $O(\mathcal{R}_c)$ time for the loop that starts from step 3. The time cost for the loop that starts from step 4 is r_{max}, and step 5-6 costs $O(k)$ time for updating parameters. Therefore, the total time cost of the iteration of step 2-6 is $O(|\mathcal{R}_c|\cdot r_{max}\cdot k\cdot m_2)$, where m_2 is the maximum iteration number for Alg. 2. Notice that, r_{max}, k and m_2 are small constants, so the total time cost of Alg. 2 can be written as $O(|\mathcal{R}_c|)$.

Table 2
The statistics of the five data sets.

Data	Users	Items	Ratings	Rating Scale
MovieLens	6,040	3,706	1,000,209	[1,2,3,4,5]
EachMovie	72,916	1,628	2,464,792	[1,2,3,4,5]
Yelp	43,873	11,537	194,538	[1,2,3,4,5]
Amazon Automotive	851,433	320,117	1,373,794	[1,2,3,4,5]
Amazon Electronic	4,201,732	476,005	7,824,546	[1,2,3,4,5]

For RAPARE-MF, if we use the straightforward gradient descent method in Alg. 1, its average-case time complexity is $O(|\mathcal{R}_c|\cdot|\overline{\mathcal{R}_w^i}|)$, where $|\overline{\mathcal{R}_w^i}|$ is the average number of ratings from existing users to each item. Thus, Alg. 2 is much faster than Alg. 1 because $r_{max}\ll|\overline{\mathcal{R}_w^i}|$.

Next, we prove (P2) for calibrating latent profiles of cold-start users via RAPARE-KNN. For user-based RAPARE-KNN, we need $(O|\overline{\mathcal{U}_w(i)}|)$ time for updating parameters in step 4-5 of Alg. 3, where $|\overline{\mathcal{U}_w(i)}|$ is the average number of ratings from existing users to the item i. We need $O(|\mathcal{R}_c|\cdot|\overline{\mathcal{U}_w(i)}|)$ time for each iteration in Alg. 3. Therefore, the total time cost of Alg. 3 is $O(|\mathcal{R}_c|\cdot|\overline{\mathcal{U}_w(i)}|\cdot m_3)$, where m_3 is the maximum iteration number for Alg. 3. m_3 is small constant, so the time complexity of Alg. 3 will be $O(|\mathcal{R}_c|\cdot|\overline{\mathcal{U}_w(i)}|)$.

For item-based RAPARE-KNN, we need $O(|\overline{\mathcal{I}(u)}|)$ time for updating parameters in step 4-5 of Alg. 4, where $|\overline{\mathcal{I}(u)}|$ is the average number of ratings from the user u. We need $O(|\mathcal{R}_c|\cdot|\overline{\mathcal{I}(u)}|)$ time for each iteration in Alg. 4. Therefore, the total time cost of Alg. 4 is $O(|\mathcal{R}_c|\cdot|\overline{\mathcal{I}(u)}|\cdot m_4)$, where m_4 is the maximum iteration number in Alg. 4, and is a small constant. Therefore, the time complexity of Alg. 4 will be $O(|\mathcal{R}_c|\cdot|\overline{\mathcal{I}(u)}|)$. □

5 EXPERIMENTS

In this section, we present the experimental evaluations. All the experiments are designed to answer the following questions:

- *Effectiveness*: How accurate is the proposed approach for the cold-start scenario?
- *Efficiency*: How fast is the proposed approach for updating the latent profiles of users/items in cold-start scenario?

5.1 Experimental Setup

5.1.1 Data Sets and Evaluation Metrics

We use two real, benchmark data sets: *MovieLens*[2] and *EachMovie* and three data sets from real e-commerce websites: *Yelp*[3], *Amazon Automotive* and *Amazon Electronic*. The *MovieLens* data contains 1M ratings that are collected and published from the MovieLens website[4]. Each user in this data set rated at least 20 movies. *EachMovie* is organized by HP/Compaq Research. The *Yelp* data set is provided by Yelp.com that was used for Yelp Business Rating Prediction Competition in RecSys 2013. *Amazon Automotive* and *Amazon Electronic* are both crawled and organized by Julian

2. http://www.grouplens.org/datasets/movielens/
3. https://www.kaggle.com/c/yelp-recsys-2013/data
4. http://movielens.org

McAuley [21], [22] on Amazon.com. The statistics of the five data sets are summarized in Table 2.

As for evaluation metrics, we adopt the commonly used root mean square error (RMSE) for effectiveness comparison

$$RMSE = \sqrt{\frac{\sum_{r_{ui} \in \mathcal{E}} (\hat{r}_{ui} - r_{ui})^2}{|\mathcal{E}|}} \quad (29)$$

where test set $\mathcal{E}$ contains the ratings for evaluation, $\hat{r}_{ui}$ is the predicted rating from user u to item i, r_{ui} is the ground truth, and $|\mathcal{E}|$ represents the number of ratings in $\mathcal{E}$. For efficiency, we report the wall-clock time of the compared methods for updating the latent profiles of cold-start users/items.

5.1.2 Evaluation Protocol for cold-start scenario

Here, we describe how we evaluate the performance of our method for recommending items to cold-start users. Similar evaluation protocol can be applied to the cold-start items. Notice that our evaluation protocol follows the existing work [15].

We summarize the overall evaluation protocol for the cold-start users with the following descriptions.

1) Setup the cold-start user scenario
   - Find cold-start users, put them into set $\mathcal{U}_c$, and put all their ratings into the set $\mathcal{E}_c$. The rest users are considered as existing users
   - For RAPARE-MF, train the model for existing users via the MF method [17]. After this step, we can obtain the latent profiles of existing users and the latent profiles for all items. For RAPARE-KNN, we randomly initialize latent profiles for all cold-start users
2) Evaluate the cold-start user scenario
   - Create an empty set $\mathcal{R}_c$
   - for $n = 1, ..., 10$ do:
     - for each cold-start user $u \in \mathcal{U}_c$ do:
       * randomly pick up one of his ratings from the $\mathcal{E}_c$ and move it to $\mathcal{R}_c$
       * learn the latent profile of user u by the proposed methods RAPARE-MF/RAPARE-KNN
       * calculate the error between the predicted result and the actual result for the rest of u's ratings in the $\mathcal{E}_c$
     - calculate the error over all cold-start users

Since each user in *MovieLens* and *EachMovie* has more than 20 ratings, we randomly choose 25% users as cold-start users in these two data sets. For other three data sets, we select users that rated no more than 11 ratings as cold-start users for evaluation. To give the overview of the data sets over this evaluation protocol, the statistics of users and ratings for cold-start user scenario are listed in Table 3. Due to the difference between {*MovieLens*, *EachMovie*} and {*Amazon Automotive*, *Amazon Electronic* and *Yelp*}, the numbers of cold-start users in *MovieLens* and *EachMovie* are not changed during this evaluation process. Whereas the numbers of cold-start users in *Amazon Automotive*, *Amazon Electronic* and *Yelp* vary, because the numbers of ratings that cold-start users have in these data sets are not guaranteed to be equal. That is to say, for example, if we want to evaluate approaches on cold-start users that have already rated 5 ratings, all cold-start users that have only 4 ratings will be discarded.

Table 3
The statistics of users and ratings for the evaluation in the five data sets.

Data	*MovieLens*	*EachMovie*	*Yelp*	*Amazon Auto.*	*Amazon Elec.*
C.S. users	1,510	18,229	37,860	846,740	390,734
Ratio	25%	25%	92.6%	99.4%	82.1%
Ratings (C.S.)	237,851	982,140	88,156	1,287,877	6,950,369
Ratio	23.8%	39.8%	45.3%	93.7%	88.9%
W. users	4,530	54,687	3,011	4,692	85,270
Ratio	75%	75%	7.4%	0.6%	17.9%
Ratings (W.)	762,358	1,482,652	106,382	85,917	874,177
Ratio	76.2%	60.2%	54.7%	6.3%	11.1%

5.1.3 Compared Methods

In our evaluation protocol, the ratings in $\mathcal{R}_c$ are treated as the early ratings for the cold-start users/items in a dynamic manner. These ratings can also be regarded as the historical information for existing methods in a static manner. Here, we compare our RAPARE-MF and RAPARE-KNN with several classic collaborative filtering methods for both effectiveness and efficiency aspects. The compared methods include: *user/item based neighborhood methods* (i.e., user KNN and item KNN) [18], *incremental SVD* (iSVD) [13], *matrix factorization* (MF) [17], and *Online Update* which is proposed for updating the latent profiles of cold-start users/items only [15]. *Random initialization* is used for MF in this paper. The parameters are tuned through standard cross validation. We search through a set of values on the training set to find the optimal one for each method, and apply the optimal parameters to the test set. We set $\alpha = 0.01$, and $\lambda = 0.1$ for MF and Online Update. For RAPARE-MF, we assign $\alpha = 0.01$, and $\lambda = 100$. For RAPARE-KNN, we set $\alpha = 0.001$, and $\lambda = 0.001$. The dimension of latent profiles for MF, Online Update, and RAPARE-MF, are set to 20 in this paper.

5.2 Effectiveness Results

In this section, we present a series of experiment results on effectiveness evaluation for cold-start scenario.

A.1. Effectiveness Comparisons: RAPARE-MF. We first evaluate the effectiveness of the proposed RAPARE-MF with the compared methods over the cold-start scenario. We use *average initialization* as the initialization method for RAPARE-MF. The results for both cold-start users and cold-start items on the five data sets are shown in Fig. 1, Fig. 2 and Fig. 3. The x-axis of these three figures indicates the cold-start degree (i.e., the number of ratings given by each cold-start user/item), and the y-axis indicates the RMSE value. As the evaluation metric, smaller RMSE is better. In this paper, we only choose *logistic* to participate the effectiveness comparisons, since it performs better than other two difference functions. Besides, KNN and RAPARE-KNN are worse than the other methods. So we exclude these two methods in these series of comparisons and discuss them later.

There are several observations from the results on the *MovieLens* data (Fig. 1(a), Fig. 1(b)). First of all, in general, RAPARE-MF performs better than all the compared methods for both cold-start users and cold-start items. The iSVD method performs very well at the beginning of the evaluation. This is because iSVD fills in all the missing values in the rating matrix with the mean rating values of the corresponding items. As the evaluation continues, no significant improvement is observed from the iSVD method. The MF method and Online Update method perform relatively well in the compared methods. The RMSE of these two methods decreases as the evaluation continues. However, RAPARE-MF still outperforms these two methods in all cases. This is due to the special calibration on the latent profiles of cold-start users/items. In other words, our proposed method indeed helps decrease the prediction error. Notice that, small improvements in RMSE could gain practically significant improvement in recommendation [23]. Similar results are observed on the *EachMovie* data (Fig. 1(c) and Fig. 1(d)).

Next, we evaluate RAPARE-MF and compared approaches on *Amazon Automotive*, *Amazon Electronic* and *Yelp* data sets (Fig. 2 and Fig. 3). We evaluate RAPARE-MF, MF, Online Update and iSVD on *Yelp* data. For *Amazon Automotive* and *Amazon Electronic*, we exclude iSVD because of the incompatibility of iSVD with large data set. In general, RAPARE-MF performs better than other competitors for both cold-start user and cold-start item scenarios on all of these three data sets. In detail, RAPARE-MF performs better than MF method and Online Update method in all cases (Fig. 2(a) and Fig. 2(b)) on *Amazon Automotive*. Although MF performs slightly better than RAPARE-MF in cold-start user scenario (Fig. 2(c)) on *Amazon Electronic* at the beginning (i.e., number of ratings from cold-start user is 1 or 2), our proposed method still gains huge advantage in the whole process. Online Update method is worse than other two methods on these three data sets. The iSVD method performs relatively well on *Yelp* data set, but its disadvantage restricts its expansibility of usage to large data set.

Meanwhile, we perform statistical t-test over each specific number of ratings for cold-start user/item, which indicates that the improvement of RAPARE-MF is significant in almost all cases. For example, when the cold-start degree is 5, the p-value is less than 0.05 in all settings (i.e., two cold-start scenarios on five data sets), against the corresponding best competitors. Overall, these results show the effectiveness of the proposed method for the cold-start problem.

A.2. Effectiveness Comparisons: RAPARE-KNN. Next, we evaluate RAPARE-KNN over cold-start user/item scenarios. Since user-based RAPARE-KNN and user-based KNN are much worse than item-based methods for cold-start user scenario, we only evaluate item-based RAPARE-KNN and item-based KNN at that time. In contrast, user-based methods are much better than user-based methods for cold-start item scenario. Fig. 4 presents the results over cold-start user and cold-start item scenarios on the two data sets. As we can see, RAPARE-KNN performs better than KNN for both scenarios on *MovieLens* (Fig. 4(a) and Fig. 4(b)). For *Yelp*, RAPARE-KNN performs better than the compared method in all cases for cold-start user scenario, but performs worse in cold-start item scenario.

B. Component Analysis. Next, we analyze the impact of each component of our RAPARE strategy. Since RAPARE-MF is better than RAPARE-KNN on evaluation, we use RAPARE-MF as the subject in this experiment. We have two components in RAPARE-MF: difference functions (see Section 3.1) and initialization methods (see Section 4.1). We evaluate all the component choices, and report their effects on the RMSE metric. For brevity, we only report the results on the *MovieLens* data. Similar results are observed on other data sets.

We first fix the initialization method as the *average initialization*, and evaluate different choices of difference functions including the *linear* difference, the *logistic* difference, and the *elo* difference. The results for cold-start user and cold-start item scenarios are shown in Fig. 6(a) and Fig. 6(b), respectively. As we can see, the *logistic* difference performs much better than the other two difference functions in the cold-start user scenario. In the cold-start item scenario, the *logistic* difference and the *elo* difference have similar performance. Both *logistic* difference and *elo* difference are much better than *linear* difference. In practice, we recommend the *logistic* difference.

Next, we fix the difference function as *logistic* difference and evaluate different choices for initialization methods, i.e., the *random initialization*, the *average initialization*, and the *clustering-based initialization*. The results for cold-start user and cold-start item scenarios are shown in Fig. 5(a) and Fig. 5(b), respectively. As we can see, *average initialization* outperforms the other two initialization methods, especially for the cold-start user scenario. For the cold-start item scenario, the *clustering-based initialization* performs close to the *average initialization* when the cold-start item received more than 5 ratings. Overall, we recommend the *average initialization* in practice.

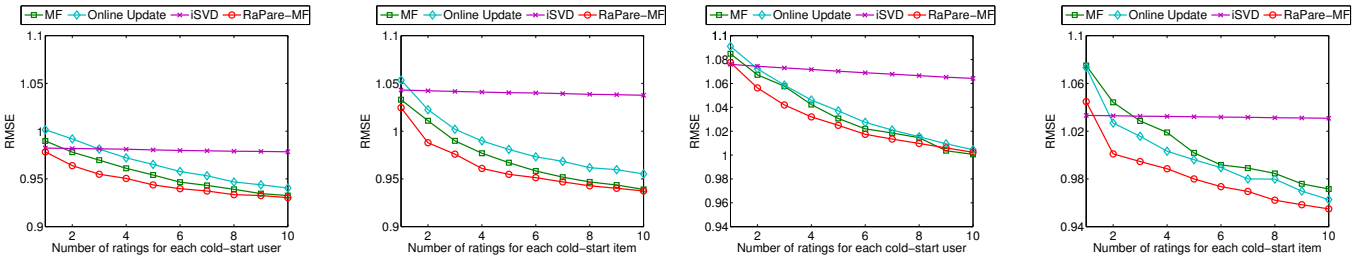
5.3 Efficiency Results

Next, we present the efficiency results of RAPARE-MF and the compared methods. We still report the results on the *MovieLens* data, while similar results are observed on the other data sets. All the experiments are run on a Macbook Pro. The machine has four 2.2GHz Intel i7 Cores and 8GB memory.

For efficiency, we first evaluate the quality-speed balance of different methods over both cold-start user and cold-start item scenarios. The results are shown in Fig. 7. In the figures, we plot the RMSE on the y-axis and the wall-clock time (in log scale) on the x-axis. An ideal method would locate at the left-bottom corner. As we can see from Fig. 7(a) and Fig. 7(b), our RAPARE-MF shows good balance between quality and speed in both cold-start user and cold-start item scenarios. For RAPARE-KNN, the performance is also better than KNN. However, RAPARE-KNN loses the efficiency due to the time cost of similarity calibration. In terms of the speed comparisons, our RAPARE-MF is 40% faster on average than the best competitor (i.e., Online Update).

Finally, we study the scalability of the proposed RAPARE-MF in Fig. 8. In the figures, we show the scalability of our method in terms of both the number of ratings from existing users/items (Fig. 8(a) and Fig. 8(b)) and the number of ratings from cold-start users/items (Fig. 8(c) and Fig. 8(d)). As we can see from the figures, our proposed RAPARE-MF scales linearly in all four cases.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2615039, IEEE Transactions on Knowledge and Data Engineering



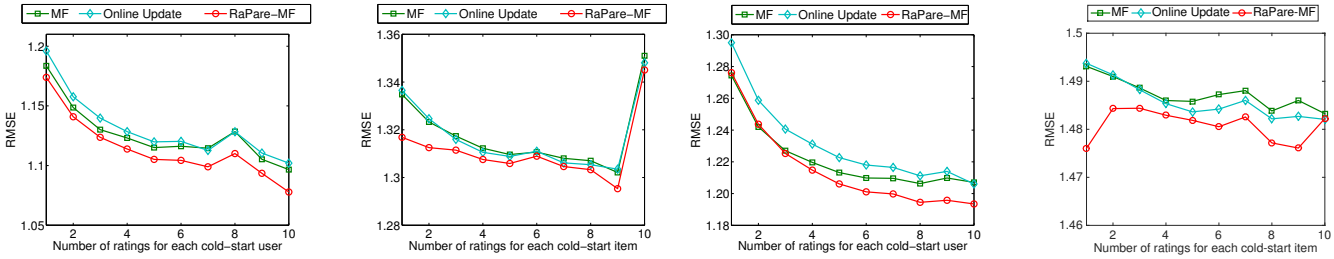
(a) Cold-start user scenario on *MovieLens* (p-value<0.05, except x = 7, 8)

(b) Cold-start item scenario on *MovieLens* (p-value<0.05)

(c) Cold-start user scenario on *EachMovie* (p-value<0.05, except x = 7, 8)

(d) Cold-start item scenario on *EachMovie* (p-value<0.05)

Figure 1. The effectiveness comparisons for cold-start scenarios on the *EachMovie* data and the *MovieLens* data. In general, RAPARE-MF outperforms all the compared methods over both cold-start user/item scenarios on both data sets.



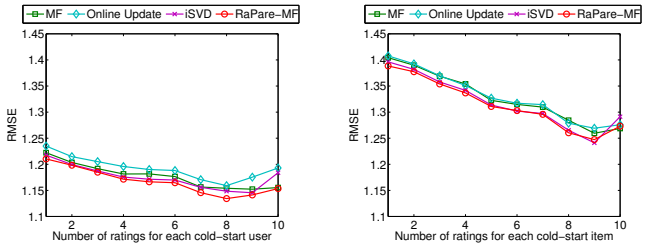
(a) Cold-start user scenario on *Amazon Automotive*(p-value<0.05)

(b) Cold-start item scenario on *Amazon Automotive*(p-value<0.05)

(c) Cold-start user scenario on *Amazon Electronic* (p-value<0.05, except x = 1, 2)

(d) Cold-start item scenario on *Amazon Electronic* (p-value<0.05)

Figure 2. The effectiveness comparisons for cold-start scenarios on the *Amazon Automotive* data and the *Amazon Electronic* data. In general, RAPARE-MF outperforms all the compared methods over both cold-start user/item scenarios on both data sets.



(a) Cold-start user scenario on *Yelp* (p-value<0.05, except x = 2)

(b) Cold-start item scenario on *Yelp* (p-value<0.05, except x = 9)

Figure 3. The effectiveness comparisons for cold-start scenarios on the *Yelp* data. In general, RAPARE-MF outperforms the compared methods over both cold-start user/item scenarios on this data set.

6 RELATED WORK

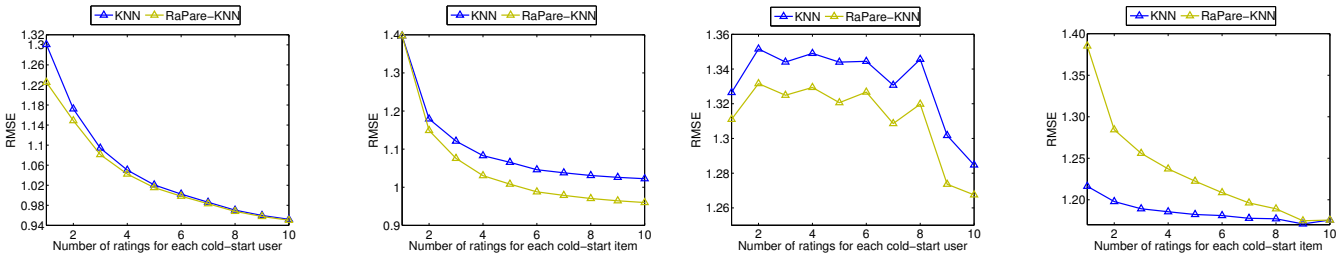
In this section, we review the related work including the classic collaborative filtering methods and the existing recommendation approaches for the cold-start problem.

Classic Collaborative Filtering. Since it was first introduced in the Tapestry recommender system [24], collaborative filtering (CF) has become an indispensable building block in many recommender systems. The intuition of CF is that users who have similar tastes may show similar preferences to the same item, and the only required input for CF is users' historical feedback.

Two prevalent CF methods are the neighborhood method and the matrix factorization method. The basic idea of neighborhood method is to calculate the similarities between users/items, and make recommendations based on the most similar users/items. As for the matrix factorization method, it aims to learn the latent factors with the assumption that the ratings are based on the interactions between user latent factors and item latent factors [17], [25]. Mnih et al. [26] gave their explanation on matrix factorization in probabilistic way. Further, they improve their probabilisitic matrix factorization by bayesian treatment to control parameters and hyperparameters of the model automatically [27]. Instead of RMSE as evaluation metric for collaborative filtering, Balakrishnan et al. [28] proposed a new ranking metric for latent factor models. In 2010, Agarwal et al. [29] first assembled latent Dirichlet allocation (LDA) [30] and matrix factorization as fLDA. Wang et al. [31] also used matrix factorization with LDA as the collaborative topic regression (CTR) model to help researchers find relevant scientific articles. Chen et al. [32] used content from social networks as the input of CTR to improve the quality of recommendation for users.

Recommendations over Cold-Start Scenario. Existing efforts for the cold-start problem can be divided into three classes. In the first class, an additional step of rating collection is required. For example, Park et al. [33] use the *naïve filterbot* algorithm [34] to inject pseudo users to produce additional ratings. More commonly, an interview process is involved: a set of items are presented for the cold-start users to extract their ratings [35], [36], [37], [38]. The main focus of these interview processes lies in the selection of a proper item set.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2615039, IEEE Transactions on Knowledge and Data Engineering



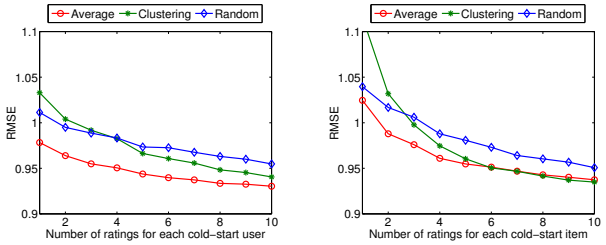
(a) Cold-start user scenario on *MovieLens* (p-value<0.05, except x = 9, 10)

(b) Cold-start item scenario on *MovieLens* (p-value<0.05)

(c) Cold-start user scenario on *Yelp* (p-value<0.05)

(d) Cold-start item scenario on *Yelp*

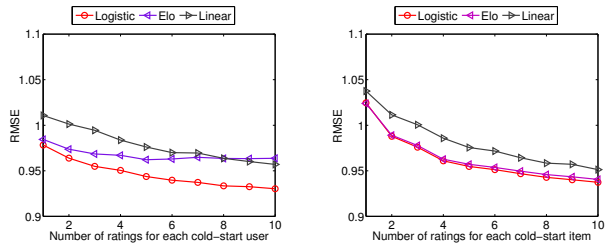
Figure 4. The effectiveness comparisons between RAPARE-KNN and KNN over cold-start scenarios on the *MovieLens* data and the *Yelp* data. In general, RAPARE-KNN outperforms KNN over cold-start user/item scenarios on these data sets.



(a) Cold-start user scenario

(b) Cold-start item scenario

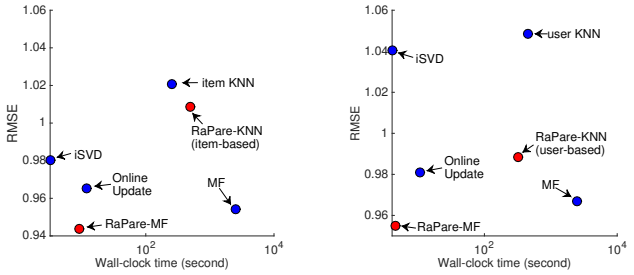
Figure 5. The effects of different initialization methods on the *MovieLens* data. Overall, the *average* method performs best.



(a) Cold-start user scenario

(b) Cold-start item scenario

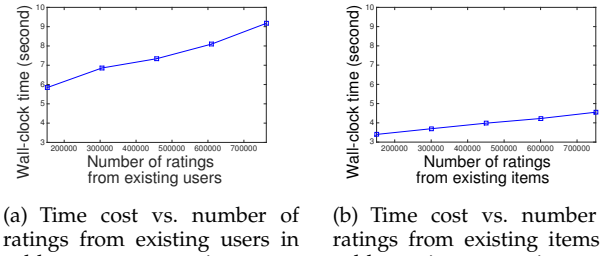
Figure 6. The effects of difference functions on the *MovieLens* data. Overall, the *logistic* difference performs best.



(a) Cold-start user scenario

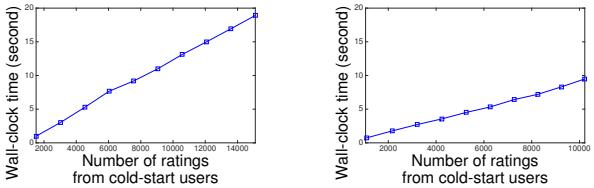
(b) Cold-start item scenario

Figure 7. The quality-speed balance of different methods on the *MovieLens* data. Wall-clock times are plotted in log scale. The proposed RAPARE-MF achieves a good balance between the prediction quality and the efficiency (in the left-bottom corner). RAPARE-KNN also gains better performance compared with KNN.



(a) Time cost vs. number of ratings from existing users in cold-start user scenario

(b) Time cost vs. number of ratings from existing items in cold-start item scenario



(c) Time cost vs. number of ratings from cold-start users in cold-start user scenario

(d) Time cost vs. number of ratings from cold-start items in cold-start item scenario

Figure 8. Scalability of RAPARE-MF. RAPARE-MF scales linearly wrt the data size.

For example, Zhou et al. [35] and Golbandi et al. [36] adopt decision tree to select the item set; active learning for CF is also used for selecting proper interview items [37]. Rashid et al. [10] also discuss the item selection strategy, and they point out that such interview process should be deliberately controlled to avoid user loss.

Methods in the second class resort to *side information* such as the user/item attributes [11] and social relationships [12], [39] to tackle the cold-start problem. Gu et al. [40] provided a graph regularized weighted nonnegative MF by considering several types of side information together. The advantage is that these methods could be applicable for a new user/item with not rating at all. However, they rely on the access of such side information. These methods are inapplicable when the information is not available due to some reasons (e.g., privacy issue, user's social network structure not existing [12]), and has a higher computational cost compared with its side information free counterpart.

In the third class, no additional rating collection or side information is required. Instead, the cold-start problem is considered and tackled in a dynamic manner. Methods in this class emphasize on the importance of new ratings from

cold-start users, and aim to adjust the recommendation for these users as their new ratings arrive. For example, Sarwar et al. [13] introduce an incremental singular value decomposition (iSVD) algorithm for the cold-start users; Tackcs et al. [14] and Rendle et al. [15] also provide incremental algorithms to update the latent factor vectors for cold-start users when they give new ratings. However, methods in the third class cannot serve users with no rating in the recommender system. Our method falls into this class. Different from the existing work whose focus is on providing fast recommendations, we propose a rating comparison model to give special treatments to the cold-start users/items.

7 Conclusion

In this paper, we have proposed a generic rating comparison strategy (RaPare) to make proper recommendations for cold-start problem. In particular, the RaPare strategy provides a special, fine-grained treatment for cold-start users and cold-start items. This generic strategy can be instantiated to many existing methods for recommender systems. We proposed the RaPare-MF (instantiating with matrix factorization method) and RaPare-KNN (instantiating with nearest neighborhood method) models as well as algorithms to solve them. Experimental evaluations on five real data sets show that our approach outperforms several benchmark collaborative filtering and online updating methods in terms of prediction accuracy, and RaPare-MF can provide fast recommendations with linear scalability .

Appendix A
Probabilistic Interpretation of RaPare

Here, we present the probabilistic interpretation for our proposed RaPare strategy. Suppose we have M items, N users, and the scale of ratings from 1 to r_{max}. Let r_{ui} represent the rating that user u gives to item i. Let D_{uvi} represent the *difference* between r_{ui} and r_{vi}, which is calculated by the function $g(r_{ui}, r_{vi})$. $\mathcal{P}_c$ is the set of latent profiles of cold-start users, and p_u represents the specific latent profile for cold-start user u. In order to deal with the cold-start user problem, we adopt a probabilistic model with Gaussian distribution noise. We define the conditional distribution over the observed *differences* as follow

$$p(D|R, \mathcal{P}_c, \sigma^2) = \prod_{u=1}^{N_c} \prod_{v=1}^{N_w} \prod_{i=1}^{M} [\mathcal{N}(D_{uvi}|g(r(p_u, i), r_{vi}), \sigma^2)]^{I_{uvi}} \tag{30}$$

where $\mathcal{N}(x|\mu, \sigma^2)$ is the probability density function of Gaussian distribution with mean μ and variance σ^2, function $g(r_1, r_2)$ represents the *difference* function to calculate the *difference* between r_1 and r_2, function $r(p_u, i)$ is the function to predict rating to item i for the user u with her latent profile p_u, N_c is the number of cold-start users, N_w is the number of existing users, and I_{uvi} is the indicator that equals to 1 if user u and user v are both rated item i and equals to 0 otherwise. We choose zero-mean Gaussian priors on latent profiles of cold-start users

$$p(\mathcal{P}_c|0, \sigma^2_{\mathcal{P}_c}) = \prod_{u=1}^{N_c} \mathcal{N}(p_u|0, \sigma^2_{\mathcal{P}_c}) \tag{31}$$

Then, the log of posterior distribution over latent profiles of cold-start users is as follow

$$\begin{aligned} &\ln p(\mathcal{P}_c|D, R, \sigma^2, \sigma^2_{\mathcal{P}_c}) = \\ &-\frac{1}{2\sigma^2} \sum_{u=1}^{N_c} \sum_{v=1}^{N_w} \sum_{i=1}^{M} I_{uvi} (g(r_{ui}, r_{vi}) - g(\hat{r}_{ui}, r_{vi}))^2 \\ &-\frac{1}{2\sigma^2_{\mathcal{P}_c}} \sum_{u=1}^{N_c} p_u^2 - \sum_{u=1}^{N_c} \sum_{v=1}^{N_w} \sum_{i=1}^{M} I_{uvi} \ln \sigma^2 - N_c \ln \sigma^2_{\mathcal{P}_c} + C \end{aligned} \tag{32}$$

where $g(r_{ui}, r_{vi})$ is used for calculating D_{uvi}, $\hat{r}_{ui}$ is the result of $r(p_u, i)$, and C is a constant. Maximizing this log posterior over latent profiles of cold-start users with hyperparameters (e.g., σ^2 and $\sigma^2_{\mathcal{P}_c}$) kept fixed is equivalent to minimizing the optimization task in Eq. (12)

$$E = \sum_{r_{ui} \in \mathcal{R}_c} \sum_{r_{vi} \in \mathcal{R}^i_w} (g(r_{ui}, r_{vi}) - g(\hat{r}_{ui}, r_{vi}))^2 + \lambda \sum_{p_u \in \mathcal{P}_c} p_u^2 \tag{33}$$

where $\lambda = \sigma^2/\sigma^2_{\mathcal{P}_c}$, and $\sum_{r_{ui} \in \mathcal{R}_c} \sum_{r_{vi} \in \mathcal{R}^i_w}$ is equivalent to $\sum_{u=1}^{N_c} \sum_{v=1}^{N_w} \sum_{i=1}^{M} I_{uvi}$.

Appendix B
Discussions: RaPare-Universal

Here, we apply RaPare to the classic rating prediction scenario. To deal with classic rating prediction problem, matrix factorization method targets to minimize the difference between predicted rating and actual rating. In order to enhance the power of MF, we propose RaPare-Universal which adds RaPare to Eq. 2 as a term of regularization

$$\arg\min_{\mathbf{P}, \mathbf{Q}} \text{RaPare-Uni-Opt}(\mathcal{T}, \mathcal{R}, \mathbf{P}, \mathbf{Q}) \tag{34}$$

with

$$\begin{aligned} \text{RaPare-Uni-Opt} = &\sum_{r_{ui} \in \mathcal{T}} (r_{ui} - \hat{r}_{ui})^2 + \lambda(||\mathbf{P}||^2_F + ||\mathbf{Q}||^2_F) \\ &+ \lambda_1 \sum_{r_{ui} \in \mathcal{T}} \sum_{r_{mn} \in \mathcal{R}_u \cup \mathcal{R}_i} (\underbrace{g(r_{ui}, r_{mn})}_{actual\ diff} - \underbrace{g(\hat{r}_{ui}, r_{mn})}_{expected\ diff})^2 \end{aligned}$$

where $\mathcal{T}$ is the training set, $\mathcal{R}_u$ is the set of ratings from user u, $\mathcal{R}_i$ is the set of ratings from item i, λ_1 is the weight of RaPare regularization term, and $\mathbf{P}$ and $\mathbf{Q}$ are the matrices of latent profiles for users and items, respectively.

Similar to RaPare-MF, we can transfer Eq. (34) into the fast model. The optimization equation in Eq. (34) could be re-written as

$$\begin{aligned} &\text{RaPare-Uni-Opt*} = \\ &\sum_{r_{ui} \in \mathcal{T}} (r_{ui} - \hat{r}_{ui})^2 + \lambda(||\mathbf{P}||^2_F + ||\mathbf{Q}||^2_F) \\ &+ \lambda_1 \sum_{r_{ui} \in \mathcal{T}} \sum_{r=1}^{r_{max}} |\Omega_{\mathcal{R}_u \cup \mathcal{R}_i, r}| (g(r_{ui}, r) - g(\hat{r}_{ui}, r))^2 \end{aligned}$$

where r_{max} is the maximal rating scale, and $|\Omega_{\mathcal{R}_u \cup \mathcal{R}_i, r}|$ indicates the number of ratings in $\mathcal{R}_u \cup \mathcal{R}_i$ with a value r.

Inference Algorithm for RaPare-Universal. We also use stochastic gradient descent with mini-batch on RaPare regularization term to learn the parameters of RaPare-Uni-Opt*. The basic update process is as follow

$$\theta \leftarrow \theta - \alpha \nabla^*_\theta \tag{35}$$

Algorithm 5: Fast learning RAPARE-UNIVERSAL

Input: ratings in training set $\mathcal{T}$, and the maximal rating scale r_{max}
Output: user latent profile matrix $\mathbf{P}$ and item latent profile matrix $\mathbf{Q}$

initialize $\mathbf{P}$ and $\mathbf{Q}$
repeat
  for $r_{ui} \in \mathcal{T}$ **do**
    for $f \leftarrow 1, ..., k$ **do**
      $\nabla^*_{p_{u,f}} = -2 \cdot (r_{ui} - \hat{r}_{ui}) * q_{i,f} + 2\lambda \cdot p_{u,f}$
      $\nabla^*_{q_{i,f}} = -2 \cdot (r_{ui} - \hat{r}_{ui}) * p_{u,f} + 2\lambda \cdot q_{i,f}$
      for $r \leftarrow 1, ..., r_{max}$ **do**
        $\nabla^*_{p_{u,f}} += -2\lambda_1 \cdot \sum_{r=1}^{r_{max}} |\Omega_{\mathcal{R}_u \cup \mathcal{R}_i, r}| \cdot err_{RaPare} \cdot \frac{e^{-(\hat{r}_{ui}-r)}}{(1+e^{-(\hat{r}_{ui}-r)})^2} \cdot q_{i,f}$
        $\nabla^*_{q_{i,f}} += -2\lambda_1 \cdot \sum_{r=1}^{r_{max}} |\Omega_{\mathcal{R}_u \cup \mathcal{R}_i, r}| \cdot err_{RaPare} \cdot \frac{e^{-(\hat{r}_{ui}-r)}}{(1+e^{-(\hat{r}_{ui}-r)})^2} \cdot p_{u,f}$
      update $p_{u,f} \leftarrow p_{u,f} - \alpha \nabla^*_{p_{u,f}}$
      update $q_{i,f} \leftarrow q_{i,f} - \alpha \nabla^*_{q_{i,f}}$
until *convergence*;
return $\mathbf{P}$ and $\mathbf{Q}$

where θ represents the parameter that needs to be learned, α is the learning rate, and ∇^*_θ is the basic update unit. If we use Eq. (1) as the rating prediction function, the derivative of parameters (i.e., p_u and q_i) in Eq. (35) are listed below

$$\nabla^*_{p_{u,f}} = \frac{\partial \text{RAPARE-UNI-OPT*}}{\partial p_{u,f}} = -2 \cdot (r_{ui} - \hat{r}_{ui}) \cdot q_{i,f} + 2\lambda \cdot p_{u,f} - 2\lambda_1 \cdot \sum_{r=1}^{r_{max}} |\Omega_{\mathcal{R}_u \cup \mathcal{R}_i, r}| \cdot err_{RaPare} \cdot \frac{e^{-(\hat{r}_{ui}-r)}}{(1+e^{-(\hat{r}_{ui}-r)})^2} \cdot q_{i,f}$$

and

$$\nabla^*_{q_{i,f}} = \frac{\partial \text{RAPARE-UNI-OPT*}}{\partial q_{i,f}} = -2 \cdot (r_{ui} - \hat{r}_{ui}) \cdot p_{u,f} + 2\lambda \cdot q_{i,f} - 2\lambda_1 \cdot \sum_{r=1}^{r_{max}} |\Omega_{\mathcal{R}_u \cup \mathcal{R}_i, r}| \cdot err_{RaPare} \cdot \frac{e^{-(\hat{r}_{ui}-r)}}{(1+e^{-(\hat{r}_{ui}-r)})^2} \cdot p_{u,f}$$

where

$$err_{RaPare} = (g(r_{ui}, r) - g(\hat{r}_{ui}, r)) \quad (36)$$

Alg. 5 shows the parameter updating process. The time complexity of each iteration is $O(|\mathcal{T}| \cdot r_{max} \cdot k)$. Since r_{max} and k are constants in the target recommender system, the time complexity could be re-written as $O(|\mathcal{T}|)$, which is same as the time complexity of original matrix factorization method.

Compatibility of RaPare. Here, we discuss the compatibility of RAPARE with existing models that deals with the classic rating prediction problem. In this paper, we assemble RAPARE with matrix factorization method as RAPARE-UNIVERSAL. As shown in Eq. (35), to construct our optimization function, we treat RAPARE as a regularization term that instantiates to an existing method. In other words, Eq. (35) can be generalized as

$$\text{RAPARE-UNI-OPT} = \textit{existing model} + \text{RAPARE} \quad (37)$$

In *existing model* part, we can implement this part with many of existing models (e.g., MF, biased MF, biased MF with implicit feedback, etc) in recommender systems. Since the final optimization target is ensembled with *existing model* and RAPARE linearly, there is no dependency between them. With this property, we can enhance the power of the given *existing model* by ensembling RAPARE. The only restriction is that the chosen *existing* model must have the latent profiles for users and items, and these latent profiles should be treated as parameters to be learned.

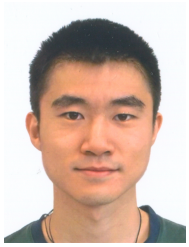
ACKNOWLEDGMENTS

This work was supported in part by National Basic Research 973 Program (Grant No. 2015CB352202), National Natural Science Foundation (Grant Nos. 61373011, 91318301, 61321491) of China, the Collaborative Innovation Center of Novel Software Technology and Industrialization. This work is supported by the program B for Outstanding PhD candidate of Nanjing University. This work is partially supported by the National Science Foundation under Grant No. IIS1017415, by DTRA under the grant number HDTRA1-16-0017, by Army Research Office under the contract number W911NF-16-1-0168, by National Institutes of Health under the grant number R01LM011986, Region II University Transportation Center under the project number 49997-33 25 and a Baidu gift.

REFERENCES

[1] P. Resnick and H. R. Varian, "Recommender systems," *Communications of the ACM*, vol. 40, no. 3, pp. 56–58, 1997.
[2] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston *et al.*, "The youtube video recommendation system," in *RecSys*. ACM, 2010, pp. 293–296.
[3] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in *WWW*. ACM, 2007, pp. 271–280.
[4] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *Internet Computing, IEEE*, vol. 7, no. 1, pp. 76–80, 2003.
[5] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender system-a case study," DTIC Document, Tech. Rep., 2000.
[6] The netflix prize. http://www.netflixprize.com/.
[7] A. Merve Acilar and A. Arslan, "A collaborative filtering method based on artificial immune network," *Expert Systems with Applications*, vol. 36, no. 4, pp. 8324–8332, 2009.
[8] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The adaptive web*. Springer, 2007, pp. 291–324.
[9] A. Töscher, M. Jahrer, and R. M. Bell, "The bigchaos solution to the netflix grand prize," *Netflix prize documentation*, 2009.
[10] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl, "Getting to know you: learning new user preferences in recommender systems," in *IUI*. ACM, 2002, pp. 127–134.
[11] M. Zhang, J. Tang, X. Zhang, and X. Xue, "Addressing cold start in recommender systems: A semi-supervised co-training algorithm," in *SIGIR*, 2014.
[12] J. Lin, K. Sugiyama, M.-Y. Kan, and T.-S. Chua, "Addressing cold-start in app recommendation: latent user models constructed from twitter followers," in *SIGIR*. ACM, 2013, pp. 283–292.
[13] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Incremental singular value decomposition algorithms for highly scalable recommender systems," in *Fifth International Conference on Computer and Information Science*. Citeseer, 2002, pp. 27–28.

[14] G. Takács, I. Pilászy, B. Nemeth, and D. Tikk, "Investigation of various matrix factorization methods for large recommender systems," in *Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on*. IEEE, 2008, pp. 553–562.
[15] S. Rendle and L. Schmidt-Thieme, "Online-updating regularized kernel matrix factorization models for large-scale recommender systems," in *RecSys*. ACM, 2008, pp. 251–258.
[16] A. E. Elo, *The rating of chessplayers, past and present*. Batsford London, 1978, vol. 3.
[17] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.
[18] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *WWW*. ACM, 2001, pp. 285–295.
[19] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: an open architecture for collaborative filtering of netnews," in *CSCW*. ACM, 1994, pp. 175–186.
[20] D. P. Bertsekas and J. N. Tsitsiklis, "Gradient convergence in gradient methods with errors," *SIAM Journal on Optimization*, vol. 10, no. 3, pp. 627–642, 2000.
[21] J. McAuley, R. Pandey, and J. Leskovec, "Inferring networks of substitutable and complementary products," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 785–794.
[22] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, "Image-based recommendations on styles and substitutes," in *SIGIR*. ACM, 2015, pp. 43–52.
[23] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *SIGKDD*. ACM, 2008, pp. 426–434.
[24] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, 1992.
[25] B. Shapira, *Recommender systems handbook*. Springer, 2011.
[26] A. Mnih and R. Salakhutdinov, "Probabilistic matrix factorization," in *NIPS*, 2007, pp. 1257–1264.
[27] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *ICML*. ACM, 2008, pp. 880–887.
[28] S. Balakrishnan and S. Chopra, "Collaborative ranking," in *WSDM*. ACM, 2012, pp. 143–152.
[29] D. Agarwal and B.-C. Chen, "flda: matrix factorization through latent dirichlet allocation," in *WSDM*. ACM, 2010, pp. 91–100.
[30] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *JMLR*, vol. 3, pp. 993–1022, 2003.
[31] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *SIGKDD*. ACM, 2011, pp. 448–456.
[32] C. Chen, X. Zheng, Y. Wang, F. Hong, and Z. Lin, "Context-aware collaborative topic regression with social matrix factorization for recommender systems," in *AAAI*, 2014.
[33] S.-T. Park, D. Pennock, O. Madani, N. Good, and D. DeCoste, "Naïve filterbots for robust cold-start recommendations," in *SIGKDD*. ACM, 2006, pp. 699–705.
[34] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl, "Combining collaborative filtering with personal agents for better recommendations," in *AAAI/IAAI*, 1999, pp. 439–446.
[35] K. Zhou, S.-H. Yang, and H. Zha, "Functional matrix factorizations for cold-start recommendation," in *SIGIR*. ACM, 2011, pp. 315–324.
[36] N. Golbandi, Y. Koren, and R. Lempel, "Adaptive bootstrapping of recommender systems using decision trees," in *WSDM*. ACM, 2011, pp. 595–604.
[37] A. S. Harpale and Y. Yang, "Personalized active learning for collaborative filtering," in *SIGIR*. ACM, 2008, pp. 91–98.
[38] M. Sun, F. Li, J. Lee, K. Zhou, G. Lebanon, and H. Zha, "Learning multiple-question decision trees for cold-start recommendation," in *WSDM*. ACM, 2013, pp. 445–454.
[39] Y. Yao, H. Tong, G. Yan, F. Xu, X. Zhang, B. Szymanski, and J. Lu, "Dual-regularized one-class collaborative filtering," in *CIKM*, 2014.
[40] Q. Gu, J. Zhou, and C. H. Ding, "Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs." in *SDM*. SIAM, 2010, pp. 199–210.

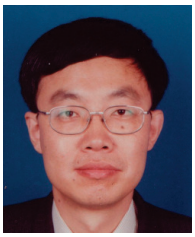


Jingwei Xu is a Ph.D. student in the Department of Computer Science and Technology, Nanjing University, China. He received the B.Sc. degree in software engineering from Nanjing University of Posts & Telecommunications in 2009. After that, he received the M.Sc. degree in computer technology from Nanjing University in 2012. His current research interests include recommender systems and course analysis on MOOCs.

Yuan Yao received Ph.D. degree in the Department of Computer Science and Technology, Nanjing University, China. He received the B.Sc. degree in computer science from Nanjing University in 2009. His current research interests include social network analysis, trust inference, and reputation systems.

Hanghang Tong Hanghang Tong is currently an assistant professor at School of Computing, Informatics, and Decision Systems Engineering (CIDSE), Arizona State University. Before that, he was an assistant professor at Computer Science Department, City College, City University of New York, a research staff member at IBM T.J. Watson Research Center and a Post-doctoral fellow in Carnegie Mellon University. He received his M.Sc and Ph.D. degree from Carnegie Mellon University in 2008 and 2009, both majored in Machine Learning. His research interest is in large scale data mining for graphs and multimedia. He has received several awards, including one 'test of time' award (ICDM 10-Year highest impact paper award), four best paper awards and four 'best of conference'. He has published over 100 referred articles and more than 20 patents.

Xianping Tao received his M.Sc. and Ph.D. degrees in computer science from Nanjing University in 1994 and 2001, respectively. He is currently a professor in the Department of Computer Science at Nanjing University. His research interests include software agents, middleware systems, Internetware methodology, and pervasive computing. He is a member of CCF and IEEE.



Jian Lu Jian Lu received the B.Sc., M.Sc., and Ph.D. degrees in computer science from Nanjing University, China. He is currently a Professor in the Department of Computer Science and Technology, and the Director of the State Key Laboratory for Novel Software Technology at Nanjing University, China. He serves as the vice chairman of the China Computer Federation. His current research interests include software methodology, software automation, software agents, and middleware systems.