Ice-Breaking: Mitigating Cold-Start Recommendation Problem by Rating Comparison

Jingwei Xu, Yuan Yao
State Key Laboratory for Novel Software Technology, China
{jingwei.xu,yyao}@smail.nju.edu.cn

Hanghang Tong
Arizona State University, USA
hanghang.tong@asu.edu

Xianping Tao and Jian Lu
State Key Laboratory for Novel Software Technology, China
{txp,lj}@nju.edu.cn

Abstract

Recommender system has become an indispensable component in many e-commerce sites. One major challenge that largely remains open is the *cold-start* problem, which can be viewed as an ice barrier that keeps the cold-start users/items from the warm ones. In this paper, we propose a novel rating comparison strategy (RAPARE) to break this ice barrier. 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 warm users/items. We instantiate our RAPARE strategy on the prevalent method in recommender system, i.e., the matrix factorization based collaborative filtering. Experimental evaluations on two real data sets validate the superiority of our approach over the existing methods in cold-start scenarios.

1 Introduction

Since the concept of recommender systems emerged in 1990s [Resnick and Varian, 1997], both industry and academia have witnessed the rapid advancement in this field. For example, during the last decade, many mainstream e-commerce companies have reported significant profit growth by integrating recommender systems [Davidson *et al.*, 2010; Das *et al.*, 2007; Linden *et al.*, 2003; Sarwar *et al.*, 2000].

Despite the success of existing recommender systems, the *cold-start* problem [Merve Acilar and Arslan, 2009; Schafer *et al.*, 2007], 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 less than 10 items) and cold-start items (e.g., which have received less than 10 ratings) occupy a large proportion in many real applications such as Netflix [Töscher *et al.*, 2009]. 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 the effectiveness for cold-start users/items with very few ratings could degrade dramatically.

To date, many efforts have been made to mitigate the cold-start problem, and these efforts can be divided into three classes. In the first class, an initial *interview process* is introduced for cold-start users [Rashid *et al.*, 2002]. 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 [Zhang *et al.*, 2014] and social relationships [Lin *et al.*, 2013]. However, these methods become infeasible in many real recommender systems where such side information may not be available. In the third class, the cold-start problem is tackled in a *dynamic way*. The intuition is that, compared to warm users/items, ratings for cold-start users/items may be more valuable to improve their prediction accuracy; consequently, methods in this class aim to provide fast recommendations for cold-start users/items, and then dynamically adjust their latent profiles as they give/receive new ratings. Examples in this class include the incremental singular value decomposition (iSVD) method [Sarwar *et al.*, 2002] and the incremental matrix factorization method [Takács *et al.*, 2008; Rendle and Schmidt-Thieme, 2008].

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

In particular, we make the following analogy, i.e., to view the cold-start problem as an ice barrier between the cold-start users/items and the warm ones, and such an ice barrier could be broken with the help of warm users/items. To this end, we propose an ice-breaker via a novel rating comparison strategy (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 warm users) on this item, and then 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 [Elo, 1978] 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 rating comparison strategy RAPARE

to serve as an ice-breaker for the cold-start problem. The key idea of RAPARE is to exploit the knowledge from warm users/items to help calibrate the latent profiles of cold-start users/items.

- We instantiate the proposed RAPARE strategy on matrix factorization based collaborative filtering (RAPARE-MF), together with an effective and efficient solver.
- We conduct extensive experimental evaluations on two real data sets, showing that our approach (1) outperforms several benchmark collaborative filtering methods and online updating methods in terms of prediction accuracy; (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 and the proposed model. In Section 4, we present the experimental results. In Section 5, we review related work. Finally, we conclude the paper in Section 6.

2 Problem Statement

In this section, we present the problem statement of cold-start recommendation. Suppose we have sets of users $\mathcal{U}$, items $\mathcal{I}$ and observed 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 less than a certain amount of ratings (e.g., 10 ratings) as cold-start users and the rest as warm users. Meanwhile, $\mathcal{R}_c$ and $\mathcal{R}_w$ denote the sets of ratings that belong to cold-start users and warm users, respectively. We use $\mathcal{R}_w(i)$ to represent the set of ratings on item i from warm users. 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 warm 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 warm 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.

Preliminary #1. To date, matrix factorization has been one of the most dominate methods in recommender systems. Matrix factorization (MF) [Koren *et al.*, 2009] 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 common latent factor space to predict the rating (e.g., $\hat{r}_{ui}$) by the following equation

$$\hat{r}_{ui} = p_u^T \cdot q_i \tag{1}$$

where vector p_u and q_i are latent profiles for user u and item i, respectively. To learn these latent profiles, the square loss is usually used as the loss function

$$\min_{q*,p*} \sum_{r_{ui} \in \mathcal{R}} (r_{ui} - p_u^T \cdot q_i)^2 + \lambda(||q_i||^2 + ||p_u||^2) \tag{2}$$

where *L2* regularization is to avoid over-fitting.

In Eq. (2), the learned latent profiles may be less accurate for cold-start users/items due to the lack of sufficient ratings. 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 warm users/items. We will describe how to calibrate the latent profiles for cold-start users/items in the following section.

Preliminary #2. Elo Rating System, which is first adopted in chess, can be used to measure the relative skill levels between players in a certain competition. The basic idea behind Elo 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.

3 The Proposed Approach

In this section, we present our rating comparison strategy (i.e., RAPARE) and the instantiation of RAPARE on matrix factorization based collaborative filtering (i.e., RAPARE-MF). For brevity, we use the cold-start users as the subject throughout this section. It is straightforward to apply the proposed method for the cold-start items.

3.1 The RaPare Strategy

We aim to break the ice barrier between cold-start users and warm users with the help of warm users, so that we could better build the latent profiles of cold-start users. Specially, 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 *actual* result from the rating comparison strategy to update the latent profiles of cold-start users. For example, suppose u is a cold-start user who has just rated item i, and v is an existing warm user who rated item i in the past. 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}. 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.

The basic idea of our RAPARE strategy is to tune the ratings (i.e., the latent profiles) according to the difference be-

tween the expected result and actual result of a rating comparison. From the perspective of optimization, the RAPARE strategy is equivalent to minimize the difference between the expected result and actual result by learning/tuning the the latent profiles of cold-start users. Therefore, we can minimize the following equation

$$\sum_{r_{ui}\in\mathcal{R}_c}\sum_{r_{vi}\in\mathcal{R}_w^i}(\underbrace{g(r_{ui},r_{vi})}_{actual\ diff}-\underbrace{g(\hat{r}_{ui},r_{vi})}_{expected\ diff})^2 \tag{3}$$

where g is the loss/difference function for a rating comparison, and the square loss is used as the loss function, calculating the difference between expected result and actual result of a rating comparison. 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 warm 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, the are defined as follows

$$g_{linear}(r_{ui},r_{vi}) = r_{ui} - r_{vi} \tag{4}$$

$$g_{logistic}(r_{ui},r_{vi}) = \frac{1}{1+e^{-(r_{ui}-r_{vi})}} \tag{5}$$

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 from Elo Rating System

$$g_{elo_{actual}}(r_{ui},r_{vi})\begin{cases}1, & \text{if } r_{ui}>r_{vi}\\ 0.5, & \text{if } r_{ui}=r_{vi},\ and\\ 0, & \text{otherwise}\end{cases} \tag{6}$$

$$g_{elo_{expected}}(\hat{r}_{ui},r_{vi}) = \frac{1}{1+e^{-(r_{ui}-r_{vi})}} \tag{7}$$

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 The RAPARE-MF Model

Next, we instantiate RAPARE with the matrix factorization method. Formally, we have the following optimization problem for RAPARE-MF

$$\arg\min_{\mathbf{P}_c} Opt(\mathcal{R}_w,\mathcal{R}_c,\mathbf{P}_c,\mathbf{Q}) \tag{8}$$

with

$$Opt(\mathcal{R}_w,\mathcal{R}_c,\mathbf{P}_c,\mathbf{Q}) = \sum_{r_{ui}\in\mathcal{R}_c}\sum_{r_{vi}\in\mathcal{R}_w^i}(\underbrace{g(r_{ui},r_{vi})}_{actual\ diff}-\underbrace{g(\hat{r}_{ui},r_{vi})}_{expected\ diff})^2+\lambda||\mathbf{P}_c||_F^2 \tag{9}$$

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

As we can see from Eq. (8), given the input of the existing ratings from warm users, the arrival ratings from cold-start users, the initial latent profiles of cold-start users, and the latent profiles of items (learnt by MF method beforehand), the RAPARE-MF model aims to minimize the loss function defined in Eq. (9) by adjusting the $\mathbf{P}_c$. The reason we focus on calibrating $\mathbf{P}_c$ while fixing $\mathbf{Q}$ 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 the $\mathbf{P}_c$ is solved, we can estimate the rating from a cold-start user by Eq. (1).

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

$$\begin{aligned}\mathcal{L} &= \sum_{r_{ui}\in\mathcal{R}_c}\sum_{r_{vi}\in\mathcal{R}_w^i}((r_{ui}-r_{vi})-(\hat{r}_{ui}-r_{vi}))^2+\lambda||\mathbf{P}_c||_F^2\\ &= \sum_{r_{ui}\in\mathcal{R}_c}|\mathcal{R}_w^i|(r_{ui}-\hat{r}_{ui})^2+\lambda||\mathbf{P}_c||_F^2\end{aligned} \tag{10}$$

As we can see, similar to the traditional MF method [Rendle and Schmidt-Thieme, 2008], 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.

3.3 Fast Model Inference Algorithm

Here, we propose a fast learning algorithm to solve the optimization problem in Eq. (8), 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. (8), the contribution of the ratings from different warm users to item i is equal to each other if they share the same rating on item i. Specially, the optimization equation in Eq. (8) could be re-written as

$$\mathcal{L} = \sum_{r_{ui}\in R_c}\sum_{r=1}^{r_{max}}|\Omega_{\mathcal{R}_w^i,r}|(g(r_{ui},r)-g(\hat{r}_{ui},r))^2+\lambda||\mathbf{P}_c||_F^2 \tag{11}$$

where 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 a value r.

For a given rating r_{ui} (from the cold-start user u to item i), we can now aggregate over the existing ratings (from the warm users to item i) with the same value, to form the basic update unit. Specially, we have the following general updating rule for Eq. (11)

$$p_{u,f} \leftarrow p_{u,f} - \alpha\nabla^*_{p_{u,f}} \tag{12}$$

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

$$\begin{aligned}\nabla^*_{p_{u,f}} = \frac{\partial}{\partial p_{u,f}}\mathcal{L} = 2\cdot|\Omega_{\mathcal{R}_w^i,r}|\cdot(g(r_{ui},r)-\\ g(\hat{r}_{ui},r))\cdot\frac{\partial}{\partial p_{u,f}}g(\hat{r}_{ui},r)+2\lambda p_{u,f}\end{aligned} \tag{13}$$

Algorithm 1: Fast learning RAPARE-MF

Input: ratings from warm users $\mathcal{R}_w$, ratings from cold-start users $\mathcal{R}_c$, item latent factors $\mathbf{Q}$, and the maximal rating scale r_{max}

Output: cold-start user latent factors $\mathbf{P_c}$

1 initialize $\mathbf{P_c}$
2 **repeat**
3 **for** $r_{ui} \in \mathcal{R}_c$ **do**
4 **for** $r \leftarrow 1, ..., r_{max}$ **do**
5 **for** $f \leftarrow 1, ..., k$ **do**
6 update $p_{u,f} \leftarrow p_{u,f} - \alpha \nabla^*_{p_{u,f}}$ as defined in Eq. (12)
7 **until** *convergence*;
8 **return** $\mathbf{P}_c$

where

$$\frac{\partial}{\partial p_{u,f}} g_{linear} = q_{i,f} \tag{14}$$

for the *linear* difference function, and

$$\frac{\partial}{\partial p_{u,f}} g_{logistic} = \frac{e^{-(\hat{r}_{ui} - r_{vi})}}{(1 + e^{-(\hat{r}_{ui} - r_{vi})})^2} \cdot q_{i,f} \tag{15}$$

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 follows.

$$\nabla^*_{p_{u,f}} = |\Omega_{\mathcal{R}^i_w, r}|(g_{elo_{actual}} - g_{elo_{expected}}) \cdot q_{i,f} \tag{16}$$

We use mini-batch gradient descent method to learn the parameters, and the algorithm is summarized in Algorithm 1. As we can see from the algorithm, we first initialize the $\mathbf{P}_c$ matrix. There are several initialization methods such as random initialization or cluster-based initialization. In this work, we adopt a heuristic averaging method. That is, for each cold-start user u, we first find the set of warm users who have given the same rating to item i as u does, and computes the average $\overline{p}_i$ over latent profiles from these warm users. Then, we can have the initial p_u for user u by further averaging $\overline{p}_i$ over all rated items by user u. After initialization, the algorithm starts the iteration.

The time complexity of each iteration (Steps 3-6) in Algorithm 1 is $O(|\mathcal{R}_c| \cdot r_{max} \cdot k)$. Notice that if we use a straightforward gradient descent method, its average-case time complexity is $O(|\mathcal{R}_c| \cdot |\overline{\mathcal{R}^i_w}| \cdot k))$. Thus, Algorithm 1 is much faster, since $r_{max} \ll |\overline{\mathcal{R}^i_w}|$ where $|\overline{\mathcal{R}^i_w}|$ is the average number of ratings from warm users to each item.

4 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 problem?
- *Efficiency*: How fast is the proposed approach for updating the latent profiles of cold-start users/items?

Table 1: The statistics of the two 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]

4.1 Experimental Setup

Data Sets and Evaluation Metrics

We use two real, benchmark data sets: *MovieLens*[1] and *EachMovie*. The *MovieLens* data contains 1M ratings that are collected and published from the MovieLens website[2]. Each user in this data set rated at least 20 movies. *EachMovie* is organized by HP/Compaq Research. The statistics of the two data sets are summarized in Table 1.

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

$$RMSE = \sqrt{\frac{\sum_{r_{ui} \in \mathcal{E}} (\hat{r}_{ui} - r_{ui})^2}{|\mathcal{E}|}} \tag{17}$$

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.

Evaluation Protocol

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 [Rendle and Schmidt-Thieme, 2008].

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

1. Setup the cold-start user scenario
   - Randomly choose 25% users (as 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 warm users.
   - Train the model for warm users via the MF method [Koren *et al.*, 2009]. After this step, we can obtain the latent profiles of warm users and the latent profiles for all items
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[3] one of his ratings from the $\mathcal{E}_c$ and move it to $\mathcal{R}_c$

[1] http://www.grouplens.org/datasets/movielens/

[2] http://movielens.org

[3] The timestamps provided by these two datasets are the time that users rated the movies on their websites for data collection. Thus, all these timestamps cannot reflect the time sequences that user watch these movies. In other words, such timestamps do not capture the cold-start scenario.

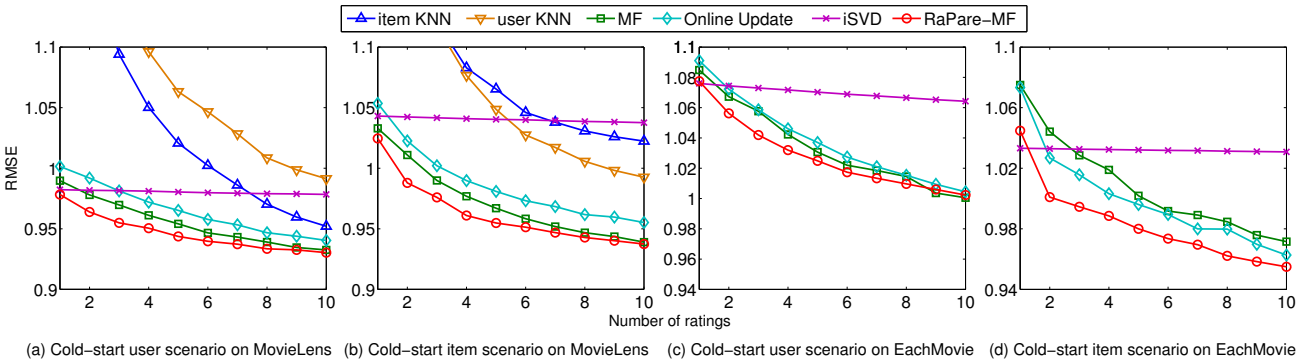


Figure 1: The effectiveness comparisons for cold-start scenarios on the *EachMovie* data and the *MovieLens* data. In general, RAPARE-MF outperforms all the compared methods for both cold-start users and cold-start items on both data sets.

- ∗ learn the latent profile of user u by the proposed method RAPARE-MF
- ∗ 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

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. For example, MF method may take the ratings from warm users and cold-start users together as input. Here, we compare our RAPARE-MF model with several classic collaborative filtering methods for both effectiveness and efficiency. The compared methods include: *user/item based neighborhood methods* (i.e., user KNN and item KNN) [Sarwar *et al.*, 2001], *incremental SVD* (iSVD) [Sarwar *et al.*, 2002], *matrix factorization* (MF) [Koren *et al.*, 2009], and *Online Update* which is proposed for updating the latent profiles of cold-start users/items [Rendle and Schmidt-Thieme, 2008].

4.2 Effectiveness Results

We first evaluate the effectiveness of the proposed method with the compared methods under the cold-start scenario. The results for both cold-start users and cold-start items on both data sets are shown in Fig. 1. The x-axis of Fig. 1 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. Smaller RMSE is better. Due to the limited page length, we do not provide the comparison of effectiveness on the three proposed difference functions in section 3.1. In this paper, we only choose *logistic* to evaluate the effectiveness, since it performs better than other two difference functions. On the *EachMovie* data, the performance of user KNN and item KNN are significantly worse than the other methods. Therefore, we exclude these two KNN methods in the comparison on the *EachMovie* data.

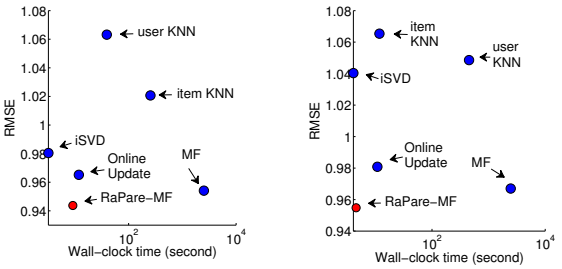
There are several observations from the results on the *MovieLens* data (Fig. 1(a) and Fig. 1(b)). First of all, in general, RAPARE-MF performs better than all the compared methods for both cold-start users and cold-start items. The item KNN and user KNN perform relatively poorly, especially at the beginning of the evaluation. The reason is that the preferences of the cold-start users/items are not accurately captured. 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. However, as the evaluation continues, no significant improvement is observed in 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 to lower the prediction error. Notice that, small improvements in RMSE could gain practically significant improvement in recommendation [Koren, 2008].

Similar results are observed on the *EachMovie* data (Fig. 1(c) and Fig. 1(d)). Meanwhile, we perform statistical t-test, which indicates that the improvement of RAPARE-MF is significant. For example, when the cold-start degree is 5, the p-value is less than 0.05 in all four settings (i.e., two cold-start scenarios on two data sets), against the corresponding best competitors. Overall, these results show the effectiveness of the proposed method for the cold-start problem.

4.3 Efficiency Results

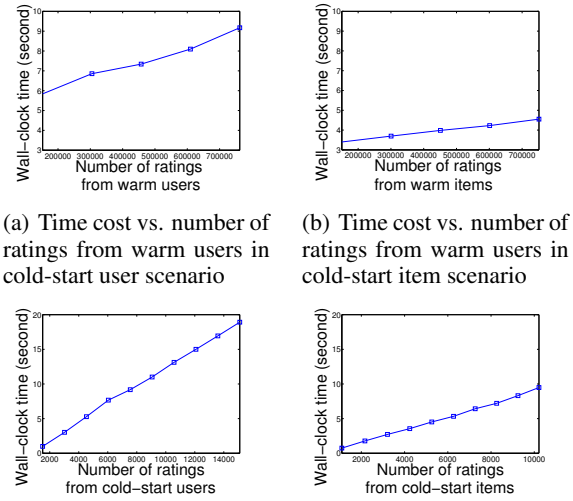
Next, we present the efficiency results of the compared methods. We still report the results on the *MovieLens* data, while similar results are observed on the *EachMovie* data. All the experiments are run on a Macbook Pro. The machine has four 2.2GHz Intel i7 Cores and 8GB memory.

For efficiency, we first evaluate the quality-speed balance of different methods in both cold-start user and cold-start item scenarios. The results are shown in Fig. 2. 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. 2(a) and Fig. 2(b), our RAPARE-MF shows good balance between quality and speed



(a) Cold-start user scenario (b) Cold-start item scenario

Figure 2: 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).



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

(b) Time cost vs. number of ratings from warm users 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 3: Scalability of RAPARE-MF. RAPARE-MF scales linearly wrt the data size.

in both cold-start user and cold-start item scenarios. 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. 3. In the figures, we show the scalability of our method in terms of both the number of ratings from warm users/items (Fig. 3(a) and Fig. 3(b)) and the number of ratings from cold-start users/items (Fig. 3(c) and Fig. 3(d)). As we can see from the figures, our proposed RAPARE-MF scales linearly in all four cases.

5 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. Collaborative filtering (CF) has become an indispensable building block in many recommender systems. 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 [Koren *et al.*, 2009; Shapira, 2011].

Recommendations for Cold-Start Scenarios. 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, an interview process where a set of items are usually presented for the cold-start users to provide their ratings [Zhou *et al.*, 2011; Golbandi *et al.*, 2011; Harpale and Yang, 2008; Sun *et al.*, 2013]. The main focus in different interview processes lies in the selection of a proper item set. However, Rashid et al. [Rashid *et al.*, 2002] 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 [Zhang *et al.*, 2014] and social relationships [Lin *et al.*, 2013; Yao *et al.*, 2014] to tackle the cold-start problem. However, these methods become infeasible in many real recommender systems where such side information may not be available.

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. [Sarwar *et al.*, 2002] introduce an incremental SVD (iSVD) algorithm for the cold-start users; Tackcs et al. [Takács *et al.*, 2008] and Rendle et al. [Rendle and Schmidt-Thieme, 2008] also provide incremental algorithms to update the latent factor vectors for cold-start users when they give new ratings. Our method falls into this class, and we propose a rating comparison model to give special treatments to the cold-start users/items.

6 Conclusions

In this paper, we have proposed a rating comparison strategy (RAPARE) to make proper recommendations for cold-start users/items. In particular, the RAPARE strategy provides a special, fine-grained treatment for cold-start users and cold-start items. Instantiating RAPARE with the matrix factorization method, we further propose the RAPARE-MF model as well as an effective and efficient algorithm to solve it. Experimental evaluations on two real data sets show that our approach outperforms several benchmark collaborative filtering and online updating methods in terms of prediction accuracy, and it provides fast recommendations with linear scalability.

7 Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61373011, 91318301, 61321491). This material is supported by the National Science Foundation under Grant No. IIS1017415, by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053,by Defense Advanced Research Projects Agency (DARPA) under Contract Number W911NF-11-C-0200 and W911NF-12-C-0028, by National Institutes of Health under the grant number R01LM011986, Region II University Transportation Center under the project number 49997-33 25.

The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[Das *et al.*, 2007] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *WWW*, pages 271–280. ACM, 2007.

[Davidson *et al.*, 2010] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. The youtube video recommendation system. In *RecSys'10*, pages 293–296. ACM, 2010.

[Elo, 1978] Arpad E Elo. *The rating of chessplayers, past and present*, volume 3. Batsford London, 1978.

[Golbandi *et al.*, 2011] Nadav Golbandi, Yehuda Koren, and Ronny Lempel. Adaptive bootstrapping of recommender systems using decision trees. In *WSDM'11*, pages 595–604. ACM, 2011.

[Harpale and Yang, 2008] Abhay S Harpale and Yiming Yang. Personalized active learning for collaborative filtering. In *SIGIR*, pages 91–98. ACM, 2008.

[Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *SIGKDD*, pages 426–434. ACM, 2008.

[Lin *et al.*, 2013] Jovian Lin, Kazunari Sugiyama, Min-Yen Kan, and Tat-Seng Chua. Addressing cold-start in app recommendation: latent user models constructed from twitter followers. In *SIGIR*, pages 283–292. ACM, 2013.

[Linden *et al.*, 2003] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[Merve Acilar and Arslan, 2009] A Merve Acilar and Ahmet Arslan. A collaborative filtering method based on artificial immune network. *Expert Systems with Applications*, 36(4):8324–8332, 2009.

[Rashid *et al.*, 2002] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K Lam, Sean M McNee, Joseph A Konstan, and John Riedl. Getting to know you: learning new user preferences in recommender systems. In *IUI*, pages 127–134. ACM, 2002.

[Rendle and Schmidt-Thieme, 2008] Steffen Rendle and Lars Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *RecSys'08*, pages 251–258. ACM, 2008.

[Resnick and Varian, 1997] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[Sarwar *et al.*, 2000] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000.

[Sarwar *et al.*, 2001] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295. ACM, 2001.

[Sarwar *et al.*, 2002] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*, pages 27–28. Citeseer, 2002.

[Schafer *et al.*, 2007] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.

[Shapira, 2011] Bracha Shapira. *Recommender systems handbook*. Springer, 2011.

[Sun *et al.*, 2013] Mingxuan Sun, Fuxin Li, Joonseok Lee, Ke Zhou, Guy Lebanon, and Hongyuan Zha. Learning multiple-question decision trees for cold-start recommendation. In *WSDM'13*, pages 445–454. ACM, 2013.

[Takács *et al.*, 2008] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Investigation of various matrix factorization methods for large recommender systems. In *ICDMW'08*, pages 553–562. IEEE, 2008.

[Töscher *et al.*, 2009] Andreas Töscher, Michael Jahrer, and Robert M Bell. The bigchaos solution to the netflix grand prize. *Netflix prize documentation*, 2009.

[Yao *et al.*, 2014] Yuan Yao, Hanghang Tong, Guo Yan, Feng Xu, Xiang Zhang, Boleslaw Szymanski, and Jian Lu. Dual-regularized one-class collaborative filtering. In *CIKM*, 2014.

[Zhang *et al.*, 2014] Mi Zhang, Jie Tang, Xuchen Zhang, and Xiangyang Xue. Addressing cold start in recommender systems: A semi-supervised co-training algorithm. In *SIGIR*, 2014.

[Zhou *et al.*, 2011] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. Functional matrix factorizations for cold-start recommendation. In *SIGIR*, pages 315–324. ACM, 2011.