Noname manuscript No.
(will be inserted by the editor)

Multi-level Preference Regression for Cold-start Recommendations

Furong Peng [1,3] · Xuan Lu [2] · Chao Ma [1] · Yuhua Qian [3] · Jianfeng Lu [1] · Jingyu Yang [1]

Received: date / Accepted: date

Abstract Due to the absence of historical ratings of new users/items, cold-start recommendation remains a challenge for collaborative filtering. Many matrix factorization based methods are used to predict new user's/item's latent profile before predicting ratings. This kind of methods is usually non-convex. In this work, we design a new convex framework for cold-start recommendations, multi-level preference regression (MPR), directly to predict the ratings rather than latent profiles. We suppose that ratings are mainly affected by three components: 1) correlation between user's attributes (such as age and gender) and item's attributes (such as genre and producer); 2) each user's preference on item's attributes; 3) item's popularity in a group of users with some attributes. Adjusting the impact of the three components, we can tackle three cold-start scenarios of user, item, and system. In the MPR framework, three different learning strategies are discussed: pointwise regression, pairwise regression, and large-margin learning. Experimental results on three datasets demonstrate that the proposed model can achieve the state of the art in the user cold-start scenario and the best performance in other scenarios.

Keywords Recommendation System · Cold-start Recommendation · Preference Regression

✉Jianfeng Lu
E-mail: lujf@njust.edu.cn

1. School of Computer Sci. & Eng., Nanjing Univ. of Sci. and Tech., Nanjing, Jiangsu 210094, P.R.China
2. Department of Electronic Info. Eng., Shanxi Univ., Taiyuan, Shanxi, 030013, P.R.China
3. School of Computer & Info. Tech., Shanxi Univ., Taiyuan, Shanxi, 030006, P.R.China

1 Introduction

Recommender systems (RSs) are tools to help people deal with the information overload problem. RSs recommend items (such as movies, books, and products) for particular users according to their interest [1–3]. RSs are essential for nowadays online services and stores, such as Amazone[1], Netflix[2], Taobao[3], and Facebook[4]. A comprehensive literature review of the application of RSs can be found in a recent work [4].

RSs can be classified into four categories: 1) collaborative filtering, 2) demographic filtering, 3) content-based filtering, 4) hybrid filtering. The most widely used method is Collaborative Filtering (CF), which recommends items based on the preference of users with similar historical ratings [5–7]. CF uses nothing of users and items but their past ratings. However, for the new users and items, CF is not applicable. This challenge is nominated as cold-start recommendations [8–12]. Auxiliary information of new users and items are often required to solve cold-start problems.

Different sources of auxiliary information can be applied to cold-start problems. 1) Interview of new users can be used to extract their interest for user cold-start scenario [13–16]. 2) New user's friends in social networks have also been proved useful for cold-start problems [17–19]. 3) Through analyzing user's demographic information, reliable recommendations can be produced for new users [20–23]. 4) The content of new items can be used to calculate the correlation between new items and existing ones, and consequently can be used for item cold-start scenario [24–26]. However, the content

[1] http://www.amazone.com
[2] https://www.netflix.com
[3] https://www.taobao.com
[4] https://www.facebook.com

of items may be heterogeneous, such as the synopsis, poster, and trailer of a movie. A user may not like to share their social networks with the recommendation system. These factors limit the application of existing methods. In this work, we mainly focus on utilizing users' demographic information and items' attributes to deal with cold-start recommendations. User's demographics are easier to access such as age, gender, occupation, living location, and nationality. Usually, this kind of information can be acquired when users sign up. For ease of description, we use users' attribute to describe users' demographic information. The item's attributes can be obtained when the producer release products, such as category, publisher, publication year. This kind of information commonly exists in real life world.

Many methods [20,27,21,28–30,9,31] have proved that it is efficient to solve cold-start recommendations with users' and items' attributes. Apart from user and item cold-start scenarios, attributes can also be used for the system cold-start scenario, which requires to recommend new items to new users. For such kinds of cold-start recommendations, most existing models are matrix factorization based ones [29,30,9,28,32,31], which predict new users' and new items' latent profile before predicting ratings. However, the matrix factorization based methods are non-convex and arduous to implement. We will discuss a simpler linear framework with an analytical solution.

In this work, we propose a novel framework, multi-level preferences regression (MPR), that directly predicts ratings with multi-level preferences. Three-level preferences are involved. 1) attribute-attribute preference describes the choice of a group of users with some attributes for a group of items with specific attributes. 2) user-attribute preference describes a particular user's preference on a group of items with some attributes. 3) attribute-item preference captures the preference of a group of users with some attributes on a specific item. Using the three different preferences, the proposed model can deal with three cold-start scenarios of user, item, and system. Compared with matrix factorization based models, it is a linear regression framework with an analytical solution. We analyze the regression framework with different learning strategies: pointwise regression [8], pairwise regression [8] and large-margin learning [33,34]. The proposed model is estimated on three datasets for three cold-start scenarios.

The rest of this paper is organized as follows. A literature review is presented in Section 2. Some necessary notations and a basic model are depicted in Section 3. The proposed model MPR is introduced in Section 4 and followed by experiments in Section 5. We conclude this work in Section 6.

2 Related work

As discussed in many literature reviews, matrix factorization is the most popular collaborative filtering method for RSs [35,36]. For cold-start problems, many methods were proposed based on matrix factorization. The underlined presumption is that user's and item's latent profile can be predicted by their attributes respectively. Agarwal and Chen [29] tried to predict latent profiles with linear regression. Zhang *et al.*[30] extended the regression to a tree-based one, because the linear regression is not adaptive for general cases. Gantner *et al.*[9] proposed a Bayesian pairwise regression based linear mapping. In the model, the latent profile and mapping are trained separately. Peng *et al.*[31] proposed learning latent profile with Markov random field (MRF) constraint, such that users with similar attribute have similar latent profiles. With MRF constraint, the profile of new users/items can be predicted with neighbors in the attribute space. All these matrix factorization methods need to predict latent profiles before inferring preference, and do not have analytical solutions.

Along with predicting latent profiles, multiple matrix factorization techniques are also applied for cold-start problems. Saveski and Mantrach [32] proposed simultaneously factorizing user-item matrix and item content matrix in the item cold-start scenario. Barjasteh *et al.*[37,28] proposed a two-stage algorithm that firstly factorizes user-item matrix and then factorizes user similarity matrix and item similarity matrix. They argued that simultaneously factorizing user-item matrix and similarity matrix will cause error propagation. However, computing the similarity matrix of items or users may become prohibitive in a large-scale dataset.

Although matrix factorization based models achieved much success in cold-start scenarios, they are complicated to implement and hard to search for the optimal parameters. In some cases, a simple rule-based and linear regression methods can give considerable results [38, 8]. Park and Chu [38,8] proposed a pairwise preference regression, named PPR. The joint of user's attribute and item's attribute were considered to predict user's preference. For better performance, a pairwise learning strategy was applied. As the attributes can only discriminate people and item in groups, the feature of individual user or item was ignored. They applied filterbot tehcnique [39] as an extended feature to describe user's and item's unique character. However, the filterbot techniques may drop off the performance in cold-start scenarios due to overfitting. In this work, we pro-

pose to learn user's and item's individual character, and simplify the model at the same time.

3 Preliminaries

Before describing the proposed model, we first define some necessary notations and discuss a linear regresion model.

3.1 Notations

The user-item rating matrix is denoted as $\mathbf{R} \in \mathbb{R}^{n \times m}$, where n and m are the numbers of users and items respectively. Usually, $\mathbf{R}$ is a very sparse matrix that only has a very little portion of observed elements which are indexed by $\mathcal{O}$. For clearness, we use u and i to represent a particular user and item respectively. $\mathcal{O}_u$ is the item set that is rated by u, and $\mathcal{O}_i$ is the user set who have rated i. Suppose that a user's attribute is $\mathbf{A}_u \in \mathbb{R}^{1 \times p}$, and an item's attribute is $\mathbf{B}_i \in \mathbb{R}^{1 \times q}$, where p and q are the numbers of users' and items' attributes respectively. A predicted rating is denoted as $\hat{\mathbf{R}}_{ui}$.

3.2 Pairwise Preference Regression

An efficient method for cold-start recommendation is pairwise preference regression that was proposed by Park and Chu[39]. They assumed that ratings could be predicted based on the features of users and items. User's feature is supposed to be concatenated by attributes and filterbots, and is denoted as $\hat{\mathbf{A}}_u = [\mathbf{A}_u, \tilde{\mathbf{A}}_u]$ where $\tilde{\mathbf{A}}_u$ indicates the filterbot features that are extracted according to user's historical ratings. $\hat{\mathbf{A}}_u$ can not only represent the preference of a group of users with specific attributes but also represent a user's individual preference. Similarly, item's feature is denoted as $\hat{\mathbf{B}}_i = [\mathbf{B}_i, \tilde{\mathbf{B}}_i]$ where $\tilde{\mathbf{B}}_i$ is item's filterbot features. $\hat{\mathbf{B}}_i$ can simultaneously represent group-level character and individual-level character. With these notations, a rating can be predicted as

$$\hat{\mathbf{R}}_{u,i} = \hat{\mathbf{A}}_u \mathbf{W} \hat{\mathbf{B}}_i^T, \tag{1}$$

where $\mathbf{W}$ is the model parameter. This formulation is illustrated in Fig. 1.

Applying the pointwise regression, we can obtain the following object function

$$\min_{\mathbf{W}} \sum_{u,i \in \mathcal{O}} (\mathbf{R}_{u,i} - \hat{\mathbf{R}}_{u,i})^2 + \lambda ||\mathbf{W}||_F^2, \tag{2}$$

where λ is the regularization coefficient. In RSs, users may have different rating bias. Park and Chu proposed to employ pairwise regression that is not only capable of removing user bias but also has an analytical solution. The Pairwise regression can be described as

$$\min_{\mathbf{W}} \sum_{u} \frac{1}{|\mathcal{O}_u|} \sum_{i,j \in \mathcal{O}_u} \Big((\mathbf{R}_{u,i} - \mathbf{R}_{u,j}) - (\hat{\mathbf{R}}_{u,i} - \hat{\mathbf{R}}_{u,j}) \Big)^2 + \lambda ||\mathbf{W}||_F^2, \tag{3}$$

where i, j are items that are rated by u, λ is regularization coefficient, $|\mathcal{O}_u|$ represents the number of items rated by user u.

Although PPR achieved much success in cold-start scenarios, the filterbots resulted in an inferior performance due to the rule-based strategy of building individual feature and overfitting. We will introduce a flexible way to learn the individual feature and deal with overfitting.

4 Multi-level Preference Regression

We first introduce the proposed MPR framework for cold-start scenarios, and then implement three different learning models underlying this framework. Owing to its user-level and item-level regression coefficients, the MPR possesses higher adaptability than PPR model. Compared with the existing matrix factorization based models, MPR is a simpler linear framework with analytical solutions.

4.1 Multi-level Preference

By analyzing the cold-start scenarios, we found that user's rating can be composed of three parts: 1) the correlations between user's and item's attributes, 2) each user's preference in item's attributes, 3) item's popularity in a group of users with some attribute. If we use the attributes to represent a group of users or items, then the first part represents group-level preference, the second part describes a user's preference in a group of items, and the third part depicts the preference of a group of users in an item. Based on this observation, we name the proposed framework as **M**ultiple-level **P**reference **R**egression (MPR).

This MPR framework can be evolved from PPR. When splitting the regression coefficient $\mathbf{W}$ into four sub-matrices as shown in Fig. 1, we can obtain the rating prediction model of PPR in Equation 1 as

$$\begin{aligned} \hat{\mathbf{R}}_{u,i} =& \mathbf{A}_u \mathbf{W}_{1,1} \mathbf{B}_i^T + \tilde{\mathbf{A}}_u \mathbf{W}_{2,1} \mathbf{B}_i^T \\ &+ \mathbf{A}_u \mathbf{W}_{1,2} \tilde{\mathbf{B}}_i^T + \tilde{\mathbf{A}}_u \mathbf{W}_{2,2} \tilde{\mathbf{B}}_i^T. \end{aligned} \tag{4}$$

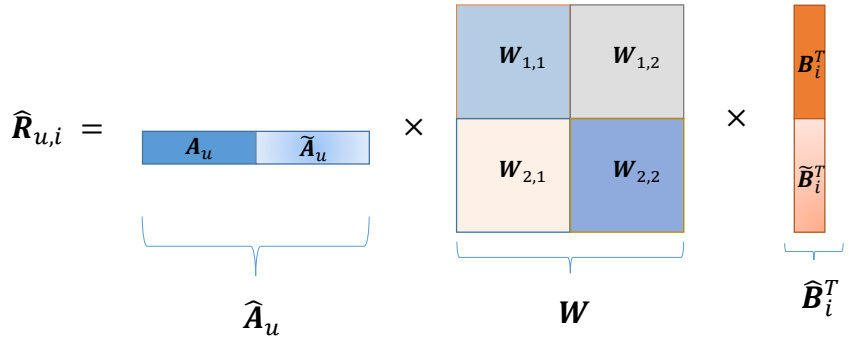


Fig. 1: Rating prediction model used by PPR

$\mathbf{A}_u\mathbf{W}_{1,1}\mathbf{B}_i^T$ represents the first part: correlations between user's and item's attributes. $\tilde{\mathbf{A}}_u\mathbf{W}_{2,1}\mathbf{B}_i^T$ represents the second part: u's preference on item's attributes $\mathbf{B}_i$. $\mathbf{A}_u\mathbf{W}_{1,2}\tilde{\mathbf{B}}_i^T$ represents the third part: i's popularity in a group of users with the attribute $\mathbf{A}_u$. Because at least user's or item's historical ratings are not existing in cold-start scenarios, $\tilde{\mathbf{A}}_u$ or $\tilde{\mathbf{B}}_i$ in the fourth term is not available. Fitting $\tilde{\mathbf{A}}_u\mathbf{W}_{2,2}\tilde{\mathbf{B}}_i^T$ in training phase will lead to overfitting and inferior performance. In this work, we consider this part as zero-mean Gaussian noise.

Besides removing $\tilde{\mathbf{A}}_u\mathbf{W}_{2,2}\tilde{\mathbf{B}}_i^T$, we regard $\tilde{\mathbf{A}}_u$ and $\tilde{\mathbf{B}}_i$ as variables for better-describing user's and item's individual character instead of filterbot features. Absorbing $\mathbf{W}_{2,2}$ into $\tilde{\mathbf{A}}_u$, we define $\mathbf{U}_u = \tilde{\mathbf{A}}_u\mathbf{W}_{2,1}$ to describe user u's unique character. Similarly, we define $\mathbf{V}_i = \mathbf{W}_{1,2}\tilde{\mathbf{B}}_i^T$ to describe item i's feature. Based on those definitions, we obtain the MPR framework to predict ratings

$$\hat{\mathbf{R}}_{u,i} = \mathbf{A}_u\mathbf{W}\mathbf{B}_i^T + \mathbf{U}_u\mathbf{B}_i^T + \mathbf{A}_u\mathbf{V}_i^T. \tag{5}$$

We repeatly use $\mathbf{W}$ for regression weights, when having no confusion. A general regression framework can be described as

$$\mathcal{L} = \Phi(\mathbf{R}, \hat{\mathbf{R}}) + \Omega(\mathbf{W}, \mathbf{U}, \mathbf{A}), \tag{6}$$

where Φ is a general regression function and Ω is a regularization function. If Φ and Ω are both convex functions, then we can obtain an analytical solution. Based on this framework, we discuss three different regression models: 1) pointwise regression [8], 2) pairwise regression [8], and 3) large-margin learning [33].

4.2 Pointwise Learning

In pointwise settings, the predicted ratings are supposed to be as close to real ratings as possible. We can obtain the following optimization problem.

$$\begin{aligned}\min_{\mathbf{W},\mathbf{U},\mathbf{V}} &\sum_{u,i\in\mathcal{O}} (\mathbf{R}_{u,i} - \hat{\mathbf{R}}_{u,i})^2 \\ &+ \lambda_w||\mathbf{W}||_F^2 + \lambda_u||\mathbf{U}||_F^2 + \lambda_v||\mathbf{V}||_F^2,\end{aligned} \tag{7}$$

where λ_w, λ_u and λ_v are three regularization coefficients that control the model complexity in different cold-start scenarios. For example, because $\mathbf{U}$ of new users is not inferable in user cold-start scenario, we set λ_u a large value to avoid overfitting.

The analytical solution of the pointwise learning model can be deduced in the following way. Let

$$\mathbf{w} \equiv [vec(\mathbf{W}), vec(\mathbf{U}), vec(\mathbf{I})], \tag{8}$$

where $vec(\cdot)$ converts a matrix to a row vector.

$$\mathbf{x}^{(1)}(u,i) \equiv vec(\mathbf{A}_u \otimes \mathbf{B}_i), \tag{9}$$

where $\otimes$ represents Kronecker product.

$$\mathbf{x}^{(2)}(u,i) \equiv [\mathbf{0}, \cdots, \mathbf{0}, \mathbf{B}_i, \mathbf{0}, \cdots, \mathbf{0}], \tag{10}$$

$\mathbf{x}^{(2)}(u,i) \in \mathbb{R}^{1\times nq}$ is concatnated by n $\mathbf{0}$ vectors, and the u-th $\mathbf{0}$ is replaced with $\mathbf{B}_i$.

$$\mathbf{x}^{(3)}(u,i) \equiv [\mathbf{0}, \cdots, \mathbf{0}, \mathbf{A}_u, \mathbf{0}, \cdots, \mathbf{0}], \tag{11}$$

$\mathbf{x}^{(3)}(u,i) \in \mathbb{R}^{1\times mp}$ is composed of m $\mathbf{0}$ vectors, and the i-th $\mathbf{0}$ vector is replaced with $\mathbf{A}_u$. Concatnating $x_{ui}^{(1)}$, $x_{ui}^{(2)}$ and $x_{ui}^{(3)}$, we obtain a very sparse data vector

$$\mathbf{x}(u,i) \equiv [\mathbf{x}^{(1)}(u,i), \mathbf{x}^{(2)}(ui), x^{(3)}(u,i)], \tag{12}$$

where $\mathbf{x}(u,i) \in \mathbb{R}^{1\times(pq+nq+mp)}$. Let $\mathbf{X}$ be the data matrix whose rows belong to $\{\mathbf{x}(u,i)|u,i \in \mathcal{O}\}$. Denote $\mathbf{y}$ as the target vector that is composed of $\{\mathbf{R}_{u,i}|u,i \in \mathcal{O}\}$. Using some algebra operations, we obtain the following loss function

$$\mathcal{L} = ||\mathbf{w}\mathbf{X}^T - \mathbf{y}||_F^2 + \mathbf{w}\mathbf{D}\mathbf{w}^T, \tag{13}$$

where $\mathbf{D}$ is a diagonal matrix of coeficients

$$\begin{aligned}\mathbf{D} \equiv diag(&D_{(1,1)} = \lambda_w, \cdots, D_{(pq,pq)} = \lambda_w,\\ &D_{(pq+1,pq+1)} = \lambda_u, \cdots, D_{(pq+nq,pq+nq)} = \lambda_u,\\ &D_{(pq+nq+1,pq+nq+1)} = \lambda_i,\\ &\cdots, D_{(pq+nq+mp,pq+nq+mp)} = \lambda_i).\end{aligned} \tag{14}$$

Obvioursely, Eq.(13) is a ridge regression and its analytical solution is

$$\mathbf{w}^* = \mathbf{y}\mathbf{X}^T(\mathbf{X}^T\mathbf{X} + \mathbf{D})^{-1}. \tag{15}$$

However, $\mathbf{X}^T\mathbf{X} + \mathbf{D}$ is a very large sparse matrix with the size of $(pq + nq + mp) \times (pq + nq + mp)$. Take Movielens-100k dataset as an example. The matrix is about 80000×80000. Computing the inverse of such a large sparse matrix is time-consuming and will generate unstable results. To avoid computing the prohibitive inverse operations, the gradient descent can be applied

$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \eta\frac{\partial\mathcal{L}}{\partial\mathbf{w}}, \tag{16}$$

where the gradient is

$$\frac{\partial\mathcal{L}}{\partial\mathbf{w}} = \mathbf{w}(\mathbf{X}^T\mathbf{X} + \mathbf{D}) - \mathbf{y}\mathbf{X}, \tag{17}$$

and η is the learning rate. However, the learning rate is sometimes very trick in practices.

In this work, we apply alternating least squares (ALS) to optimize this model. Because the pointwise regression model has a global solution, ALS can return the unique solution. If fixing two variables of $(\mathbf{W}, \mathbf{U}, \mathbf{V})$, we can obtain the closed-form solution of the remaining one. Their solutions are

$$\begin{aligned}vec(\mathbf{W}) = &\left(\sum_{u,i\in\mathcal{O}}(\mathbf{R}_{u,i} - \mathbf{U}_u\mathbf{B}_i^T - \mathbf{A}_u\mathbf{V}_i^T)vec(\mathbf{A}_u \otimes \mathbf{B}_i)\right)\\ &\left(\sum_{u,i\in\mathcal{O}} vec(\mathbf{A}_u \otimes \mathbf{B}_i)^T vec(\mathbf{A}_u \otimes \mathbf{B}_i) + \lambda_w\boldsymbol{I}\right)^{-1},\end{aligned} \tag{18}$$

$$\begin{aligned}\mathbf{U}_u = &\left(\sum_{i\in\mathcal{O}_u}(\mathbf{R}_{u,i} - \mathbf{A}_u\mathbf{W}\mathbf{B}_i - \mathbf{A}_u^T\mathbf{V}_i)\mathbf{B}_i\right)\\ &\left(\sum_{i\in\mathcal{O}_u}\mathbf{B}_i^T\mathbf{B}_i + \lambda_u\boldsymbol{I}\right)^{-1},\end{aligned} \tag{19}$$

$$\begin{aligned}\mathbf{V}_i = &\left(\sum_{u\in\mathcal{O}_i}(\mathbf{R}_{u,i} - \mathbf{A}_u\mathbf{W}\mathbf{B}_i - \mathbf{U}_u^T\mathbf{B}_i)\mathbf{A}_u\right)\\ &\left(\sum_{u\in\mathcal{O}_i}\mathbf{A}_u^T\mathbf{A}_u + \lambda_v\boldsymbol{I}\right)^{-1}.\end{aligned} \tag{20}$$

$\mathbf{U}$ and $\mathbf{V}$ are solved in rows for efficiency. Because the solutions of rows of $\mathbf{U}$ and $\mathbf{V}$ are independent, rows can be updated in parallel. ALS can get convergence in dozens of iterations. In our experiments, we set the number of iterations to 50.

4.3 Pairwise Learning

In RSs, users may have different bias when making ratings. For example, a user may commonly vote higher/lower ratings than others. With this observations, the pairwise regression was proposed for learning user's preference [5]. Based on MPR framework, we can obtain the following pairwise regression model

$$\begin{aligned}\min_{\mathbf{W},\mathbf{U},\mathbf{V}}\sum_u\frac{1}{|\mathcal{O}_u|}\sum_{i,j\in\mathcal{O}_u}&\Big((\mathbf{R}_{u,i} - \mathbf{R}_{u,j}) - (\hat{\mathbf{R}}_{u,i} - \hat{\mathbf{R}}_{u,j})\Big)^2\\ &+ \lambda_w||\mathbf{W}||_F^2 + \lambda_u||\mathbf{U}||_F^2 + \lambda_v||\mathbf{V}||_F^2.\end{aligned} \tag{21}$$

It is also a convex optimization problem with an analytical solution. In order to avoid computing the inverse of a large sparse matrix, ALS optimization framework is applied. In each sub-optimization problem, we have a closed-form solution again. When fixing others, each variable's solution is listed as follows.

$$\begin{aligned}vec(\mathbf{W}) = &(\sum_u\sum_{i\in\mathcal{O}_u}(\mathbf{U}_u(\bar{\mathbf{B}}_u - \mathbf{B}_i)^T + (\bar{\mathbf{V}}_u - \mathbf{V}_i)\mathbf{A}_u^T\\ &- (\bar{\mathbf{R}}_u - \mathbf{R}_{ui}))vec(\mathbf{A}_u \otimes \mathbf{B}_i))\\ &(\sum_u\sum_{i\in\mathcal{O}_u} vec(\mathbf{A}_u \otimes \mathbf{B}_i)^T vec(\mathbf{A}_u \otimes (\mathbf{B}_i - \bar{\mathbf{B}}_u)\\ &+ \frac{1}{2}\lambda_w\mathbf{I})^{-1},\end{aligned} \tag{22}$$

$$\begin{aligned}\mathbf{U}_u = &(\sum_{i\in\mathcal{O}_u}(\mathbf{A}_u\mathbf{W}(\bar{\mathbf{B}}_u - \mathbf{B}_i)^T + (\bar{\mathbf{V}}_u - \mathbf{V}_i)\mathbf{A}_u^T\\ &- (\bar{\mathbf{R}}_u - \mathbf{R}_{ui}))\mathbf{B}_i)\\ &(\sum_{i\in\mathcal{O}_u}\mathbf{B}_i^T(\mathbf{B}_i - \bar{\mathbf{B}}_u) + \frac{1}{2}\lambda_u\mathbf{I})^{-1},\end{aligned} \tag{23}$$

$$\begin{aligned}\mathbf{V}_i = &(\sum_{u\in\mathcal{O}_i}(1 - \frac{1}{|\mathcal{O}_u|})(\mathbf{A}_u\mathbf{W}(\bar{\mathbf{B}}_u - \mathbf{B}_i)^T + \mathbf{U}_u(\bar{\mathbf{B}}_u - \mathbf{B}_i)^T\\ &+ \frac{\sum_{j\in\mathcal{O}_u\setminus\{\mathbf{V}_i\}}\mathbf{V}_j}{|\mathcal{O}_u|}\mathbf{A}_u^T - (\bar{\mathbf{R}}_u - \mathbf{R}_{ui}))\mathbf{A}_u)\\ &\left(\sum_{u\in\mathcal{O}_i}(1 - \frac{1}{|\mathcal{O}_u|})\mathbf{A}_u^T\mathbf{A}_u + \frac{1}{2}\gamma_v\mathbf{I}\right)^{-1}.\end{aligned} \tag{24}$$

In these solutions, we define

$$\bar{\mathbf{B}}_u = \sum_{i\in\mathcal{O}_u}\mathbf{B}_i/|\mathcal{O}_u|, \tag{25a}$$

$$\bar{\mathbf{V}}_u = \sum_{i \in \mathcal{O}_u} \mathbf{V}_i / |\mathcal{O}_u|, \tag{25b}$$

$$\bar{\mathbf{R}}_u = \sum_{i \in \mathcal{O}_u} \mathbf{R}_{u,i} / |\mathcal{O}_u|. \tag{25c}$$

The number of training samples is an order of magnitude higher than that of pointwise regression, due to the pairwise strategy. However, the computational complexity is similar to that of pointwise when applying ALS.

4.4 Large-margin Learning

Ranksvm is one of the popular methods for ranking task. Compared to the pairwise regression, ranksvm tries to classify preference pairs and enlarge classification margin as large as possible[33,34]. In RSs, the large-margin strategy can also be introduced. Denote a user's preference set as $\mathcal{P}_u \equiv \{(i,j)|\mathbf{R}_{u,i} > \mathbf{R}_{u,j}\}$, then we can obtain the following large-margin model.

$$\begin{aligned} \min_{\mathbf{W},\mathbf{U},\mathbf{V}} \sum_u \sum_{(i,j)\in\mathcal{P}_u} & max(0, 1 - (\hat{\mathbf{R}}_{u,i} - \hat{\mathbf{R}}_{u,j})) \\ & + \lambda_w ||\mathbf{W}||_F^2 + \lambda_u ||\mathbf{U}||_F^2 + \lambda_v ||\mathbf{V}||_F^2. \end{aligned} \tag{26}$$

Although this model contains three variable to optimize, if we use $\hat{\mathbf{R}}_{ui} = \mathbf{w} x_{ui}^T$ as we did in Eq. 13, then the SVM optimization method can be applied to our model. Eq. 26 can be written as

$$\begin{aligned} \min_{\mathbf{W},\mathbf{U},\mathbf{V}} \sum_u \sum_{(i,j)\in\mathcal{P}_u} & max(0, 1 - \mathbf{w}(\mathbf{x}(u,i) - \mathbf{x}(u,j))^T) \\ & + \mathbf{w}\mathbf{D}\mathbf{w}^T. \end{aligned} \tag{27}$$

However, $\mathbf{x}(u,i)$ is a high dimensional sparse vector: a vector $pq + nq + mp$ contains $pq + p + q$ non-zero elements, where n and m commonly takes $10^3 \sim 10^4$. The training samples will be more than tens of millions and dimensions will be more than tens of thousands. It is very challenging to solve such large-scale optimization problem. Lee and Lin proposed a technique of order-statistic tree to speed up training[34]. In this work, we apply this technique for efficiency. As the samples of such model are very sparse, the kernel ranksvm can not significantly improve performance but dramatically slow down training speed. In experiments we mainly consider the linear ranksvm. For more details in order-statistic tree, please refer to the Ref.[34].

4.5 Inference

Since the three cold-start scenarios have different available data, we will apply difference inference strategies.

In the user cold-start scenario, because testing users do not have any historical ratings, the associated $\mathbf{U}$ is not trained in training set. However, the items of this scenario have already been in training set, and can be used for prediction. Meanwhile, attributes of both users and items are available, $\mathbf{W}$ can also be used for prediction. Consequently, the ratings in user cold-start scenario can be infered by

$$\hat{\mathbf{R}}_{u,i} = \mathbf{A}_u \mathbf{W} \mathbf{B}_i^T + \mathbf{V}_i \mathbf{A}_u^T. \tag{28}$$

In the item cold-start scenario, user's historical ratings are available but items' ratings are not. We predict ratings with

$$\hat{\mathbf{R}}_{u,i} = \mathbf{A}_u \mathbf{W} \mathbf{B}_i^T + \mathbf{U}_u \mathbf{B}_i^T. \tag{29}$$

In the system cold-start scenario, both user's and item's historical ratings are not available. We can only apply attributes to predict ratings.

$$\hat{\mathbf{R}}_{u,i} = \mathbf{A}_u \mathbf{W} \mathbf{B}_i^T. \tag{30}$$

5 Experiments

In this section, we evaluate the performance of MPR in three cold-start scenarios with a series of experiments.

5.1 Datasets

Three datasets, Bookcrossing[5], Movielens-100K[6], and Movielens-1M were chosen for evaluations. Because the attributes of both users and items are available in these datasets, all three cold-start scenarios can be assessed. The three datasets were summarized in Table 1.

5.2 Evaluation Metrics

In experiments, all algorithms return a list of recommendations, in which items with higher ratings for a user should be in more top ranks. To quantify the recommendations result, we employed four ranking based metrics: NDCG, Precision, Recall and F1 [8]. When computing Precision, Recall, and F1, the ratings larger than or equal to 4 in Movielens-100K and Movielens-1M, and 8 in Bookcrossing were regarded as positive

[5] http://www2.informatik.uni-freiburg.de/~cziegler/BX/
[6] http://grouplens.org/datasets/movielens/

Table 1: Overview of datasets

Datasets	#Users	#Items	#Ratings	Rating Rage	Sparsity	User Attributes	Item Attributes
Bookcrossing	270,170	92,107	1,031,175	1-10	99.9959%	location, age	publisher, year
Movielens 100K	1,681	943	100,000	1-5	93.7%	occupation, age, gender	genre, year
Movielens 1M	3,883	6,040	1,000,209	1-5	94.89%	occupation, age, gender	genre, year

samples and the rest were negative ones. Because people are commonly willing to accept about ten recommendations, we computed the metrics at top 10 of the recommendation list.

5.3 Evaluation Protocol

Ten-fold cross validation was applied as the protocol. Specifically, we divided users and items into ten groups simultaneously. At each testing time, one group of users and items were chosen as new users and new items, and the rest were considered as existing ones. Users and items were then split into four groups: training samples (existing users and existing items), user cold-start set (new users and existing items), item cold-start set (existing users and new items), and system cold-start set (new users and new items). The average result of the ten folds was reported as the final scores.

5.4 Baseline methods

Seven baseline methods and the proposed methods were compared in experiments.

Vibes Affinity (**VA**) [38,8] recommends items based on attributes of users and items with some rules that were inferred by conditional probability of historical ratings. This method was applied in several Yahoo! properties [8].

Pairwise Preference Regression (**PPR**)[8] is a linear regression model with pairwise strategies. Ratings were supposed to be predicted based on the joint-features of users and items. **PPR+filterbots** indicates the method of PPR that adds filterbots to the joint-features. More details can be found in Section 3.

Markov Random Field based Matrix Factorization (**MRF-MF** [31]) utilizes the Markov random filed to constrain the distribution of latent profile of users and items, and then predict latent profile with neighbors.

LightFM [40] is a hybrid matrix factorization method that represents users and items via their attributes' latent profiles. The model considers the recommendation as a classification problem. In Movielens dataset, ratings above or equal to 4 were considered as positive and the rest were negative for training. The positive and negative samples in Bookcrossing were splitted at the rating of 8.

Tag-Keyword Relation (**TKR**[11]) is a 3-factor matrix factorization model that is incorporated with a tag-keyword relation matrix. User's interest on tags and item's correlation with keywords were modeled. In our experiments, the tags were replaced by users' attributes and keywords by items' attributes.

DecRec[37] is a two-step matrix factorization based method that first completes a subset of the rating matrix and then transduces the existing completed ratings to new users and items. The singular value thresholding method[41] was applied to conduct nuclear matrix completion in our experiments. Before selecting the sub-matrix, the rating matrix was rearranged such that 1) users with the most ratings are on the top and 2) items with most ratings are on the left. A square sub-matrix with size $n \times n$ on the left-top was selected to complete. $\tau \cdot n$ was used as the regularization coefficient for nuclear matrix completion.

XGBoost [42] is a scalable tree boosting algorithm that was widely applied in information retrieval and classification. If we consider the attributes of users and items as features, then the cold-start recommendation can be regarded as an information retrieval task. The users' and items' attributes were concatenated as the input of XGBoost and the ratings were considered as the ranking label.

MPR(*pointwise*)(in Section 4.2) is the proposed model that utilizes pointwise regression as the loss function.

MPR(*pairwise*)(in Section 4.3) is the proposed model that utilizes pairwise regression as the loss function.

MPR(*large-margin*)(in Section 4.4) is the proposed model that utilizes large-margin as the loss function.

5.5 Performance Comparision

Three cold-start scenarios were simulated for comprehensive comparisons. We executed grid-search for reporting the best performance of each method. In the grid-search, the number of the latent profile was searched from 5 to 30 with a step of 5. The regularization coeffi-

cients were searched in $\{10^n|n = -3, \cdots 6\} \cup \{0\}$. Algorithms were all evaluated on three datasets except for DecRec and MRF-MF in Bookcrossing dataset. DecRec demanded too large memory for our computer workstation (64GM memory), due to calculating a similarity matrix of users and its eigenvalues. MRF-MF was very slow in Bookcrossing dataset for updating neighbor's latent profile, so that we could not conduct grid-search.

Table 2, 3, and 4 list the best performance of all algorithms in three cold-start scenarios. Results demonstrated that MPR could achieve the state of art (outperform the average of the top 3 baselines). We will analyze the experimental results by scenarios.

In user cold-start scenario, testing item's historical ratings are available and can be used to infer its individual popularity. XGBoost, VA, and PPR obtained lower performance than others because each testing item's popularity is ignored. Based on PPR, PPR+filterbots considered filterbot feature to describe individual item's popularity and got higher performance. MRF-MF, TKR, and LightFM are matrix factorization based methods that utilize the latent feature to describe testing item's popularity. Therefore, these methods could achieve the state of the art. Although DecRec is a matrix factorization based method, it doesn't consider individual item's popularity when making recommendations. The proposed MPR framework uses $\mathbf{V}$ to describe each item's popularity and thus could achieve high performances. Because MPR only utilized linear regression coefficients to describe item's popularity, its representative power was a little lower than matrix factorization based methods. However, MPR is simpler and has a global solution, for instance, MPR(*pointwise*).

In item cold-start scenario, each testing user's preference can be inferred by historical ratings. VA as a kind of rule-based method, that only considered user's attributes for recommendations, could not obtain as high performance as others. XGBoost and PPR got more accurate results than VA, owing to their model adaptability. After adding filterbots, PPR's performance didn't gain much improvement, due to overfitting of filterbots. Filterbots were created based on the ratings that were also used as regression target in the training phase. However, the testing ratings are unkown to PPR and could not be used to create filterbots. Matrix factorization based methods could obtain better performance benefiting from the latent profile that describe each user's preference. In this scenario, MPR obtained the best performance, because $\mathbf{U}$ modeled testing user's preference and item-level popularity was suppressed.

In system cold-start scenario, the recommendation can only be inferred by attributes. Although XGBoost outperformed VA, obtained lower score than PPR due to the sparsity of attributes. Linear regression method is more suitable than non-linear ones for a sparse dataset in most cases. The overfitting of filterbots led to lower scores that were more notable than the previous scenarios. Matrix factorization based methods could obtain better performance than PPR because each user's preference and items' popularity were modeled consistently with training scenarios other than Gaussian noise in PPR. MPR achieved the best performance, benefiting from appropriately modeling individual user's preference and individual item's popularity in the training phase. We will demonstrate that over-suppressing these terms will reduce the performance in the next section.

In MPR framework, pointwise learning method can output promising results, and sometimes can obtain the best performance, for instance, the item cold-start scenario of Bookcrossing. The pairwise regression and large-margin can usually get a little better performance. As the parameters of pairwise and large-margin in Movielens-1M and Bookcrossing were searched with a step size of 10^2 instead of 10 for saving time, their performance can be further improved. For simplicity, we suggest practicing pointwise regression before pairwise regression and large-margin.

5.6 Hyper Parameter Analysis

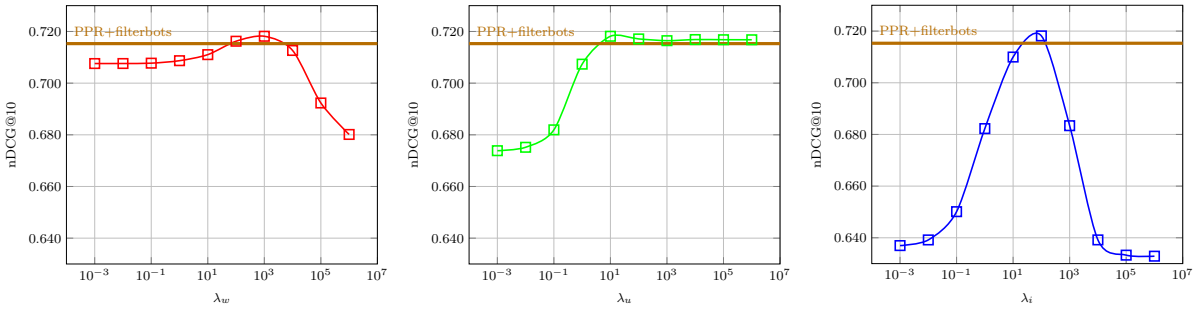
MPR has three parameters: λ_w, the regularization coefficient of attribute correlation weights; λ_u, the regularization coefficient of user's preference on item attributes; and λ_v, the regularization coefficient of item's popularity in the group of users with same attributes. We take MPR(*pointwise*) as an example to analyze parameters in three cold-start scenarios separately. In each cold-start scenario, the curve of NDCG@10 and regularization weights are plotted for analysis.

Hyperparameter impacts in user cold-start scenario are illustrated in Fig. 2, in which we can obtain the following three observations. 1) λ_w would reduce the performance with a large value. Because the correlation between user's and item's attributes are useful for inferring user's preference, a large value of λ_w would suppress this correlation and then impacts performances. 2) λ_u could improve NDCG@10 with a large value. Because individual user's preference can not be inferred in the testing phase, a large value of λ_u can adjust MPR to avoid overfitting training user's preference. 3) λ_i should be set carefully for a high performance (not too small nor too large). As the testing items' historical ratings are available for both training set and testing set, individual item's popularity in each group of users should be kept. λ_i should be set modestly in case of overfitting and underfitting.

Table 2: Results on Bookcrossing dataset

Cold-start scenarios	Algorithms	parameters	nDCG@10	Precision@10	Recall@10	F1@10
User Cold-start (Recommend Existing items to new users)	XGBoost	num-round=1000,subsample=0.4 η=0.1,λ=0.1,max-depth=4	0.5449	0.4437	0.6625	0.4795
	VA	—	0.5553	0.4493	0.6683	0.4844
	PPR	$\lambda = 10^3$	0.5415	0.4413	0.6586	0.4768
	PPR+filterbots	$\lambda = 10^4$	0.5717	0.4563	0.6718	0.4882
	LightFM	$d = 30, \alpha = 10^{-4}$	0.5647	0.4518	0.6691	0.4857
	TKR	$k = 10, \lambda = 10^2$	*0.5800*	*0.4596*	*0.6763*	*0.4916*
	Avg. of top 3 baselines	—	0.5721	0.4559	0.6724	0.4885
	MPR(*pointwise*)	$\lambda_w = 10^6, \lambda_u = 10^4, \lambda_v = 10^2$	**0.5818**	**0.4611**	**0.6767**	**0.4926**
	MPR(pairwise)	$\lambda_w = 10^4, \lambda_u = 10^2, \lambda_v = 10^0$	0.5605	0.4528	0.6690	0.4860
	MPR(large-margin)	$\lambda_w = 10^6, \lambda_u = 10^4, \lambda_v = 10^4$	0.5638	0.4524	0.6669	0.4847
Item Cold-start (Recommend New items to Existing Users)	XGBoost	num-round=1000,subsample=0.3 η=0.1,λ=0.1,max-depth=4	0.5720	0.4015	0.7280	0.4838
	VA	—	0.5702	0.3999	0.7262	0.4825
	PPR	$\lambda = 10^3$	0.5725	0.4024	0.7269	0.4838
	PPR+filterbots	$\lambda = 10^4$	0.5722	0.4025	0.7271	0.4839
	LightFM	$d = 25, \alpha = 0$	0.5817	0.4091	0.7352	0.4903
	TKR	$k = 10, \lambda = 10$	0.5734	0.4033	0.7296	0.4855
	Avg. of top 3 baselines	—	0.5759	0.4050	0.7309	0.4866
	MPR(*pointwise*)	$\lambda_w = 10^2, \lambda_u = 10^1, \lambda_v = 10^3$	**0.5930**	**0.4161**	**0.7434**	**0.4970**
	MPR(*pairwise*)	$\lambda_w = 10^6, \lambda_u = 10^1, \lambda_v = 10^6$	0.5887	0.4145	0.7418	0.4955
	MPR(*large-margin*)	$\lambda_w = 10^4, \lambda_u = 10^2, \lambda_v = 10^6$	*0.5925*	*0.4155*	*0.7432*	*0.4967*
System Cold-start (Recommend New Items to New Users)	XGBoost	num-round=1000,subsample=0.2 η=0.1,λ=0.001,max-depth=4	0.5755	0.4001	0.7252	0.4832
	VA	—	0.5741	0.3983	0.7237	0.4816
	PPR	$\lambda = 10^4$	*0.5767*	0.4005	0.7247	0.4830
	PPR+filterbots	$\lambda = 10^4$	0.5766	0.3990	0.7225	0.4814
	LightFM	$d = 25, \alpha = 0$	0.5750	0.3987	0.7240	0.4821
	TKR	$k = 10, \lambda = 0.1$	0.5765	0.3985	0.7244	0.4820
	Avg. of top 3 baselines	—	0.5766	0.3999	0.7248	0.4828
	MPR(*pointwise*)	$\lambda_w = 10^2, \lambda_u = 10^2, \lambda_v = 10^6$	0.5766	**0.4010**	**0.7274**	**0.4846**
	MPR(*pairwise*)	$\lambda_w = 10^6, \lambda_u = 10^2, \lambda_v = 10^2$	0.5735	0.4004	*0.7248*	*0.4836*
	MPR(*large-margin*)	$\lambda_w = 10^6, \lambda_u = 10^4, \lambda_v = 10^2$	**0.5786**	*0.4008*	0.7247	0.4835

Boldface indicates the best performance in each group.
Italics indicates the second best performance in each group.



(a) nDCG@10 versus λ_w (b) nDCG@10 versus λ_u (c) nDCG@10 versus λ_i

Fig. 2: Hyper parameters impact in user cold-start scenario

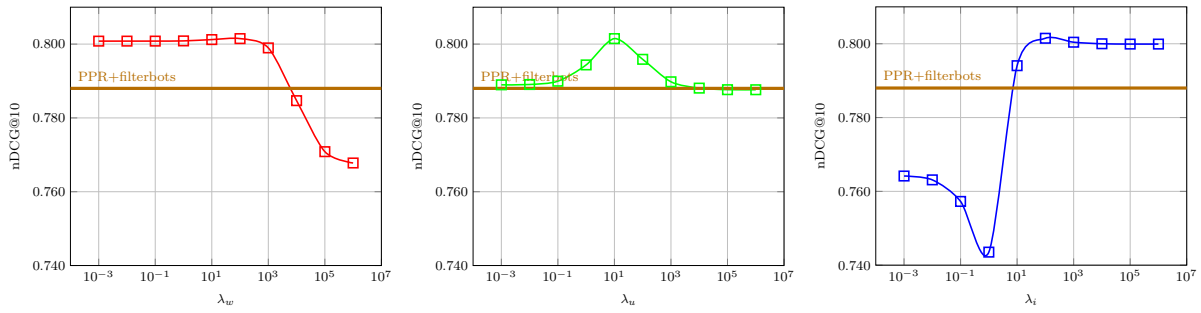
Fig. 3 illustrates the impact of parameters in item cold-start scenario. We could obtain the following three observations. 1) NDCG@10 began to drop when $\lambda_w > 10^3$. The reason is similar to the situation in user cold-start scenario. 2) MPR achieved the best performance when λ_u near 10. Because the testing users' preference could be inferred, keeping user's preference on item's attributes would raise NDCG@10. A very large (small) value of λ_u would lead to under-fitting(over-fitting). This situation is similar to the third observation in user cold-start scenario. 3) A large value of λ_v imporved MPR's performance. Because testing item's preference could not be inferred, fitting individual item's popularity in training phase resulted in over-fitting.

The impact of hyperparameters in system cold-start scenario is illustrated in Fig. 4. We can see that: 1) λ_w would reduce the performance with a large value; 2) a large value of λ_u and λ_v could improve the performance of MPR. Because both the preference of individual testing users and the popularity of individual testing items could not be inferred, we need to enlarge λ_u and λ_v to resist overfitting. It should be noted that an appropriately large value of λ_u and λ_v could further imporve MPR (for example $\lambda_u = 10^2$, $\lambda_v = 10^2$). Because very large values will degenerate the distribution of individual user's preference and individual item's popularity to be a zero-mean distribution with a very small variance. This assumption is too rigorous for MPR.

Table 3: Results on Movielens-100K dataset

Cold-start scenarios	Algorithms	parameters	nDCG@10	Precision@10	Recall@10	F1@10
User Cold-start (Recommend Existing items to new users)	XGBoost	num-round=800,subsample=0.4 η=0.1,λ=0.1,max-depth=5	0.6775	0.7431	0.2697	0.3460
	VA	—	0.6989	0.7720	0.2810	0.3608
	PPR	$\lambda = 10^3$	0.6341	0.7009	0.2564	0.3288
	PPR+filterbots	$\lambda = 10^4$	0.7153	0.7790	0.2830	0.3629
	MRF-MF	$d = 15, \lambda = 10, k = 100$	*0.7155*	**0.7839**	*0.2867*	**0.3672**
	TKR	$k = 10, \lambda = 10$	0.7119	0.7793	0.2839	0.3636
	LightFM	$d = 5, \alpha = 10^{-3}$	0.7112	0.7796	0.2826	0.3628
	DecRec	$\tau = 10, n = 400$	0.6268	0.6978	0.2538	0.3263
	Avg. of top 3 baselines	—	0.7142	0.7809	0.2845	0.3646
	MPR(*pointwise*)	$\lambda_w = 10^3, \lambda_u = 10^1, \lambda_v = 10^2$	**0.7181**	*0.7834*	0.2858	0.3662
	MPR(*pairwise*)	$\lambda_w = 10^4, \lambda_u = 10^6, \lambda_v = 10^2$	0.7104	0.7803	**0.2868**	*0.3664*
	MPR(*large-margin*)	$\lambda_w = 10^5, \lambda_u = 10^4, \lambda_v = 10^4$	0.7152	0.7817	0.2855	0.3653
Item Cold-start (Recommend New items to Existing Users)	XGBoost	num-round=800,subsample=0.4 η=0.1,λ=0.1,max-depth=3	0.7838	0.6342	0.7793	0.6609
	VA	—	0.7462	0.6061	0.7554	0.6365
	PPR	$\lambda = 10^2$	0.7882	0.6365	0.7825	0.6634
	PPR+filterbots	$\lambda = 10^3$	0.7880	0.6362	0.7822	0.6631
	MRF-MF	$d = 10, \lambda = 10, k = 30$	0.7903	0.6397	0.7850	0.6661
	TKR	$k = 10, \lambda = 10$	0.7862	0.6353	0.7808	0.6621
	LightFM	$d = 10, \alpha = 0$	0.7929	0.6417	0.7868	0.6681
	DecRec	$\tau = 10, n = 400$	0.7824	0.6334	0.7793	0.6605
	Avg. of top 3 baselines	—	0.7905	0.6393	0.7848	0.6659
	MPR(*pointwise*)	$\lambda_w = 10^2, \lambda_u = 10, \lambda_v = 10^2$	**0.8015**	0.6450	*0.7898*	*0.6708*
	MPR(*pairwise*)	$\lambda_w = 10^3, \lambda_u = 10^2, \lambda_v = 10^6$	*0.7997*	*0.6454*	**0.7911**	**0.6716**
	MPR(*large-margin*)	$\lambda_w = 10^5, \lambda_u = 10^4, \lambda_v = 10^5$	0.7988	**0.6455**	**0.7911**	**0.6716**
System Cold-start (Recommend New Items to New Users)	XGBoost	num-round=800,subsample=0.4 η=0.1,λ=0.1,max-depth=3	0.7789	0.6332	0.7786	0.6590
	VA	—	0.7389	0.6003	0.7484	0.6293
	PPR	$\lambda = 10^2$	0.7882	0.6346	0.7819	0.6609
	PPR+filterbots	$\lambda = 10^3$	0.7874	0.6336	0.7811	0.6600
	MRF-MF	$d = 5, \lambda = 10^1, k = 30$	0.7817	0.6337	0.7794	0.6598
	TKR	$k = 10, \lambda = 10^{-4}$	0.7803	0.6293	0.7742	0.6551
	LightFM	$d = 15, \alpha = 0$	**0.7902**	0.6343	0.7820	0.6611
	DecRec	$\tau = 10, n = 600$	0.7795	0.6284	0.7748	0.6551
	Avg. of top 3 baselines	—	0.7886	0.6342	0.7817	0.6607
	MPR(*pointwise*)	$\lambda_w = 10^2, \lambda_u = 10^2, \lambda_v = 10^3$	0.7876	0.6350	0.7826	0.6615
	MPR(*pairwise*)	$\lambda_w = 10^3, \lambda_u = 10^5, \lambda_v = 10^2$	*0.7896*	**0.6380**	**0.7866**	**0.6647**
	MPR(*large-margin*)	$\lambda_w = 10^4, \lambda_u = 10^4, \lambda_v = 10^3$	0.7881	*0.6374*	*0.7833*	*0.6633*

Boldface indicates the best performance in each group.
Italics indicates the second best performance in each group.



(a) nDCG@10 versus λ_w (b) nDCG@10 versus λ_u (c) nDCG@10 versus λ_i

Fig. 3: Hyper parameters impact in item cold-start scenario

When $\lambda_w = 10^3$, $\lambda_u = 10^1$, and $\lambda_v = 10^2$, MPR can achieve the high performance that is very colose to the optimal result in each scenario. It indicates that three cold-start scenarios can be effectively solved by a single trained model.

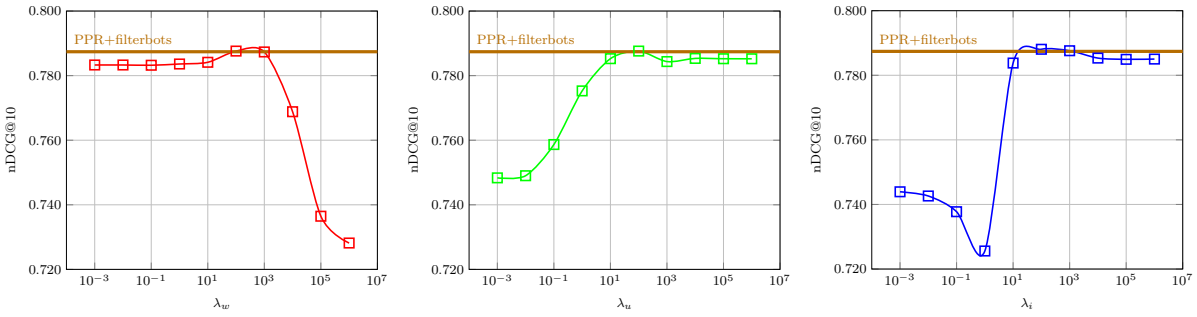
5.7 Discussions

Comparing the result of three different learning strategies, pointwise, pairwise, and large-margin, we can find that the simplest strategy pointwise can get better performance sometimes, for instances, user and item cold-start scenarios of Bookcrossing. Because Bookcrossing is much more sparse than other two datasets as showed in Table 1 and a complex model may get overfitting in Bookcrossing. When more dense data are available, the pairwise and large-margin strategies could obtain improvements, for instances, item cold-start scenarios of Movielens. Because the performances of the three strategies are very competitive, we suggest adopting

Table 4: Results on Movielens-1M dataset

Cold-start scenarios	Algorithms	parameters	nDCG@10	Precision@10	Recall@10	F1@10
User Cold-start (Recommend Existing items to new users)	XGBoost	num-round=800,subsample=0.4 η=0.4,λ=0.1,max-depth=4	0.7094	0.8033	0.2097	0.2937
	VA	—	0.7475	0.8421	0.2230	0.3113
	PPR	$\lambda = 10^3$	0.6315	0.7441	0.1965	0.2744
	PPR+filterbots	$\lambda = 10^5$	0.7524	0.8416	0.2223	0.3103
	MRF-MF	$d = 15, \lambda = 10, k = 200$	*0.7614*	**0.8496**	**0.2243**	**0.3133**
	TKR	$k = 10, \lambda = 1$	0.7512	0.8409	0.2222	0.3102
	LightFM	$d = 10, \alpha = 0$	0.7550	0.8459	0.2233	0.3118
	DecRec	$\tau = 1, n = 1000$	0.6236	0.7359	0.1943	0.2717
	Avg. of top 3 baselines	—	0.7563	0.8459	0.2235	0.3121
	MPR(*pointwise*)	$\lambda_w = 10^4, \lambda_u = 10^2, \lambda_v = 10^2$	**0.7616**	0.8478	*0.2241*	0.3128
	MPR(*pairwise*)	$\lambda_w = 10^4, \lambda_u = 10^2, \lambda_v = 10^2$	0.7597	0.8482	**0.2243**	*0.3131*
	MPR(*large-margin*)	$\lambda_w = 10^4, \lambda_u = 10^2, \lambda_v = 10^4$	0.7549	0.8435	0.2226	0.3109
Item Cold-start (Recommend New items to Existing Users)	XGBoost	num-round=400,subsample=0.2 η=0.2,λ=0.001,max-depth=2	0.7575	0.6720	0.6949	0.6310
	VA	—	0.7056	0.6291	0.6677	0.6004
	PPR	$\lambda = 10^2$	0.7565	0.6710	0.6943	0.6302
	PPR+filterbots	$\lambda = 10^4$	0.7566	0.6709	0.6943	0.6302
	MRF-MF	$d = 15, \lambda = 10, k = 50$	*0.7695*	0.6818	0.7025	0.6389
	TKR	$k = 10, \lambda = 1$	0.7566	0.6704	0.6940	0.6299
	LightFM	$d = 10, \alpha = 0$	0.7679	0.6805	0.7014	0.6378
	DecRec	$\tau = 10, n = 1000$	0.7540	0.6699	0.6929	0.6291
	Avg. of top 3 baselines	—	0.7650	0.6781	0.6996	0.6359
	MPR (*pointwise*)	$\lambda_w = 10^2, \lambda_u = 10^2, \lambda_v = 10^4$	0.7673	0.6805	0.7012	0.6377
	MPR(*pairwise*)	$\lambda_w = 10^4, \lambda_u = 10^2, \lambda_v = 10^4$	**0.7714**	*0.6832*	*0.7032*	*0.6398*
	MPR(*large-margin*)	$\lambda_w = 10^6, \lambda_u = 10^4, \lambda_v = 10^6$	**0.7714**	**0.6841**	**0.7036**	**0.6403**
System Cold-start (Recommend New Items to New Users)	XGBoost	num-round=600,subsample=0.2 η=0.2,λ=1,max-depth=2	0.7565	0.6720	0.6949	0.6310
	VA	—	0.7057	0.6302	0.6691	0.6014
	PPR	$\lambda = 10^3$	0.7557	0.6701	0.6937	0.6294
	PPR+filterbots	$\lambda = 10^4$	0.7553	0.6697	0.6935	0.6292
	MRF-MF	$d = 5, \lambda = 10, k = 50$	0.7553	0.6708	0.6945	0.6304
	TKR	$k = 10, \lambda = 10^{-3}$	0.7545	0.6705	0.6942	0.6299
	LightFM	$d = 10, \alpha = 10^{-4}$	**0.7580**	*0.6720*	**0.6960**	**0.6315**
	DecRec	$\tau = 1, n = 1000$	0.7544	0.6710	0.6943	0.6303
	Avg. of top 3 baselines	—	0.7567	0.6717	0.6951	0.6310
	MPR(*pointwise*)	$\lambda_w = 10^3, \lambda_u = 10^2, \lambda_v = 10^3$	0.7567	**0.6723**	0.6952	*0.6313*
	MPR(*pairwise*)	$\lambda_w = 10^4, \lambda_u = 10^6, \lambda_v = 10^4$	0.7565	0.6719	*0.6953*	0.6311
	MPR(*large-margin*)	$\lambda_w = 10^6, \lambda_u = 10^6, \lambda_v = 10^6$	*0.7568*	0.6715	0.6950	0.6307

Boldface indicates the best performance in each group.
Italics indicates the second best performance in each group.



(a) nDCG@10 versus λ_w (b) nDCG@10 versus λ_u (c) nDCG@10 versus λ_i

Fig. 4: Hyper parameters impact in system cold-start scenario

pointwise learning for its simplicity and high performance, before trying other strategies.

Comparing the parameter impacts in the three cold-start scenarios, we can see that 1) when user's(item's) historical ratings are not available, enlarging the corresponding regularization coefficient $\lambda_u(\lambda_v)$ will improve the performance; 2) when user's (item's) historical ratings are available, the corresponding regularization coefficient $\lambda_u(\lambda_v)$ should be carefully tunned to avoid overfitting and underfitting. Because if a kind of data is only availiale in trainging phase and not availabe in testing phase, a large regulization coefficient should be given to avoid overfitting. If user's and item's historical ratings are very sparse, it is easy to get overfitted and regularization coefficent is sensitive. However, the attributes are shared among all users and items, λ_w is not sensive as showed in Fig. 2(a), 3(a) , and 4(a).

6 Conclusion and Future Work

In this work, we have analyzed the difference of three cold-start scenarios, and proposed a multi-level preference regression framework MPR. With different regression strategies, we implemented three recommendation algorithms: MPR (pointwise), MPR (pairwise), and MPR (large-margin). Comprehensive experiments were conducted on three recommendation datasets to evaluate the proposed framework. The results demonstrated that MPR could archive the state of the art or the best performance in each cold-start scenario. Compared with existing factorization based models, MPR is a linear regression based model, and consequencely, is convex and holds the analytical solution. MPR is more adaptable, when compared to the linear model PPR.

As we know, users or items will become existing ones when they give or obtain some ratings. The users/items with very few ratings should also be considered as new ones to some extent. However, our model treats them differently. In the future, we will study a more general regression that can be adaptive for users/items with few ratings. The attributes are sometimes a little weak to reflect user's preference and item's popularity. User's social information and item's review can also be applied to further improve the performance. A possible way is to convert the social information and reviews to vectors for regression. This will be discussed in the future work.

Acknowledgements This work is supported in part by the National Natural Science Foundation of China (NSFC) (Nos. 61233011 and 61672332), by Jiangsu Natural Science Foundation (No. BK20131351), by Science Foundation for The Excellent Youth Scholars of Ministry of Education of China (No. 61322211), by Program for the Innovative Talents of Higher Learning Institutions of Shanxi (No. 02150116072021), and Shanjin Scholars Program.

References

1. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: an open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM conference on Computer supported cooperative work, ACM (1994) 175–186
2. Park, D.H., Kim, H.K., Choi, I.Y., Kim, J.K.: A literature review and classification of recommender systems research. Expert Systems with Applications **39**(11) (2012) 10059–10072
3. Ricci, F., Rokach, L., Shapira, B.: Recommender systems: introduction and challenges. In: Recommender Systems Handbook. Springer (2015) 1–34
4. Lu, J., Wu, D., Mao, M., Wang, W., Zhang, G.: Recommender system application developments: a survey. Decision Support Systems **74** (2015) 12–32
5. Park, Y., Park, S., Jung, W., Lee, S.g.: Reversed cf: A fast collaborative filtering algorithm using a k-nearest neighbor graph. Expert Systems with Applications **42**(8) (2015) 4022–4028
6. Rao, N., Yu, H.F., Ravikumar, P.K., Dhillon, I.S.: Collaborative filtering with graph information: Consistency and scalable methods. In: Advances in Neural Information Processing Systems. (2015) 2107–2115
7. Lee, J., Sun, M., Lebanon, G.: A comparative study of collaborative filtering algorithms. arXiv preprint arXiv:1205.3193 (2012)
8. Park, S.T., Chu, W.: Pairwise preference regression for cold-start recommendation. In: Proceedings of the third ACM conference on Recommender systems, ACM (2009) 21–28
9. Gantner, Z., Drumond, L., Freudenthaler, C., Rendle, S., Schmidt-Thieme, L.: Learning attribute-to-feature mappings for cold-start recommendations. In: 2010 IEEE 10th International Conference on Data Mining (ICDM). (2010) 176–185
10. Trevisiol, M., Aiello, L.M., Schifanella, R., Jaimes, A.: Cold-start news recommendation with domain-dependent browse graph. In: Proceedings of the 8th ACM Conference on Recommender Systems. RecSys '14, ACM (2014) 81–88
11. Ji, K., Shen, H.: Addressing cold-start: Scalable recommendation with tags and keywords. Knowledge-Based Systems **83** (2015) 42–50
12. Wang, Z., Liang, J., Li, R., Qian, Y.: An approach to cold-start link prediction: Establishing connections between non-topological and topological information. IEEE Transactions on Knowledge and Data Engineering **28**(11) (2016) 2857–2870
13. Zhang, X., Cheng, J., Qiu, S., Zhu, G., Lu, H.: DualDS: A dual discriminative rating elicitation framework for cold start recommendation. Knowledge-Based Systems **73** (2015) 161–172
14. Houlsby, N., Hernandez-lobato, J.M., Ghahramani, Z.: Cold-start active learning with robust ordinal matrix factorization. In: Proceedings of the 31st International Conference on Machine Learning. (2014) 766–774
15. Sun, M., Li, F., Lee, J., Zhou, K., Lebanon, G., Zha, H.: Learning multiple-question decision trees for cold-start recommendation. In: Proceedings of the sixth ACM international conference on Web search and data mining, ACM (2013) 445–454
16. Elahi, M., Ricci, F., Rubens, N.: A survey of active learning in collaborative filtering recommender systems. Computer Science Review (2016)
17. Yan, M., Sang, J., Xu, C., Hossain, M.S.: A unified video recommendation by cross-network user modeling. ACM Transactions on Multimedia Computing, Communications, and Applications **12**(4) (2016) 53
18. Sedhain, S., Sanner, S., Braziunas, D., Xie, L., Christensen, J.: Social collaborative filtering for cold-start recommendations. In: Proceedings of the 8th ACM Conference on Recommender systems, ACM (2014) 345–348
19. Qian, X., Feng, H., Zhao, G., Mei, T.: Personalized recommendation combining user interest and social circle. IEEE transactions on knowledge and data engineering **26**(7) (2014) 1763–1777
20. Al-Shamri, M.Y.H.: User profiling approaches for demographic recommender systems. Knowledge-Based Systems **100** (2016) 175–187
21. Pereira, A.L.V., Hruschka, E.R.: Simultaneous co-clustering and learning to address the cold start problem in recommender systems. Knowledge-Based Systems **82** (2015) 11–19

22. Ocepek, U., Rugelj, J., Bosnić, Z.: Improving matrix factorization recommendations for examples in cold start. Expert Systems with Applications **42**(19) (2015) 6784–6794
23. Zhao, W.X., Li, S., He, Y., Wang, L., Wen, J.R., Li, X.: Exploring demographic information in social media for product recommendation. Knowledge and Information Systems (2015) 1–29
24. Wang, H., Wang, N., Yeung, D.Y.: Collaborative deep learning for recommender systems. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM (2015) 1235–1244
25. Van den Oord, A., Dieleman, S., Schrauwen, B.: Deep content-based music recommendation. In: Advances in Neural Information Processing Systems. (2013) 2643–2651
26. Li, L., Li, T.: News recommendation via hypergraph learning: encapsulation of user behavior and news content. In: Proceedings of the sixth ACM international conference on Web search and data mining, ACM (2013) 305–314
27. Lu, K., Zhang, Y., Zhang, L., Wang, S.: Exploiting user and business attributes for personalized business recommendation. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM (2015) 891–894
28. Barjasteh, I., Forsati, R., Masrour, F., Esfahanian, A.H., Radha, H.: Cold-start item and user recommendation with decoupled completion and transduction. In: Proceedings of the 9th ACM Conference on Recommender Systems, ACM (2015) 91–98
29. Agarwal, D., Chen, B.C.: Regression-based Latent Factor Models. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, ACM (2009) 19–28
30. Zhang, L., Agarwal, D., Chen, B.C.: Generalizing Matrix Factorization Through Flexible Regression Priors. In: Proceedings of the Fifth ACM Conference on Recommender Systems, New York, NY, USA, ACM (2011) 13–20
31. Peng, F., Lu, J., Wang, Y., Yi-Da Xu, R., Ma, C., Yang, J.: N-dimensional markov random field prior for cold-start recommendation. Neurocomputing **191** (2016) 187–199
32. Saveski, M., Mantrach, A.: Item cold-start recommendations: learning local collective embeddings. In: Proceedings of the 8th ACM Conference on Recommender systems. (2014) 89–96
33. Herbrich, R., Graepel, T., Obermayer, K.: Large margin rank boundaries for ordinal regression. Advances in neural information processing systems (1999) 115–132
34. Lee, C.P., Lin, C.J.: Large-scale linear ranksvm. Neural computation **26**(4) (2014) 781–817
35. Bokde, D., Girase, S., Mukhopadhyay, D.: Matrix factorization model in collaborative filtering algorithms: A survey. Procedia Computer Science **49** (2015) 136–146
36. Bobadilla, J., Ortega, F., Hernando, A., Gutiérrez, A.: Recommender systems survey. Knowledge-Based Systems **46** (2013) 109–132
37. Barjasteh, I., Forsati, R., Ross, D., Esfahanian, A.H., Radha, H.: Cold-start recommendation with provable guarantees: A decoupled approach. IEEE Transactions on Knowledge and Data Engineering **28**(6) (2016) 1462–1474
38. Nag, B., Solutions, S.D.: Vibes: A platform-centric approach to building recommender systems. IEEE Data Eng. Bull. **31**(2) (2008) 23–31
39. Park, S.T., Pennock, D., Madani, O., Good, N., DeCoste, D.: Naïve filterbots for robust cold-start recommendations. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM (2006) 699–705
40. Kula, M.: Metadata embeddings for user and item cold-start recommendations. arXiv preprint arXiv:1507.08439 (2015)
41. Cai, J.F., Candès, E.J., Shen, Z.: A singular value thresholding algorithm for matrix completion. SIAM Journal on Optimization **20**(4) (2010) 1956–1982
42. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. arXiv preprint arXiv:1603.02754 (2016)