CPL: A Combined Framework of Pointwise Prediction and Learning to Rank for top-N Recommendations with Implicit Feedback

Nengjun Zhu and Jian Cao(✉)

Shanghai Institute for Advanced Communication and Data Science,
Department of Computer Science and Engineering, Shanghai Jiao Tong University,
Shanghai, China
{zhu_nj,cao-jian}@sjtu.edu.cn

Abstract. Pointwise prediction and Learning to Rank (L2R) are both widely used in recommender systems. Currently, these two types of approaches are often considered independently, and most existing efforts utilize them separately. Unfortunately, pointwise prediction tends to overfit the training data while L2R is more prone to higher variance, and both of them suffer one-class problems using implicit feedback. Therefore, we propose a new framework called CPL, where pointwise prediction and L2R are inherently combined to discriminate user preferences on unobserved items, to improve the performance of top-N recommendations. To verify the effectiveness of CPL, an instantiation of CPL, which is named CPLmg, is introduced. CPLmg is based on two components, i.e., FSLIM (Factorized Sparse LInear Method) and GAPfm (Graded Average Precision factor model), to perform pointwise prediction and L2R, respectively. The low-rank users' and item's latent factor matrices act as a bridge between FSLIM and GAPfm. Moreover, FSLIM dynamically rates an unobserved item for a user based on its similarity with observed items. These pseudo ratings are further utilized with a confidence score to rank items in GAPfm. Extensive experiments on two datasets show that CPLmg significantly outperforms the baselines.

Keywords: Recommender system · Implicit feedback · Collaborative filtering · Learning to Rank · Metrics optimization

1 Introduction

Recommender systems (RSs) have been widely adopted by many online services, since they are able to solve the information overload problem as well as facilitate interaction between users and systems. Most RSs infer users' interests through users' historical behaviors, either represented in explicit form or implicit form. Explicit feedback such as rating, which is given by users, can indicate a

R. Cheng et al. (Eds.): WISE 2019, LNCS 11881, pp. 259–273, 2019.
https://doi.org/10.1007/978-3-030-34223-4_17

user's interest in a particular product. However, this explicit feedback is not always available in practical systems. On the contrary, implicit feedback, such as users' browsing history, or even mouse movements, can be easily obtained from the system and do not burden the users. This information can also reflect users' preferences, although in an indirect way. Consequently, recommendation approaches based on implicit feedback are becoming more widely used [1].

Pointwise prediction and learning to rank (L2R) are two representative genres of the approaches for RSs. Pointwise prediction tries to estimate the value of an item to a user based on historical data with the aim of minimizing prediction errors. It is straightforward and effective when users' historical data is organized in rating forms. In domains where only implicit feedback is available, there are also two definitional levels, i.e., 1 for observed examples and 0 for missing ones, which can reflect the connection strength between a user and an item to some degree [2]. However, pointwise prediction models easily lead to large bias, i.e., overfitting of training data, since they are confined to being finely tuned to each value of individual examples, even these examples are noises. On the other hand, L2R methods explore the preferential relations among multiple items, i.e., the relation that a user prefers item i over item j, and consider the entire ranking list as a target for optimization. In contrast to pointwise prediction, L2R methods may cause high variances since they are not sensitive to small changes in the estimated value of each individual examples unless these examples are compared to the other ones. To balance variance and bias, existing approaches usually add regularization terms to target functions.

In this paper, we explore a new framework, CPL (a Combined framework of Pointwise prediction and L2R), which tries to balance bias and variance not only by regularization but also based on the inherent features of pointwise prediction and L2R. The ultimate goal of CPL is to find a balance between predicting an accurate value for each example (which is the goal of pointwise methods) and keeping the correct preferential relations between items (which is the goal of L2R methods). To verify the effectiveness of CPL, we choose SLIM [9] which is one of the pointwise prediction methods, and GAPfm [14] which is one of the L2R methods, as the two components to implement CPL. SLIM and GAPfm both have been demonstrated to have a stronger performance than other state-of-the-art approaches to top-N recommendations. SLIM utilizes the intuition of item-based K-nearest neighborhood (ItemKNN) collaborative filtering and makes use of the learning process of matrix factorization (MF) techniques to estimate the coefficients between every two items. The estimated coefficients are analogous to item similarities in the traditional ItemKNN method, but they are learned from observed data instead of being calculated based on items' attribute vectors. GAPfm, which addresses the top-N recommendation problem in domains with grade relevance data, takes the Graded Average Precision (GAP) metric as the extreme optimization objective function. However, we would like to utilize the highly discriminative trait of GAP to dynamically mine potential positive examples and to avoid the trap of suppression of preferences for items about which the user is unaware [13,17].

To combine SLIM and GAPfm in a better way, we first revise their original versions. Then, we combine the new versions into CPLmg, which is an implementation of CPL. Specifically, we improve the SLIM to be a more general factor model, namely FSLIM. FSLIM inherits all the desirable characteristics of SLIM and the difference is that FSLIM constructs a dense representation both for users and items, which can improve the recommendation performance [6]. Then, the low-rank users' and item's latent factor matrices act as a bridge between FSLIM and GAPfm, so that the learned dense representation can be transferred to each other. Moreover, FSLIM dynamically rates unobserved items for a user based on the learned item similarities. These pseudo ratings are further utilized in GAPfm, and the confidence score of a pseudo rating to be a threshold, which separates unknown items to a positive example or to a negative example, is also updated dynamically in every training round. Thus, the combination of FSLIM and GAPfm results in the considerably improved learning accuracy of GPLmg.

The main contributions of this paper are as follows: (1) We introduce a new framework CPL to combine pointwise prediction and L2R methods to address the top-N recommendation problem. (2) We propose an implementation of CPLmg for CPL. In CPLmg, we combine the FSLIM and GAPfm models. FSLIM is extended from SLIM. Moreover, strategies are designed to better integrate FSLIM and GAPfm. (3) Extensive experiments, which show that CPLmg outperforms other baselines on various evaluation metrics, are conducted.

2 Related Work

Our proposed model, which is based on a combination of pointwise prediction and learning to ranking, addresses the top-N recommendation problem with implicit feedback. Therefore, it is related to state-of-the-art top-N recommendation technologies, including matrix factorization (MF) methods and L2R approaches.

MF is one of the most popular model-based collaborative filtering (CF) methods. It learns latent factor representations with respect to users and items, and models user preferences as the dot product of latent factor vectors. SLIM [9] is a particular case of MF. It directly learns a similarity matrix from the data and thus becomes a novel learning model. To address the quadratic computation problem of SLIM, a factorized similarity model FISM [5] is proposed. FISM factorizes the similarity matrix into two low-rank matrices. However, both SLIM and FISM do not produce a user-specific latent factor matrix. Thus, LRec [13], which is interpreted as a linear classification model for each user, is proposed to overcome this limitation. Currently, some work has explored the combination of MF and deep learning for recommendations. For instance, NeuMF [4] is a neural network-based CF method, and it is essentially a fusion of generalized matrix factorization and multi-layer perceptron.

L2R becomes a hot research area, since it directly models partial ordering relations between items, which happens to be consistent with top-N recommendation tasks. One key element of L2R methods is the objective measures, defined as either ranking error functions or optimization metrics. Thus, based on different objective measures, many L2R methods have been proposed. BPR [11]

maximizes AUC metrics by utilizing the partial order relations between items. xCLiMF [15] is an L2R method based on expected reciprocal rank (ERR). Moreover, TFMAP [16] optimizes MAP metric directly. To alleviate the overfitting problem of L2R, GTRM [17] optimizes the group-oriented mean average precision (GMAP) which considers the similarities between items, and PRIGP [10] integrates item-based pairwise preferences and item group-based pairwise preferences into the framework based on BPR-OPT derived from BPR.

However, the above-mentioned approaches only utilize regularization terms to balance bias and variance, and none of them combine pointwise prediction and L2R for top-N recommendations.

3 Preliminaries

3.1 Definitions and Notations

Assume that the implicit feedback data is from M users' behaviors on N items, and we use the symbol u to index a user, the symbol i and j to index items, and the symbol k to index a latent factor. The set of all users and items are represented by $\mathcal{U} = \{1, 2, \ldots, u, \ldots, M\}$ and $\mathcal{I} = \{1, 2, \ldots, i, \ldots, N\}$, respectively. The matrices $\mathbf{P} \in \mathbb{R}^{M \times K}$ and $\mathbf{Q} \in \mathbb{R}^{N \times K}$ are latent factor matrices related to users and items, respectively. The entire set of users' historical feedback such as purchases/clicking are represented by a user-item interaction matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$, in which each entry is represented by $\mathbf{A}_{ui} \in \{0, 1\}$, where $\mathbf{A}_{ui} = 1$ means user u has at some point interacted with item i (observed items), otherwise the entry is marked as 0 (unobserved items).

In the rest of the paper, vectors and matrices are both denoted by upper bold symbols, where the symbol with no subscript represents the matrix itself. The symbol with one subscript (e.g., $\mathbf{P}_u$) represents a vector extracted from its matrix by the row/column subscript index, and the symbol with two subscripts (e.g., $\mathbf{P}_{uk}$) represents the entry. A predicted value is denoted by the symbol with a wide tilde head (e.g., $\widetilde{\mathbf{A}}_{ui}$). Unless stated differently, all vectors are column vectors by default, but the vectors with the transposed subscript $^\top$ are row vectors (e.g., $\mathbf{P}_u^\top$ denotes the u-th row of $\mathbf{P}$).

3.2 SLIM

A parse linear method SLIM [9] has demonstrated very good performance for top-N recommendations. Different from traditional similarity models that calculate similarities based on attributes according to certain criteria, SLIM learns the item similarities from the data directly. That is, SLIM estimates a sparse aggregation coefficient matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$, in which each entry $\mathbf{W}_{ij}$ can be viewed as the similarity between items i and j. Then the recommendation score from user u to an unobserved item i is computed as a sparse aggregation of all the observed items of the user, as follows:

$$\widetilde{\mathbf{A}}_{ui} = \mathbf{A}_u^\top \mathbf{W}_i \tag{1}$$

where $\mathbf{A}_u^\top$ is the row vector extracted from $\mathbf{A}$ by the row/user index u, and $\mathbf{W}_i$ is a column vector, which represents the i-th column vector of matrix $\mathbf{W}$. Then, SLIM estimates/learns the $\mathbf{W}$ by solving the following optimization problem:

$$\begin{aligned} \underset{\mathbf{W}}{\text{minimize}} \quad & \frac{1}{2}||\mathbf{A} - \mathbf{AW}||_F^2 + \frac{\beta}{2}||\mathbf{W}||_F^2 + \lambda||\mathbf{W}||_1 \\ \text{subject to} \quad & \mathbf{W} \geq 0 \\ & \text{diag}(\mathbf{W}) = 0 \end{aligned} \tag{2}$$

Here, $||\mathbf{W}||_1$ is the entry-wise ℓ_1-norm of $\mathbf{W}$ which encourages sparsity, and $\|\bullet\|_F$ is the matrix Frobenius norm. The constraint $\text{diag}(\mathbf{W}) = 0$ prevents learned item similarities from being affected by the item itself. As for the nonnegativity constraint, [7] showed that it could be ignored without affecting performance.

3.3 GAPfm

GAPfm [14] is a listwise L2R method which directly optimizes a smoothed approximation of GAP metric [12]. GAP generalizes average precision (AP) to the case of multi-graded relevance, and inherits the most important properties of AP metric to guarantee that mistakes in recommended items at the top of the list carry a higher penalty than mistakes at the bottom of the list. The definition of GAP is as follows:

$$\begin{aligned} GAP_u = \frac{1}{Z_u} \sum_{i=1}^{N} \frac{I_{ui}}{R_{ui}} \sum_{j=1}^{N} I_{uj} \mathbb{I}(R_{uj} \leq R_{ui}) \\ (\mathbb{I}(y_{ui} < y_{uj}) \sum_{l=1}^{y_{ui}} \delta_l + \mathbb{I}(y_{uj} \leq y_{ui}) \sum_{l=1}^{y_{uj}} \delta_l) \end{aligned} \tag{3}$$

where R_{ui} denotes the ranked position of item i for user u, e.g., $R_{ui} = 1$ denotes the item is ranked in the first/highest position. $I_{ui} = 1$ ($I_{ui} \in \{0, 1\}$) denotes the item is a positive example, otherwise it is a negative/missing example. y_{ui} denotes the grade of item i to user u. $\mathbb{I}(x)$ is a binary indicator function, i.e., it is equal to 1 if x is true, otherwise 0. $Z_u = \sum_{l=1}^{y_{max}} n_{ul} \sum_{c=1}^{l} \delta_c$ is a constant normalizing coefficient for user u, where n_{ul} denotes the number of items rated with grade l by user u, and δ_l denotes the thresholding probability that the user sets as a threshold of relevance at grade l, i.e., regarding items with grades equal or larger than l as relevant ones, and the others as irrelevant ones.

$$\delta_l = \begin{cases} \dfrac{2^l - 1}{2^{y_{max}}}, & y_{max} > 1 \\ 1, & y_{max} = 1 \end{cases} \tag{4}$$

where $[1, y_{max}]$ is the scale of ratings. Then, with a small manipulation, Eq. (3) can be smoothed to be an optimization objective function with respect to the learned parameters, i.e., $\mathbf{P}$ and $\mathbf{Q}$, the details are given in the following sections.

4 Proposed Methodology

In this section, we introduce two components of CPL: (1) Factorized SLIM (FSLIM), and (2) GAPfm with sampling strategy. Then, we show in detail how to combine FSLIM and GAPfm to implement CPL.

4.1 Factorized SLIM (FSLIM)

Our proposed Factorized SLIM is a new version of SLIM that incorporates ideas from traditional matrix factorization (MF) methods and similarity approaches. We still define the recommendation score from user u to an unobserved item i as a sparse aggregation of the scores of all observed items by the user. However, the score of each item is no longer a defined value, i.e., 1 and 0, but is calculated as the dot product of the item's latent factor vector and the user's latent factor vector, as shown in Eq. (5). The dense representations of users and items introduce more information capabilities.

$$\widetilde{\mathbf{A}}_{ui} = \mathbf{P}_u^\top \sum_{j \in \mathcal{N}(i) \cap \mathcal{O}(u)} \mathbf{Q}_j \mathbf{W}_{ji} \tag{5}$$

where $\mathbf{W} \in \mathbb{R}^{N \times N}$ is a sparse aggregation coefficient matrix such as that in SLIM, and $\mathcal{O}(u)$ is the set of all observed items of user u. To speed up the learning process, $\mathcal{N}(i)$ representing the set of near neighborhoods of item i, is added to select items in $\mathcal{O}(u)$. This operation can be viewed as feature selection [9]. We utilize the cosine similarity, which is calculated based on co-click/co-visitation behaviors to items by users, to retrieve $iknn$ near neighborhoods of item i, i.e., $|\mathcal{N}(i)| = iknn$. Finally, taking into account all users, the loss function is defined as follows:

$$\begin{aligned} \mathcal{L}_F = & \frac{1}{2} \sum_{u=1}^{M} \sum_{i=1}^{N} \|\mathbf{A}_{ui} - g(\mathbf{P}_u^\top \sum_{j \in \mathcal{N}(i) \cap \mathcal{O}(u)} \mathbf{Q}_j \mathbf{W}_{ij})\|_F^2 \\ & + \frac{\beta_1}{2} \|\mathbf{P}\|_F^2 + \frac{\beta_2}{2} \|\mathbf{Q}\|\|_F^2 + \frac{\beta_3}{2} \|\mathbf{W}\|_F^2 + \lambda \|\mathbf{W}\|_1 \end{aligned} \tag{6}$$

where $g(x) = 1/(1+e^{x)})$ is a sigmoid function, which is a common choice for one-class recommendation. We add the constraint $\text{diag}(\mathbf{W}) = 0$ to prevent learned item similarities from being affected by the item itself, and drop the nonnegativity constraint, i.e., $\mathbf{W} \geq 0$, compared to SLIM as the reason we aforementioned.

Stochastic gradient decent technology (SGD) is used to solve this optimization problem, and the gradients of the parameters are listed as follows:

$$\frac{\partial \mathcal{L}_F}{\partial \mathbf{P}_u} = -(\mathbf{A}_{ui} - g(\widetilde{\mathbf{A}}_{ui}))g'(\widetilde{\mathbf{A}}_{ui}) \sum_{j \in \mathcal{N}(i) \cap \mathcal{O}(u)} \mathbf{Q}_j \mathbf{W}_{ji} + \beta_1 \mathbf{P}_u \tag{7}$$

$$\frac{\partial \mathcal{L}_F}{\partial \mathbf{Q}_i} = - \sum_{j \in \mathcal{N}(i) \cap \mathcal{O}(u)} (\mathbf{A}_{uj} - g(\widetilde{\mathbf{A}}_{uj}))g'(\widetilde{\mathbf{A}}_{uj}) \mathbf{P}_u \mathbf{W}_{ij} + \beta_2 \mathbf{Q}_i \tag{8}$$

$$\frac{\partial \mathcal{L}_F}{\partial \mathbf{W}_{ij}} = -(\mathbf{A}_{ui} - g(\widetilde{\mathbf{A}}_{ui}))g'(\widetilde{\mathbf{A}}_{ui})\mathbf{P}_u^\top \mathbf{Q}_j + \beta_3 \mathbf{W}_{ij} \pm \lambda \tag{9}$$

where $g'(x) = g(x)/(1 - g(x))$ is the derivative of function $g(x)$. Then, with a learning step size η_1, the parameters are updated using SGD.

4.2 GAPfm with Sampling Strategy

The work about GAPfm in [14] mainly focuses on graded relevance domains, such as rating data, and takes GAP as the objective metric in learning to rank. However, in domains with binary relevance data, we would still like to take full advantage of high informativeness and discriminative power of GAP to dynamically mine potential preferred items and to avoid the trap of the suppression of preferences for items about which the user is unaware. That is, we utilize the sparse aggregation coefficient matrix (the item similarity matrix) learned from FSLIM to estimate the pseudo rating of each item for each user, which can be demonstrated as:

$$y_{ui} = \begin{cases} g\Big(\mathbf{P}_u^\top \sum\limits_{j \in \mathcal{N}(i) \cap \mathcal{O}(u)} \mathbf{Q}_j \mathbf{W}_{ji}\Big), & i \notin \mathcal{O}(u) \\ 1, & i \in \mathcal{O}(u) \end{cases} \tag{10}$$

It is likely that some values will be prefill into matrix $\mathbf{A}$. The closer the value of y_{ui} to 1, the more likely item i is a potential preferred item for user u, since the value of y_{ui} depicts the relationship between item i and the user. Thus, according to the value of y_{ui}, we select the top pn unobserved items as potential preferred items for user u, and record the indexes of all these items and already observed items into a set $\mathcal{O}'(u)$ as well as record the pseudo ratings (the values of y_{ui}) of all items in $\mathcal{O}'(u)$ into a set $Y(u)$. Then, we change the thresholding probability $\delta_u(y)$ as a confidence score of that pseudo rating $y \in Y(u)$ being the threshold value for user u, i.e., regarding items with a pseudo rating equal or larger than y as potential preferred ones, and the others as not preferred ones, as follows:

$$\delta_u(y) = \frac{exp(y)}{\sum_{t \in Y(u)} exp(t)} \tag{11}$$

The larger the value of $\delta_u(y)$ or of y, the more credible the result of this division. Then, we update the formulation of GAP in Eq. (3) as follows:

$$GAP_u = \frac{1}{Z_u} \sum_{i=1}^{N} \frac{I_{ui}}{R_{ui}} \sum_{j=1}^{N} S_{uij} I_{uj} \mathbb{I}(R_{uj} \leq R_{ui}) \tag{12}$$

$$Z_u = \sum_{t \in Y(u)} n_{ut} \sum_{l \in Y(u) \& l \leq t} \delta_u(l)$$

$$S_{uij} = \mathbb{I}(y_{ui} < y_{uj}) \sum_{t \in Y(u) \& t \leq y_{ui}} \delta_u(t) + \mathbb{I}(y_{uj} \leq y_{ui}) \sum_{t \in Y(u) \& t \leq y_{uj}} \delta_u(t)$$

where R_{ui} denotes the ranked position of item i for user u, I_{ui} indicates whether the index of the item is in $\mathcal{O}'(u)$, and $\mathbb{I}(x)$ is a binary indicator function, such as those in GAPfm. Z_u is a constant normalizing coefficient for user u, where n_{ut} denotes the number of items rated with pseudo rating t to user u, S_{uij} is a intermediate variable whose value is related to the sorted list.

Then we use $g(x)$ function and parameters $\mathbf{P}, \mathbf{Q}$ to estimate the term of $\frac{1}{R_{ui}} \approx g(f_{ui})$ and $\mathbb{I}(R_{uj} \leq R_{ui}) \approx g(f_{uj} - f_{ui})$, where $f_{ui} = \mathbf{P}_u^\top \mathbf{Q}_i$, in Eq. (12) to get a smoothed version of GAP as follows:

$$GAP_u \approx \frac{1}{Z_u} \sum_{i=1}^{N} I_{ui} g(f_{ui}) \sum_{j=1}^{N} S_{uij} I_{uj} g(f_{uj} - f_{ui}) \tag{13}$$

Then, taking into account all users and adding two Frobenius norms $\|\mathbf{P}\|_F$ and $\|\mathbf{Q}\|_F$ as well as parameters β_4 and β_5 to control the magnitude of regularization, the final objective function of GAPfm is shown as follows:

$$\mathcal{L}_G = \sum_{u=1}^{M} \sum_{i=1}^{N} I_{ui} g(f_{ui}) \sum_{j=1}^{N} S_{uij} I_{uj} g(f_{u(j-i)}) - \frac{\beta_4}{2} \|\mathbf{P}\|_F^2 - \frac{\beta_5}{2} \|\mathbf{Q}\|_F^2 \tag{14}$$

Note that, Eq. (14) has dropped the coefficient $1/M$ and $1/Z_u$ since they are independent of the latent factors and have no influence on the optimization procedure. Now, we use the stochastic gradient ascent (SGA) to solve this optimization problem, and the gradients of the parameters are as follows:

$$\begin{aligned} \frac{\partial \mathcal{L}_G}{\partial \mathbf{P}_u} = \sum_{i=1}^{N} I_{ui} \Big(& g'(f_{ui}) \sum_{j=1}^{N} I_{uj} S_{uij} g(f_{u(j-i)}) \cdot \mathbf{Q}_i \\ & + g(f_{ui}) \sum_{j=1}^{N} I_{uj} S_{uij} g'(f_{u(j-i)}) \cdot (\mathbf{Q}_j - \mathbf{Q}_i) \Big) - \lambda \mathbf{P}_u \end{aligned} \tag{15}$$

$$\begin{aligned} \frac{\partial \mathcal{L}_G}{\partial \mathbf{Q}_i} = I_{ui} \Big(& g'(f_{ui}) \sum_{j=1}^{N} I_{uj} S_{uij} g(f_{u(j-i)}) + \sum_{j=1}^{N} I_{uj} \\ & [S_{uji} g(f_{uj}) - S_{uij} g(f_{ui})] g'(f_{u(j-i)}) \Big) \mathbf{P}_u - \lambda \mathbf{Q}_i \end{aligned} \tag{16}$$

Then, we update the parameters in GAPfm using SGA with a learning rate η_2.

4.3 CPLmg Recommendation Model

Now, we introduce how to combine FSLIM and GAPfm under MF framework to implement CPLmg, so that they can mutually reinforce each other and can better learn from complex user-item interactions. We propose to train FSLIM and GAPfm using a multi-task learning approach [8] where the latent factor

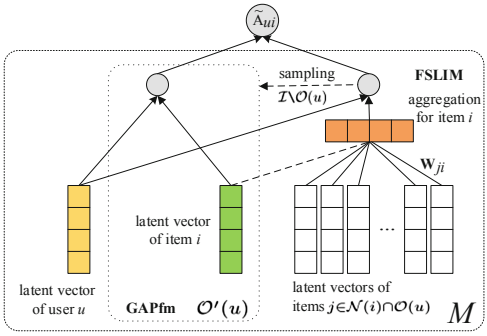


Fig. 1. The framework of CPLmg

matrices $\mathbf{P}$ and $\mathbf{Q}$ are shared underlying variables, as shown in Fig. 1. In particular, the matrices $\mathbf{P}$ and $\mathbf{Q}$ are jointly updated by FSLIM and GAPfm in each co-training round by modeling the task of pointwise methods and the task of L2R methods. The additional item similarity matrix $\mathbf{W}$ further helps GAPfm mine potential preferred items, which allows information transfer between two tasks [3]. Furthermore, the trade-off controlling parameters in CPLmg are the learning rate parameters, i.e., η_1 and η_2, since the relationship between the values of η_1 and η_2 determines the impact of each component on the model learning process. CPLmg is trained until both FSLIM and GAPfm are converged or until reaching the maximal number of iteration.

4.4 Time Complexity

The time complexity of CPLmg comprises two parts which accumulate linearly, i.e., the time cost of FSLIM and GAPfm. Thus, CPLmg finally takes $O(M|\bar{I}|^3K + M|\bar{J}|(|\bar{I}| + \ln(|\bar{J}|))$ time to update the parameters in each iteration.

4.5 Recommendation

At the prediction phase, we measure the final preference score of unobserved items to each user as follows:

$$\widetilde{\mathbf{A}}_{ui} = \mathbf{P}_u^\top \sum_{j \in \mathcal{N}(i) \cap \mathcal{O}(u) \cup \{i\}} \mathbf{Q}_j \mathbf{W}'_{ji} \tag{17}$$

where $\mathbf{W}' = \mathbf{W} + w * \mathbf{I}$, and $\mathbf{I} \in \mathbb{R}^{N \times N}$ is an identity matrix, and w is a weight parameter of the combination of prediction functions in FSLIM, i.e., $\widetilde{\mathbf{A}}_{ui} = \mathbf{P}_u^\top \sum_{j \in \mathcal{N}(i) \cap \mathcal{O}(u)} \mathbf{Q}_j \mathbf{W}_{ji}$ and in GAPfm, i.e., $\widetilde{\mathbf{A}}_{ui} = \mathbf{P}_u^\top \mathbf{Q}_i$, respectively. The value of w is related to the trade-off controlling parameters of the framework CPL, and we set the value of the ratio of $\frac{\eta_2}{\eta_1}$ to w. The items from the set $\mathcal{I}\backslash\mathcal{O}(u)$ with the largest prediction values based on Eq. (17) are recommended to the user.

Table 1. The datasets used in evaluation

Dataset	#users	#items	#trans	Density	Ratings	Threshold[a]
AppData	20,467	40,259	1,022,339 (installations)	0.124%	–	–
ML100K	943	1,682	100,000 (ratings)	6.30%	1–5	1

[a]The user-item pairs with ratings equal or greater than *threshold* are positive examples, the others are the missing ones.

So far, we have implemented CPLmg based on two components: FSLIM and GAPfm. A question then arises: why does the combination of these methods promote the top-N performance under CPL framework. There are four possible reasons: (1) The learning directions of FSLIM and GAPfm are generally consistent when learning the model parameters $\mathbf{P}$ and $\mathbf{Q}$ through the gradient approach. Both of them tend to increase the value of a dot product with respect to positive examples. (2) The sampling procedure based on the item similarity matrix $\mathbf{W}$ brings more information from FSLIM to GAPfm to make GAPfm more informative. (3) The multi-task learning approach allows information transfer between the two tasks. (4) CPLmg balances the variance and bias of the model, as previously discussed. Thus, our proposed CPLmg approach can yield better performance for top-N recommendations, which is demonstrated by the following experiments.

5 Experimental Results

5.1 Datasets and Settings

Our experiments are based on two datasets, AppData and MovieLens-100K (ML100K). The characteristics of these two datasets are shown in Table 1.

The dataset AppData is from users' log files where the users' interactive behaviors with mobile applications are recorded for six months. Since we are more concerned about which applications the user will install on their smartphone, we only keep already installed mobile applications for users. Then, each observed user-item pair represents one record of the user installing the application.

The ML100K is a public dataset and it is organized in rating forms. However, since we only discuss the one-class recommendation problem in this paper, the ratings are converted to the appropriate binary form based on the threshold h, i.e., the user-item pairs with ratings higher than h are positive examples, the others are the missing ones. To simplify the analysis, we give the best value of the threshold in our experiments, i.e., $h = 1$. In the experiment, mobile applications and movies are the items to be recommended.

We randomly select records from the users' historical data to keep a certain number of observed items for each user as the test data, and set the rest of the records as the training set. For example, "Given 10" denotes that for each user, we randomly select ten observed items as unknowns in the training set, but as

positive examples in the test set. Then, we measure the performance over these positive examples in the test data.

Precision is a widely used evaluation metric in RSs. It reflects the ratio of relevant items in the ranked list given a truncated position. In the case of top-N RSs, MAP (mean average precision) and MRR (mean reciprocal rank) are more practical, as they are position-related metrics. To better verify the properties of the model, we apply all these three metrics to evaluate the performance of the new and compared methods in this paper.

Table 2. Performance comparison based on the top-5 recommendation items

Method	AppData/Given 3					AppData/Given 10				
	Params[1]		Precision	MRR	MAP	Params		Precision	MRR	MAP
iPOP	–	–	0.0989	0.2874	0.1098	–	–	0.2830	0.6303	0.2264
ItemKNN	50	–	0.1126	0.3164	0.1290	50	–	0.3601	0.6089	0.2971
FISMauc	0.8	$1e^{-5}$	0.1002	0.2895	0.1108	0.9	$1e^{-5}$	0.2989	0.6338	0.2395
GAPfm	0.01	–	0.1186	0.3136	0.1348	0.01	–	0.3862	0.7053	0.3132
SLIM	0.1	0.5	0.1563	0.3948	0.1759	0.1	0.5	0.4017	0.7070	0.3177
FSLIM	0.06	0.14	0.1601	0.4158	0.1801	0.04	0.12	0.4072	0.7164	0.3173
NeuMF	10	–	0.1698	0.4209	0.1894	10	–	0.4098	0.7203	0.3184
CPLmg	245	22	**0.1799**	**0.4377**	**0.2022**	245	22	**0.4208**	**0.7345**	**0.3305**
Method	ML100K/Given 3					ML100K/Given 10				
	Params		Precision	MRR	MAP	Params		Precision	MRR	MAP
iPOP	–	–	0.0417	0.1153	0.0415	–	–	0.1324	0.3001	0.0823
ItemKNN	50	–	0.0697	0.1855	0.0696	50	–	0.1769	0.3885	0.1152
FISMauc	0.7	$5e^{-6}$	0.0491	0.1119	0.0413	0.6	$1e^{-6}$	0.1342	0.2948	0.0808
GAPfm	0.05	–	0.0923	0.2497	0.0987	0.05	–	0.1820	0.3998	0.1475
SLIM	0.2	0.6	0.0982	0.2519	0.1009	0.1	0.5	0.2232	0.4722	0.1537
FSLIM	0.002	0.005	0.1034	0.2590	0.1113	0.001	0.005	0.2398	0.4795	0.1599
NeuMF	8	–	0.1078	0.2601	0.1132	8	–	0.2399	0.4819	0.1614
CPLmg	350	35	**0.1194**	**0.2708**	**0.1298**	350	35	**0.2483**	**0.4916**	**0.1712**

5.2 Experimental Comparisons with Previous Models

We compare our methods CPLmg and FSLIM with six baselines as follows: (1) **iPOP** recommends a certain number of the most popular items from the training set to all users. (2) **ItemKNN** is a traditional item-based collaborative filtering method using Jaccard similarity. (3) **FISMauc** [5] considers ranking errors based on loss function and obtains better performance than FISMrmse, which considers the pointwise squared error loss function. Therefore, we do not further report on the performance of FISMrmse. (4) **NeuMF** [4], which is a state-of-the-art method using neural network-based collaborative filtering (NCF) framework. (5) **SLIM** and (6) **GAPfm** are related to two components of CPLmg, respectively. For each model, the parameters were empirically tuned to their optimal values in the experiments and they were recorded in Table 2, i.e., for ItemKNN, they are the number of neighbors; for FISMauc, they are the user-specific parameter α and the learning rate; for GAPfm, they are the regularization parameters; for SLIM and FSLIM, they are both the ℓ_1-norm and ℓ_2-norm regularization

parameter; for NeuMF, it is the number of negative samples; for CPLmg, they are the number of near neighborhoods $iknn$ and the number of candidate potential positive examples pn.

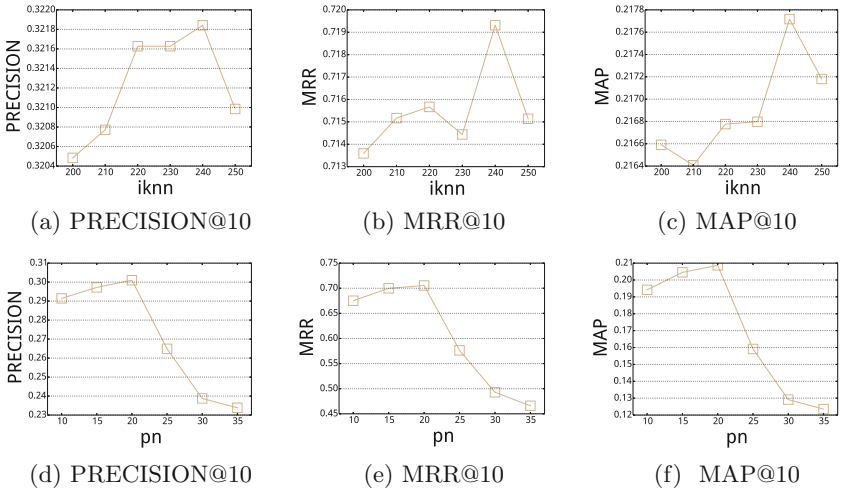
Since the size of the recommendation window is limited in practice, we measure all the performance values in the experiments which are reported in this subsection based on the top-5 recommendation, and the results for top-10 recommendations are shown in the next subsection.

Table 2 shows that CPLmg achieves the best performance than the baselines according to all three metrics. Then, it is NeuMF, which is the state-of-the-art method using implicit feedback. It proves that the proposed CPLmg is highly competitive for top-N recommendation tasks for reasons previously discussed, and also proves that the combination of FSLIM and GAPfm is effective since the performance of FSLIM and GAPfm is not good as NeuMF before the combination. We can also observe that the performance of FSLIM is better than SLIM. This indicates that dense representations of the user and item matrix can better model the users' preferences. The results also show SLIM and GAPfm outperform the remaining methods, i.e., iPOP, ItemKNN, and FISMauc. This observation provides empirical evidence that SLIM and GAPfm approaches are more effective for top-N recommendations. This is one reason why we choose SLIM and GAPfm as the two components in our new framework. Furthermore, it can be noted that the performance of all methods is better when the number of given items increases from 3 to 10. The reason for this lies in the fact that more preferred items in the test data can better reveal the preferences of users and more preferred items in the test means a higher chance of ranking potential preferred items in the top positions.

5.3 Analysis of CPLmg Components

In this section, we describe the experiments conducted to explore the influence of the main parameters on CPLmg, i.e., the number of neighborhoods when conducting feature selection for FSLIM, the size of the candidate potential preferred items in GAPfm sampling process, and the learning rates. It is worth pointing out when we change the settings of one of these parameters, the others are set to their optimal values, e.g., $iknn = 245, pn = 22, \eta_1 = 5 \times 10^{-2}, \eta_2 = 10^{-4}$ for AppData. All experiment results given in this section are under the condition, i.e., "Given 10". Due to the space limitation and without loss of generality, we only report the parameter influences for the top-10 recommendations on AppData. Similar results were observed on ML100K data.

iknn. We first conducted an experiment to investigate the influence of the number of near neighborhoods $iknn$. The results are shown in Fig. 2(a)–(c). We can observe that the values of all three metrics including precision, MRR, and MAP significantly increase at the beginning, then after the turning points, i.e., 240, all values decline. This proves the effectiveness of the feature selection algorithm and it might have an optimal value of $iknn$. The value of $iknn$ is not the bigger the better, since too many similar items may blur the preference information to



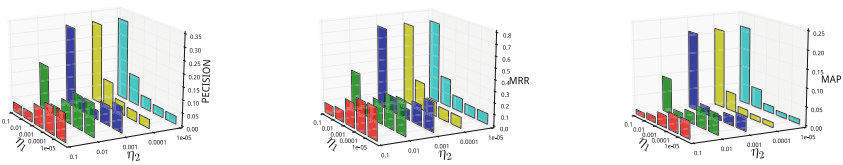
(a) PRECISION@10 (b) MRR@10 (c) MAP@10

(d) PRECISION@10 (e) MRR@10 (f) MAP@10

Fig. 2. Results on different parameters *iknn* and *pn* for top-10 recommendations

be learned for a user. Note that the *iknn* is not the final number of neighborhoods to be considered since $|\mathcal{N}(i) \cap \mathcal{O}(u)| \leq iknn, |\mathcal{O}(u)|$, and usually $|\mathcal{O}(u)|$ might be small in practice, e.g., the average number of installed applications over users in AppData is smaller than 50.

pn. The value of *pn* controls the number of candidate potential preferred items in the sampling process of GAPfm. The influence of *pn* on the recommendation performance is shown in Fig. 2(d)–(f). We can observe that precision, MRR, and MAP performance can be improved by properly increasing *pn*. However, when the increasement is over a turning point, i.e., 20, the performance starts to decline sharply. The reason for this is that a larger value of *pn* also introduces more false preferred items. This observation proves that it is critical to properly take into account missing values within the model in domains with binary implicit feedback, since the selected missing values can alleviate the overfitting risk.



(a) PRECISION@10 (b) MRR@10 (c) MAP@10

Fig. 3. Results on different parameters η_1 and η_2 for top-10 recommendations

η_1 and η_2. In this part, we provide the experiment results based on different values for two parameters η_1 and η_2 which control the learning step sizes of FSLIM and GAPfm, respectively. As previously mentioned, these two parameters also act as a trade-off between the two components of CPLmg, i.e., FSLIM and GAPfm. The influence of η_1 and η_2 is shown in Fig. 3. We observe that all criteria show the same changes on different η_1 and η_2 values. We also observe that some of the performance values are lower than the normal level. This is because η_1 and η_2 will restrain each other in some settings where both η_1 and η_2 try to dominate the learning process, i.e., η_1 and η_2 have very close values. Furthermore, the largest performance values are fastened in the top right corner while the performance values in the bottom left corner also tend to increase. All these results show that the performance values increase with the proper increasement of divergence between these two parameters.

6 Conclusions and Future Work

In this paper, we proposed a new framework, CPL, where pointwise prediction and L2R are inherently combined to discriminate user preferences on unobserved items and to improve the performance of top-N recommendations. Moreover, to verify the effectiveness of CPL, we implement CPLmg which takes FSLIM and GAPfm as its two components, where FSLIM is a variant of SLIM by infusing dense representations. The components reinforce each other through information interchange based on the dense representations and aggregation coefficients. The final experiments prove that CPLmg is effective and outperforms the others on various evaluation metrics. There are some potential research topics for future study. Firstly, the combination approach between two components of CPLmg can be extended. We would like to explore a more complex combination. For instance, we can fuse two components based on the neural network framework motivated by NCF [4]. Secondly but not lastly, other models of pointwise prediction and L2R methods can be tried in the framework.

Acknowledgement. This work is partially supported by National Key Research and Development Plan (No. 2018YFB1003800).

References

1. Anyosa, S.C., Vinagre, J., Jorge, A.M.: Incremental matrix co-factorization for recommender systems with implicit feedback. In: Proceedings of the WWW Conference on World Wide Web, pp. 1413–1418. International World Wide Web Conferences Steering Committee (2018)
2. Ding, J., Guanghui Yu, X.H., Quan, Y.: Improving implicit recommender systems with view data. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 3343–3349. AAAI Press (2018)
3. Dong, D., Zheng, X., Zhang, R., Wang, Y.: Recurrent collaborative filtering for unifying general and sequential recommender. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 3350–3356. AAAI Press (2018)

4. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the WWW Conference on the World Wide Web, pp. 173–182. International World Wide Web Conferences Steering Committee (2017)
5. Kabbur, S., Ning, X., Karypis, G.: Fism: factored item similarity models for top-n recommender systems. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 659–667. ACM (2013)
6. Larraín, S., Parra, D., Soto, A.: Towards improving top-n recommendation by generalization of slim. In: RecSys Posters (2015)
7. Levy, M., Jack, K.: Efficient top-n recommendation by linear regression. In: RecSys Large Scale Recommender Systems Workshop (2013)
8. Ma, J., Zhao, Z., Yi, X., Chen, J., Hong, L., Chi, E.H.: Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1930–1939. ACM (2018)
9. Ning, X., Karypis, G.: Slim: sparse linear methods for top-n recommender systems. In: Proceedings of the International Conference on Data Mining (ICDM), pp. 497–506. IEEE (2011)
10. Qiu, S., Cheng, J., Yuan, T., Leng, C., Lu, H.: Item group based pairwise preference learning for personalized ranking. In: Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1219–1222. ACM (2014)
11. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence, pp. 452–461. AUAI Press (2009)
12. Robertson, S.E., Kanoulas, E., Yilmaz, E.: Extending average precision to graded relevance judgments. In: Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 603–610. ACM (2010)
13. Sedhain, S., Menon, A.K., Sanner, S., Braziunas, D.: On the effectiveness of linear models for one-class collaborative filtering. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 229–235. AAAI Press (2016)
14. Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Hanjalic, A.: GAPFM: optimal top-n recommendations for graded relevance domains. In: Proceedings of the ACM International Conference on Conference on Information and Knowledge Management, pp. 2261–2266. ACM (2013)
15. Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Hanjalic, A.: xCLiMF: optimizing expected reciprocal rank for data with multiple levels of relevance. In: Proceedings of the ACM Conference on Recommender Systems, pp. 431–434. ACM (2013)
16. Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Hanjalic, A., Oliver, N.: TFMAP: optimizing map for top-n context-aware recommendation. In: Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 155–164. ACM (2012)
17. Zhu, N., Cao, J.: GTRM: a top-N recommendation model for smartphone applications. In: Proceedings of the IEEE International Conference on Web Services (ICWS), pp. 309–316. IEEE (2017)