World Wide Web (2021) 24:375–396
https://doi.org/10.1007/s11280-020-00846-3

Leveraging pointwise prediction with learning to rank for top-N recommendation

Nengjun Zhu[1] · Jian Cao[1] · Xinjiang Lu[2] · Qi Gu[1]

Received: 7 March 2020 / Revised: 12 July 2020 / Accepted: 21 September 2020 /
Published online: 23 October 2020

Abstract

Pointwise prediction and Learning to Rank (L2R) are two hot strategies to model user preference 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 cause the problem of overfitting, while L2R is more prone to higher variance. On the other hand, the advantages of multi-task learning and ensemble learning inspire us to utilize multiple approaches jointly so that methods can promote together synergistically. Therefore, we propose a new framework called CPL, where pointwise prediction and L2R are inherently combined 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., Factorized SLIM (Sparse LInear Method) and GAPfm (Graded Average Precision factor model), to perform pointwise prediction and L2R, respectively. Different from the original version of SLIM, FSLIM reconstructs a denser representation both for users and items. Moreover, the low-rank users' and item's latent factor matrices act as a bridge between FSLIM and GAPfm. Extensive experiments on four real-world datasets show that CPLmg significantly outperforms the compared methods. To explore other possible combinations for CPL further, we

This article belongs to the Topical Collection: *Special Issue on Web Information Systems Engineering 2019*
Guest Editors: Reynold Cheng, Nikos Mamoulis, and Xin Huang

✉ Jian Cao
cao-jian@sjtu.edu.cn

Nengjun Zhu
zhu_nj@sjtu.edu.cn

Xinjiang Lu
luxinjiang@baidu.com

Qi Gu
guqi@sjtu.edu.cn

[1] Shanghai Institute for Advanced Communication and Data Science, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China

[2] Baidu Business Intelligence Lab, Baidu Research, Beijing, China

selected another pointwise method, i.e., FunkSVD, and L2R approach, i.e., BPR, to implement CPLdb. The experimental results demonstrate the superiority of CPL again as it can help improve the performance of its pointwise prediction and L2R components.

Keywords Implicit feedback · Personalized recommendation · Collaborative filtering · Learning to Rank · Metrics optimization · Similarity method

1 Introduction

Recommender systems have been widely adopted by many online services, since they are able to solve the problem of information overload as well as facilitate interactions between users and systems. Most recommender systems infer users' interests through users' historical behaviors, either represented in explicit form or implicit form. Explicit feedbacks, such as ratings, are given by users and they can indicate a users' interest in a particular product. However, explicit feedbacks are not always available in practical systems. On the contrary, implicit feedbacks, such as users' browsing history, search patterns, 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.

Pointwise prediction and learning to rank (L2R) are two representative genres of the approaches for recommender systems [31]. 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 binary relevance data 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. However, pointwise prediction models are inclined to generate large bias, i.e., overfitting problem, since they are confined to being finely tuned to each value of individual examples, even it is a noise. 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 example unless it is compared to the other one. In order to balance variance and bias, regularization terms are added to the target functions of both methods.

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. We train the two models iteratively with shared underlying variables, and trade-off controlling parameters are auto-tuned. In addition to the real partial relations between items, we also apply L2R component to model potential partial relations, which are mined according to the predicted results of the pointwise prediction component. Then, the estimated value of each example is supposed to be approaching the real value as well as keeping the correct preferential relations between items.

In this paper, to verify the effectiveness of CPL, we choose SLIM [18] which is one of the pointwise prediction methods, and GAPfm [26] which is one of the L2R methods, as the two components to implement the 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 the similarities between the items' attribute vectors. GAPfm takes the Graded Average Precision (GAP) metric as the extreme optimization objective function which is proposed to address the top-N recommendation problem in domains with grade relevance data. 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 of which the user is unaware [19, 23, 30].

In order to combine SLIM and GAPfm in a better way, we revise their original versions and combine them into the CPLmg framework, 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: it has a sparse coding coefficient matrix to improve the training process and to mine the similarities between items and it can utilize feature selection in the same way as SLIM. The difference is that FSLIM reconstructs a denser representation both for users and items, which can improve the recommendation performance [13]. Then, the low-rank users' and item's latent factor matrices act as a bridge between FSLIM and GAPfm, so that the learned denser representation can be transferred to each other. Moreover, we pass the relevant information from positive examples to unknown items according to the learned item similarities for GAPfm, and the confidence score of a grade to be the threshold that separates unknown items is also updated dynamically in every training round. Thus, the combination of FSLIM and GAPfm results in the considerably improved learning accuracy of GPLmg. Extensive experiments on four real-world datasets demonstrate the effectiveness of CPL and CPLmg. The experimental results show that CPLmg significantly outperforms the compared methods, and CPL framework can help improve the performance of its pointwise prediction and L2R components.

Finally, to further verify CPL's superiority and flexibility, other possible pointwise approaches, e.g., FunkSVD, and L2R approaches, e.g., BPR, are selected to implement another instantiation of CPL, which is named CPLdb in the experiment section. Although the performance of CPLdb is not as good as CPLmg, it is still better than FunkSVD and BPR. It further proves the combination can make a more significant impact than individual efforts.

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 in domains with binary implicit feedback.
2. We propose an implementation of CPLmg for CPL. In CPLmg, we combine the FSLIM and GAPfm models. FSLIM is extended from SLIM by incorporating the idea of matrix factorization (MF) methods. Moreover, strategies are designed to better integrate FSLIM and GAPfm.
3. We explore another possibility to implement CPL and choose FunkSVD and BPR to conduct the pointwise and L2R components of CPL.
4. Extensive experiments are conducted, which show that CPLmg outperforms other baselines on various evaluation metrics, and CPL framework promotes the performance of its pointwise prediction and L2R components.

The rest of this paper is organized as follows: we first introduce the related work in Section 2. In Section 3, we give the definitions and notations as well as describe the two basic models, SLIM and GAPfm. Then, in Section 4, CPLmg is described with its two components in detail. The experimental evaluation and CPLdb are introduced in Section 5, which is followed by the conclusions and future work in Section 6.

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. Besides, since SLIM, an original component of our model, is also a kind of similarity method, we also provide an overview of recent research on similarity methods related to SLIM.

2.1 Matrix factorization (MF)

Collaborative filtering (CF) is a core technology in recommender systems, which infers user preferences by utilizing relationships between users or between items directly or indirectly. It can be divided into memory-based methods and model-based methods [6]. Matrix factorization is one of the most popular model-based methods. It learns latent factor representations with respect to users and items to model user preferences as the dot product of latent factor vectors. It has multiple extensions, such as SPMF [2], SVD++ [12], PMF [22] and HeteroMF [9]. Due to the excellent scalability of MF, some recent work has explored the combination of MF and other technologies, such as deep learning for recommendation. Neural network-based collaborative filtering (NCF) [8] is a general framework and is essentially a fusion model of Generalized Matrix Factorization (GMF) and Multi-Layer Perceptron (MLP). ConvMF [11] is a novel context-aware recommendation model that integrates the convolutional neural network (CNN) into probabilistic matrix factorization (PMF). The work in [5] proposed another hybrid model, which makes use of both the rating matrix and side information. It also combines the Additional Stacked Denoising Autoencoder (aSDAE) and matrix factorization together.

2.2 Similarity methods (SM)

The similarity method models the user preference for an item based on user similarities or item similarities, which are called user-based nearest neighbourhood (UserKNN) and item-based nearest neighbourhood (ItemKNN), respectively. In traditional CF methods, the similarity is calculated based on user historical data according to certain criteria, such as Jaccard and Cosine. SLIM [18] 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 [10] 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 [23] and GSLIM [13] are proposed to overcome this limitation. LRec is interpreted as a linear classification model for each user, and GSLIM learns the latent factor matrices of the users and the items via a traditional matrix factorization approach, followed by reconstructing the latent user or item matrix via prototypes that are learned using sparse coding. As SLIM is a particular case of MF, it inherits the excellent scalability of MF, which results in SLIM having many variants incorporating contextual information or auxiliary information, such as CSLIM [29], a contextual SLIM recommendation algorithm. Furthermore, there are many studies on the localized SLIM, such as GLSLIM [4] and LorSLIM [3].

2.3 Learning to rank approaches (L2R)

Learning to Rank (L2R) is a popular research area, since it directly models partial ordering relations between items, which happens to be in 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 [20] maximizes AUC (the area under the curve) metrics by utilizing the partial order relations between items. CLiMF [25] and xCLiMF [27] are two L2R methods based on mean reciprocal rank (MRR) and expected reciprocal rank (ERR) [1], respectively. Moreover, SVM-MAP [28] and TFMAP [24] optimize MAP metric directly. To alleviate the overfitting problem of L2R, GTRM [30] optimizes the metric group-oriented mean average precision (GMAP) that considers the similarities between items, and PRIGP [19] integrates item-based pairwise preferences and item group-based pairwise preferences into the framework based on BPR-OPT derived from BPR.

Local-based L2R is another hot research topic in recommender systems. This research learns partial ordering relations between items based on local information instead of global data. For example, LLORMA [14] and LCR [15] assume that the rating matrix is low-rank within certain neighborhoods of the metric space and thus optimize these objective measures, such as Frobenius norm loss function and pairwise loss function, based on local data.

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 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 $\mathcal{U} = \{u = 1, 2, \ldots, M\}$ and $\mathcal{I} = \{i = 1, 2, \ldots, N\}$, respectively. $\mathcal{O}(u)$ is the set of all observed items of user u and $\overline{\mathcal{O}}(u) = \mathcal{I} \backslash \mathcal{O}(u)$ denotes the set of all unobserved items of user u. 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, and each latent vector has K-dimension. The entire set of user 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 that user u has at some point interacted with item i, otherwise the entry is marked as 0.

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 ^{T} (e.g., $\mathbf{P}_u^T$) are row vectors.

3.1 SLIM

Item-based approaches based on SLIM (Sparse LInear Methods) have demonstrated very good performance for top-N recommendation. 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 $N \times N$ aggregation coefficient matrix $\mathbf{W}$, 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:

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

where $\mathbf{A}_u^T$ 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, [16] showed that it could be ignored without affecting performance.

3.2 GAPfm

GAPfm is a listwise L2R method which directly optimizes a smoothed approximation of the GAP metric. GAP generalizes average precision (AP) to the case of multi-graded relevance and inherits the most important property of the AP metric which guarantees 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:

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

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 threshold 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, as defined as follows:

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

Then, with a small manipulation, (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 of which are given in the following sections.

4 Proposed methodology

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

4.1 Factorized SLIM (FSLIM)

Factorized SLIM is a new version of SLIM that incorporates ideas from traditional matrix factorization (MF) methods. 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 representation of each item is no longer a numerical value, i.e., 1, but is a latent factor vector, i.e., $\mathbf{Q}_i$. Then, the entry of $\mathbf{A}_{ui}$ is estimated as the product of the user latent vector and the linear combination of item latent vectors as follows:

$$\widetilde{\mathbf{A}}_{ui} = g\left(\mathbf{P}_u^T \sum_{j \in \mathcal{O}(u)} \mathbf{Q}_j \mathbf{W}_{ji}\right) \tag{5}$$

where $g(x) = 1/(1 + e^{-k(x-c)})$ is a sigmoid function, which is a common choice for one-class recommendation. In order to simplify the analysis and computation process, we suppose $c = 0$ and $k = 1$ in $g(x)$. $\mathbf{W}$ is a sparse aggregation coefficient matrix like that in SLIM. Taking into account all the observed data, the final loss function is defined as follows:

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

where $diag(\mathbf{W}) = 0$, and we drop the constraint of $\mathbf{W} \geq 0$ compared to SLIM. We have previously provided the reason for this.

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

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

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

$$\frac{\partial \mathcal{L}_F}{\partial \mathbf{W}_{ij}} = -(\mathbf{A}_{ui} - g(\widetilde{\mathbf{A}}_{ui}))g'(\widetilde{\mathbf{A}}_{ui})\mathbf{P}_u^T \mathbf{Q}_j + \beta_3 \mathbf{W}_{ij} \pm \lambda \tag{9}$$

where $g'(x) = g(x)/(1 - g(x))$ is the derivative of function $g(x)$. Then, with a learning step size η_1, the updated formula for FSLIM is as follows:

$$\Theta^{(t)} \leftarrow \Theta^{(t-1)} - \eta_1 \frac{\partial \mathcal{L}_F}{\partial \Theta} \tag{10}$$

where Θ represents the model parameter, and $\Theta^{(t)}$ is the value of Θ at t-th SGD step.

4.2 GAPfm with sampling strategy

The work in [26] 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 [21] to dynamically mine potential positive examples 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 learned from FSLIM to estimate the connection strength of each unobserved item for each user, which can be demonstrated as:

$$y_{ui} = \begin{cases} g\left(\mathbf{P}_u^T \sum\limits_{j \in \mathcal{O}(u)} \mathbf{Q}_j \mathbf{W}_{ji}\right), & i \notin \mathcal{O}(u) \\ 1, & i \in \mathcal{O}(u) \end{cases} \tag{11}$$

It is likely that some values will be prefilled 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-N 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 $Y(u)$. We let pn denote the number of the potential preferred items we select. Then, we change the threshold 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:

$$\delta_u(y) = \frac{y}{\sum_{t \in Y(u)} t} \tag{12}$$

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

$$GAP_u = \frac{1}{Z_u} \sum_{i=1}^{N} \frac{I_{ui}}{R_{ui}} \sum_{j=1}^{N} S_{uij} I_{uj} \mathbb{I}(R_{uj} \leq R_{ui}) \tag{13}$$

where S_{uij} is defined as follows:

$$\begin{aligned} S_{uij} = {} & \mathbb{I}(y_{ui} < y_{uj}) \sum_{t \in Y(u) \& t \leq y_{ui}} \delta_u(t) \\ & + \mathbb{I}(y_{uj} \leq y_{ui}) \sum_{t \in Y(u) \& t \leq y_{uj}} \delta_u(t) \end{aligned} \tag{14}$$

and Z_u is defined as follows:

$$Z_u = \sum_{t \in Y(u)} n_{ut} \sum_{l \in Y(u) \& l \leq t} \delta_u(l) \tag{15}$$

Then we use $g(x)$ function and parameters $\mathbf{P}$, $\mathbf{Q}$ to estimate the term of $\frac{1}{R_{ui}}$ and $\mathbb{I}(R_{uj} \leq R_{ui})$ in (13) to get a smoothed version of GAP as follows:

$$GAP_u \approx \frac{1}{Z_u} \sum_{i=1}^{N} I_{ui} g(f_{ui}) \sum_{j=1}^{N} S_{uij} I_{uj} g(f_{u(j-i)}) \tag{16}$$

where $f_{ui} = \mathbf{P}_u^T \mathbf{Q}_i$ and $f_{u(j-i)} = \mathbf{P}_u^T (\mathbf{Q}_j - \mathbf{Q}_i)$. Then, taking into account all users and adding two Frobenius norms $\|\mathbf{P}\|_F$ and $\|\mathbf{Q}\|_F$ as well as a parameter β to control the magnitude of regularization, the final objective function of GAPfm is shown as follows:

$$\mathcal{L}_G = \sum_{u=1}^{M} \sum_{i=1}^{N} I_{ui} g(f_{ui}) \sum_{j=1}^{N} S_{uij} I_{uj} g(f_{u(j-i)}) - \frac{\beta_4}{2} \|\mathbf{P}\|_F^2 - \frac{\beta_5}{2} \|\mathbf{Q}\|_F^2 \tag{17}$$

Note that, (17) 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:

$$\begin{aligned} \frac{\partial \mathcal{L}_G}{\partial \mathbf{P}_u} = \sum_{i=1}^{N} I_{ui} \Bigg(& g'(f_{ui}) \sum_{j=1}^{N} I_{uj} S_{uij} g(f_{u(j-i)}) \cdot \mathbf{Q}_i \\ & + g(f_{ui}) \sum_{j=1}^{N} I_{uj} S_{uij} g'(f_{u(j-i)}) \cdot (\mathbf{Q}_j - \mathbf{Q}_i) \Bigg) - \lambda \mathbf{P}_u \end{aligned} \tag{18}$$

$$\begin{aligned} \frac{\partial \mathcal{L}_G}{\partial \mathbf{Q}_i} = I_{ui} \Bigg(& g'(f_{ui}) \sum_{j=1}^{N} I_{uj} S_{uij} g(f_{u(j-i)}) + \sum_{j=1}^{N} I_{uj} \\ & [S_{uji} g(f_{uj}) - S_{uij} g(f_{ui})] g'(f_{u(j-i)}) \Big) \mathbf{P}_u - \lambda \mathbf{Q}_i \end{aligned} \tag{19}$$

Then, the updated formula for GAPfm is:

$$\Theta^{(t)} \leftarrow \Theta^{(t-1)} + \eta_2 \frac{\partial \mathcal{L}_F}{\partial \Theta} \tag{20}$$

in which the direction to update the parameters is reversed compared to (10).

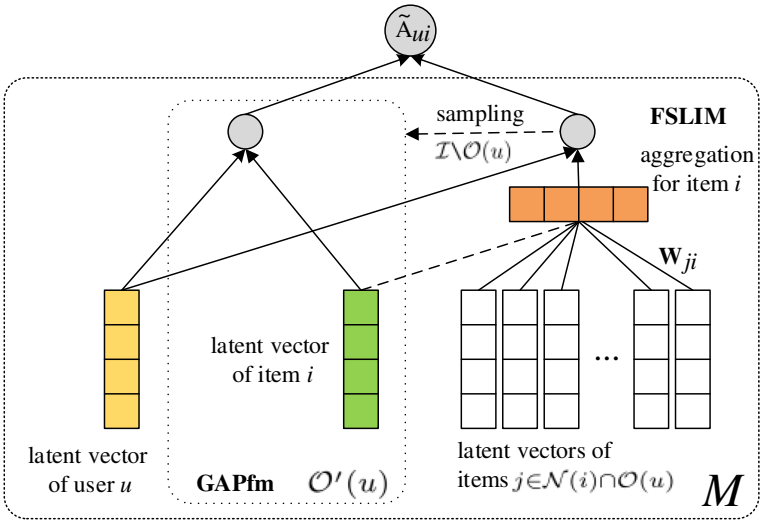


Figure 1 The framework of CPLmg

4.3 CPLmg recommendation model

Now, we introduce how to combine FSLIM and GAPfm under the MF framework to implement CPLmg, so that they can mutually reinforce each other and can better learn from the complex user-item interactions. A straightforward approach is to define an objective function as a weighted linear combination of $\mathcal{L}_F$ and $\mathcal{L}_G$, i.e., $\mathcal{L} = \alpha * \mathcal{L}_F + (\alpha - 1)\mathcal{L}_G$. However, by doing so, it might lower the training efficiency as previously mentioned. For example, it means FSLIM cannot use the stochastic gradient decent technology that updates model parameters according to each gradient of the individual examples, since the other part of CPLmg, the GAPfm, must compute GAP_u based on all observed items for each user. Besides, this combination makes it hard to apply a dynamic sampling process which will be discussed later.

We propose to train FSLIM and GAPfm using a multi-task learning approach [17]. FSLIM and GAPfm are updated iteratively according to their respective objective functions with shared underlying variables, i.e., the latent factor matrices $\mathbf{P}$ and $\mathbf{Q}$, and the learned item similarity matrix $\mathbf{W}$, as shown in Figure 1. The matrices $\mathbf{P}$ and $\mathbf{Q}$ are jointly updated by FSLIM and GAPfm in each co-training round, and the item similarity matrix $\mathbf{W}$ helps GAPfm mine potential positive examples. 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. The entire learning algorithm of CPLmg is illustrated in Algorithm 1.

Algorithm 1 CPLmg.

Input: All users with their observed items on training set, the number of near neighbourhoods $iknn$, the size of candidate potential positive examples pn, the regularization terms λ, β_1, β_2, β_3, β_4, β_5, the learning rate parameters η_1, η_2, and the maximal number of iteration $itermax$.

Output: The learned latent factors $\mathbf{P}$, $\mathbf{Q}$, and the learned item similarity matrix $\mathbf{W}$.

Initialize $\mathbf{P}^{(0)}$, $\mathbf{Q}^{(0)}$, $\mathbf{W}^{(0)}$ with random values, empty the set $Y(u)$ for all users.
while $t \leq itermax$ **do**
  for $u = 1$ *to* M **do**
    for $i \in \mathcal{O}(u)$ **do**
      for $j \in \mathcal{O}(u)$ *and* $j \neq i$ **do**
        $\mathbf{W}_{ij}^{(t+1)} \leftarrow \mathbf{W}_{ij}^{(t)} - \eta_1 \frac{\partial \mathcal{L}_F}{\partial \mathbf{W}_{ij}}$ based on (9)
      $\mathbf{P}_i^{(t+1)} \leftarrow \mathbf{P}_i^{(t)} - \eta_1 \frac{\partial \mathcal{L}_F}{\partial \mathbf{P}_i}$ based on (7)
      $\mathbf{Q}_u^{(t+1)} \leftarrow \mathbf{Q}_u^{(t)} - \eta_1 \frac{\partial \mathcal{L}_F}{\partial \mathbf{Q}_u}$ based on (8)
    update $\mathcal{O}'(u)$ and $Y(u)$ based on (11)
    for $i \in \mathcal{O}'(u)$ **do**
      $\mathbf{Q}_i^{(t+1)} \leftarrow \mathbf{Q}_i^{(t)} + \eta_2 \frac{\partial \mathcal{L}_G}{\partial \mathbf{Q}_i^{(t)}}$ based on (19)
    $\mathbf{P}_u^{(t+1)} \leftarrow \mathbf{P}_u^{(t)} + \eta_2 \frac{\partial \mathcal{L}_G}{\partial \mathbf{P}_u}$ based on (18)
  $t \leftarrow t + 1$
return $\mathbf{P} = \mathbf{P}^{(t)}, \mathbf{Q} = \mathbf{Q}^{(t)}, \mathbf{W} = \mathbf{W}^{(t)}$

4.4 Discussion

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

$$\widetilde{\mathbf{A}}_{ui} = g\left(\mathbf{P}_u^T \sum_{j \in \mathcal{O}(u) \cup \{i\}} \mathbf{Q}_j \mathbf{W}'_{ji}\right) \tag{21}$$

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., $\widetilde{\mathbf{A}}_{ui} = g\left(\mathbf{P}_u^T \sum_{j \in \mathcal{O}(u)} \mathbf{Q}_j \mathbf{W}_{ji}\right)$ and in GAPfm, i.e., $\widetilde{\mathbf{A}}_{ui} = g(\mathbf{P}_u^T \mathbf{Q}_i)$, 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 unobserved item set $\overline{\mathcal{O}}(u)$ with the largest prediction values based on (21) are recommended to the user.

Components 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.

Time complexity The time complexity of CPLmg comprises two parts: the time cost of FSLIM and the time cost of GAPfm.

Regarding FSLIM, for each user-item pair, the time complexities of updating the latent factors for each user based on (7) and for each item based on (8) are $O(|\mathcal{O}(u)|K)$ and $O(|\mathcal{O}(u)|^2K)$ respectively, where $|\mathcal{O}(u)|$ denotes the number of all observed items to user u, and K is the dimensionality of latent factor vectors. Then, taking into account all the positive user-item pairs, the total complexities of updating matrix $\mathbf{P}$ and matrix $\mathbf{Q}$ are $O(M|\bar{I}|K)$ and $O(M|\bar{I}|^3K)$ in each iteration, respectively, where the $|\bar{I}|$ is the average number of observed items across all users. Based on (9), the time cost of updating $\mathbf{W}$ is also $O(M|\bar{I}|^3K)$ in each iteration.

As for GAPfm, the time complexity of updating the latent factors for each user based on (18) is $O(|\mathcal{O}(u)|^2K)$. Taking into account all M users, the total complexity of updating matrix $\mathbf{P}$ is $O(M|\bar{I}|^2K)$ in each iteration. Based on (19), the time complexity of updating the latent factors with respect to all observed items to a given user u is also $O(|\mathcal{O}(u)|^2K)$, since each observed item is computed to every observed item ranked higher than it. Similarly, in each iteration, the total complexity of updating matrix $\mathbf{Q}$ across all users is the same as that of updating matrix $\mathbf{P}$, i.e., $O(M|\bar{I}|^2K)$. In the process of sampling, we compute the prediction value of each unobserved item $i \in \overline{\mathcal{O}}(u)$ based on (5) for each user u, followed by a sort operation, and thus the time cost is $O(|\bar{J}|(|\bar{I}| + \ln(|\bar{J}|))$, where $|\bar{J}| = N - |\bar{I}|$ represents the average number of unobserved items across all users. However, $|\bar{J}|$ is

Table 1 The datasets used in evaluation

Dataset	#users[a]	#items	#trans	Density[b]	Ratings[c]
AppData	20,467	40,259	1,022,339	0.124%	–
ML100K	943	1,682	100,000	6.30%	1–5
FilmTrust	1,508	2,071	35,497	1.14%	0.5–4.0
LastFM	1,892	17,632	92,834	0.28%	–

[a]Columns corresponding to #users, #items and #trans show the number of users, the number of items and the number of transactions in each dataset

[b]Density = #trans/(#users × #items)

[c]The column corresponding to ratings shows the rating range of each dataset, where AppData and LastFM only contain positive examples

quite large in real systems. Thus, to control the time complexity, we randomly compute the prediction value of some unobserved items.

The time costs of FSLIM and GAPfm accumulate linearly, and thus CPLmg finally takes $O(M|\bar{I}|^3K + M|\bar{J}|(|\bar{I}| + \ln(|\bar{J}|))$ time costs to update the parameters in each iteration. Note that the term $(M|\bar{I}|)$ represents the number of all observed items across all users. We substitute $|R|$ for $(M|\bar{I}|)$, and $|R| \gg |\bar{I}|, |\bar{J}|, K$. This means the time complexity in each iteration can be regarded as $O(|R|)$. This is a linear complexity with respect to the number of all positive examples in the given dataset.

5 Experimental results

We implement CPLmg based on a Java library for recommendation systems (**LibRec**) [7]. LibRec is an open-source project with some current state-of-the-art algorithms, and we choose some of these to compare them with our CPLmg method, so as to validate the effectiveness of CPLmg. We also test the influence of different component and parameter settings followed by a discussion.

5.1 Data sets and experimental setup

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

The first dataset AppData is from users' log files where the users' interactive behaviors with mobile applications are recorded for one month. 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 MovieLens-100K and FilmTrust[1] are two public dataset in the field of movie recommendation systems. They are both organized in rating forms. However, since we only discuss the one-class recommendation problem in this paper, the ratings are converted to the

[1]https://www.librec.net/datasets.html#filmtrust

appropriate binary, i.e., all user-item pairs with ratings are treated as positive examples, the others are the missing ones.

LastFM contains music artist listening information. The dataset is released in the framework of the international workshop on information heterogeneity and fusion in recommender systems.[2] In the experiment, mobile applications, movies, and artists are the items to be recommended.

5.2 Implementation details

We randomly select records from the users' historical data to keep a certain number of observed items for each user as test data, and set the rest of the records as the training set. For example, "Given 5" denotes that for each user, we randomly select five observed items as unknowns in the training set, but as positive examples in the test set, i.e., in the future, the user will be interested in these items. Then, recommendation lists are produced by the trained models. Finally, we measure the performance over these positive examples in the test data.

All the parameters of the compared methods are set to optimal values in the experiments, e.g., the number of features (the dimensionality of feature vectors) on the four datasets are all 10. To simplify the analysis, the regularization parameters of Forbenius norms in CPLmg are the same in each component, i.e., $\beta_1 = \beta_2 = \beta_3 = 0.5$ and $\beta_4 = \beta_5 = 0.01$.

5.3 Evaluation metrics

Precision is a common evaluation metric in recommender systems. In the case of top-N recommender systems, MAP 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 three metrics to evaluate the performance of the new and compared methods in this paper.

Precision reflects the ratio of relevant items in the ranked list given a truncated position, e.g., given a top-N recommendation list $R(u)$ and a set $T(u)$ that records all observed items of user u on test set, precision is defined as

$$\textbf{Precision} = \frac{\sum_{u \in \mathcal{U}} |R(u) \cap T(u)|}{\sum_{u \in \mathcal{U}} |R(u)|} \tag{22}$$

MAP is a binary list-based metric that gives larger credit to correctly recommended items in top ranks. $\textbf{AP@}N$ is defined as the average of precision computed at all positions before the position at N with a relevant item, namely,

$$\textbf{AP@}N = \frac{\sum_{k=1}^{N} \textbf{Precision@}k \times rel(k)}{\sum_{k=1}^{N} rel(k)} \tag{23}$$

where $\textbf{Precision@}k$ is the ratio of relevant items in the ranked list given a truncated position k, and $rel(k)$ is an indicator function equaling 1 if the item at position k is a relevant item, 0 otherwise. Finally, $\textbf{MAP@}N$ is defined as the mean of the $\textbf{AP@}N$ scores across all users.

[2] http://ir.ii.uam.es/hetrec2011

MRR is the average value of the reciprocal ranks of recommendation results for the first relevant item at the highest position and is defined as

$$\mathbf{MRR} = \sum_{u=1}^{M} \frac{1}{rank_u} \tag{24}$$

where $rank_u$ refers to the rank position of the first relevant item for user u.

5.4 Experimental comparisons with previous models

We compare our methods CPLmg and FSLIM with six baseline ones, namely iPOP, ItemKNN, FISMauc, NeuMF, GAPfm and SLIM. The implementations of iPOP, ItemKNN, FISMauc and SLIM are provided by LibRec. A description of the compared methods is as follows:

- **iPOP.** It recommends a certain number of the most popular items from the training set to all users.
- **ItemKNN.** It is a traditional item-based collaborative filtering method, and we use Jaccard similarity to calculate the relations between items.
- **FISMauc** [10]. FISM is a variant of SLIM and it factorizes the similarity matrix into two low rank matrices as discussed in Section 2. FISMauc considers ranking errors based on loss function and obtains better performance than FISMrmse, which considers the pointwise squared error loss function, in our experiments. Therefore, we do not further report on the performance of FISMrmse.
- **NeuMF** [8]. It is a state-of-the-art method using neural network-based collaborative filtering (NCF) framework.
- **SLIM** [18]. It is a novel learning model, which is the pointwise prediction component of our model. SLIM directly learns a similarity matrix between items from the data.
- **GAPfm** [26]. It is an another component of our model, and it belongs to one kind of L2R approach.

For each model, the parameters were empirically tuned to their optimal values in the experiments and these 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 section based on the top-5 recommendation.

Table 2 shows that both CPLmg and FSLIM achieve better performance than the baselines according to all three metrics, especially CPLmg, which is the best. It proves that the proposed CPLmg is highly competitive for top-N recommendation tasks for reasons previously discussed. We can also observe that the performance of FSLIM is better than SLIM. This indicates that denser representations of the user and item matrix can better model 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 the SLIM and GAPfm approaches are more effective for top-N recommendation. 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

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

Method	AppData/Given 3					AppData/Given 10				
	Params		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**
Method	FilmTrust/Given 3					FilmTrust/Given 10				
	params		Precision	MRR	MAP	params		Precision	MRR	MAP
iPOP	–	–	0.2928	0.5795	0.3951	–	–	0.5524	0.7345	0.4981
ItemKNN	50	–	0.2958	0.5850	0.3967	50	–	0.5420	0.7201	0.4845
FISMauc	0.6	$1e^{-5}$	0.2204	0.3805	0.2786	0.8	$1e^{-5}$	0.5004	0.7046	0.4513
GAPfm	0.02	–	0.3031	0.5998	0.4087	0.01	–	0.5626	0.7432	0.5128
SLIM	0.1	0.5	0.2948	0.5979	0.4061	0.1	0.5	0.5637	0.7487	0.5187
FSLIM	0.05	0.15	0.3072	0.6076	0.4103	0.05	0.15	0.5721	0.7546	0.5279
NeuMF	10	–	0.3164	0.6101	0.4238	10	–	0.5792	0.7503	0.5342
CPLmg	250	25	**0.3278**	**0.6257**	**0.4451**	250	25	**0.5888**	**0.7705**	**0.5505**
Method	LastFM/Given 3					LastFm/Given 10				
	params		Precision	MRR	MAP	params		Precision	MRR	MAP
iPOP	–	–	0.0276	0.0771	0.0285	–	–	0.0849	0.1883	0.0554
ItemKNN	50	–	0.0456	0.1337	0.0535	50	–	0.1350	0.2815	0.1006
FISMauc	0.5	$5e^{-5}$	0.0274	0.0724	0.0269	0.5	$1e^{-5}$	0.0849	0.1883	0.0554
GAPfm	0.05	–	0.0460	0.1167	0.0465	0.05	–	0.1288	0.2661	0.0966
SLIM	0.2	0.6	0.0512	0.1426	0.0591	0.1	0.5	0.1329	0.2928	0.1173
FSLIM	0.05	0.005	0.0647	0.1597	0.0669	0.02	0.001	0.1497	0.3068	0.1290
NeuMF	10	–	0.0723	0.1602	0.0729	8	–	0.1602	0.3182	0.1389
CPLmg	300	50	**0.0924**	**0.1812**	**0.0809**	300	50	**0.1803**	**0.3305**	**0.1529**

Bold values represent the best performance in the experiments

Table 3 Different approaches to combine pointwise prediction and L2R

Combo	Method	Pointwise	L2R	Sampling strategy
FSLIM + GAPfm	CPLmg-dy	FSLIM	GAPfm	Dynamical sampling
	CPLmg-st	FSLIM	GAPfm	Static sampling
	CPLmg-no	FSLIM	GAPfm	Without sampling
FunkSVD + BPR	CPLdb-dy	FunkSVD	BPR	Dynamical sampling
	CPLdb-st	FunkSVD	BPR	Static sampling
	CPLdb-no	FunkSVD	BPR	Without sampling

of given items increases from 3 to 10. The reason for this lies in the fact that more relevant items in the test data can better reveal the preferences of users and more relevant items in the test means a higher chance of ranking potential positive examples in the top positions.

5.5 Analysis of CPLmg components

To explore the effectiveness of our proposed framework, i.e., CPL, as well as the instantiation of CPL, i.e., CPLmg, we modify the original version of CPLmg to some variants and then compare their performance with CPLmg.

The key points of CPL include three parts: 1) the pointwise prediction module, 2) the L2R module, and 3) the bridge between the two modules. In CPLmg, the third part further involves two operations: 1) the decomposed matrices are shared between the pointwise prediction module and L2R module, and 2) the dynamical sampling strategy is conducted when training the model. To simplify the analysis, we let the variants of CPLmg all have shared decomposed matrices. Then, there are three factors whose effects on the CPLmg should be tested. They are the pointwise prediction module, L2R module, and the sampling strategy. We list some possible configurations of these three factors in Table 3.

In addition to the combination of FSLIM and GAPfm, there are many other combinations of various pointwise methods and L2R approaches. However, we can't test them all. Therefore, we choose two classical and representative methods, i.e. FunkSVD[3] and BPR [20], to replace FSLIM and GAPfm in CPLmg respectively, to implement a new instantiation of CPL named CPLdb. CPLdb is conducted to demonstrate whether the CPL can also improve the performance of other single pointwise methods and L2R approaches. Equations (25) and (26) summarize the objective functions of FunkSVD and BPR, respectively.

$$\mathcal{L}_{svd} = \sum_{(u,i)\in\Omega_{svd}} \|\mathbf{A}_{ui} - g(\mathbf{P}_u^\top \mathbf{Q}_i)\|_F^2 + \frac{1}{2}(\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|\|_F^2) \tag{25}$$

$$\mathcal{L}_{bpr} = \sum_{(u,i,j)\in\Omega_{bpr}} -g(\mathbf{P}_u^\top \mathbf{Q}_i - \mathbf{P}_u^\top \mathbf{Q}_j) + \frac{1}{2}(\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|\|_F^2) \tag{26}$$

where Ω is the set of examples. For FunkSVD, Ω_{svd} records all observed user-item pairs, while for BPR, Ω_{bpr} records all (user, positive item, negative item) triple tuples. $g(\cdot)$ is a sigmoid function as that in (5).

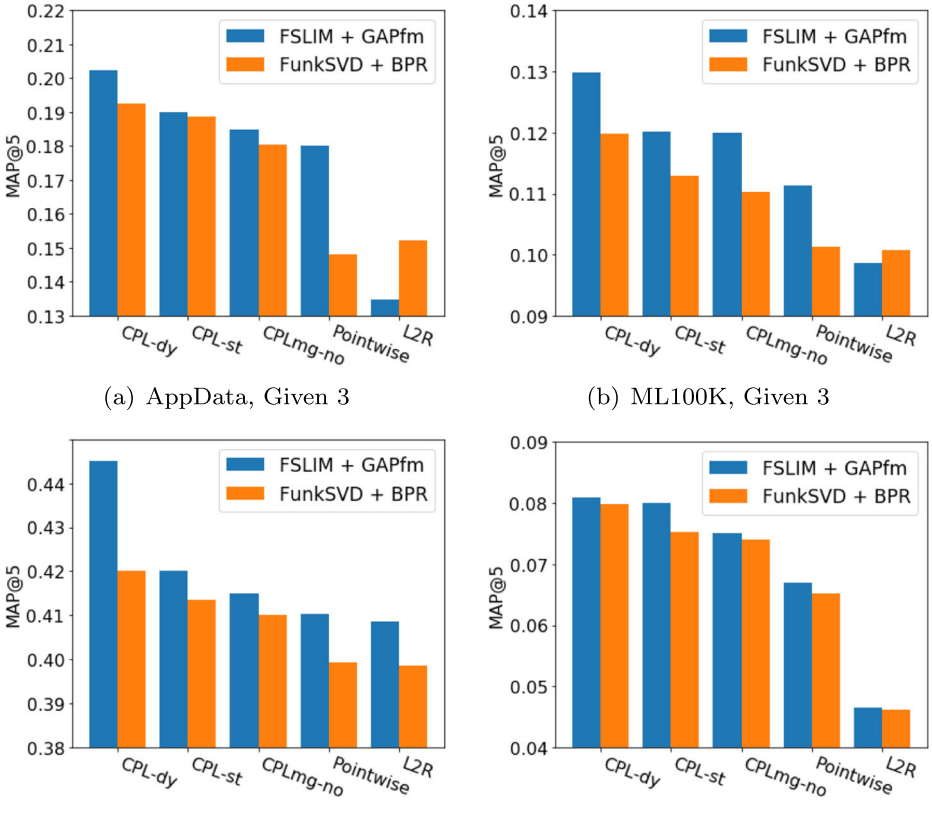
On the other hand, we further investigate the influences from different settings of the sampling strategy. We have already introduced a dynamical sampling strategy when we

[3] https://sifter.org/%7Esimon/journal

describe the CPLmg. The sampling process occurs in the joint training round with the help of the sample division mechanism of GAPfm. Alternatively, we can adopt a static sampling technique, i.e., we first train the pointwise model and then we feed the predicted values of unobserved items into the L2R method as potential positive examples. There exists new partial relations between the real and potential positive examples as well as between the potential positive and negative examples. L2R approaches have an inherent advantage in modeling these relations. Also, we want to know the performance of CPL if there is no sampling process. In this setting, we only have a unified objective function of pointwise method and L2R approach to be trained, i.e., $\mathcal{L}_{CPLmg-no} = \mathcal{L}_F - \mathcal{L}_G$ and $\mathcal{L}_{CPLdb-no} = \mathcal{L}_{svd} + \mathcal{L}_{bpr}$.

So far, we have six instantiations shown in Table 3, where CPLmg-dy in this part stands for the original version of CPLmg for a better distinction. Since BPR has no sample division mechanism like GAPfm, in each training iteration, we select top-N unobserved items with the largest values, which are predicted by FunkSVD, as potential positive examples for each user in CPLdb-dy. The value of N is finely tuned to 20 in the experiment.

Figure 2 shows the experimental results of all six instantiations. From Figure 2, we can have the following four key observations:



(a) AppData, Given 3

(b) ML100K, Given 3

(c) FilmTrust, Given 3

(d) LastFM, Given 3

Figure 2 Experimental results on different ways to combine pointwise prediction and L2R

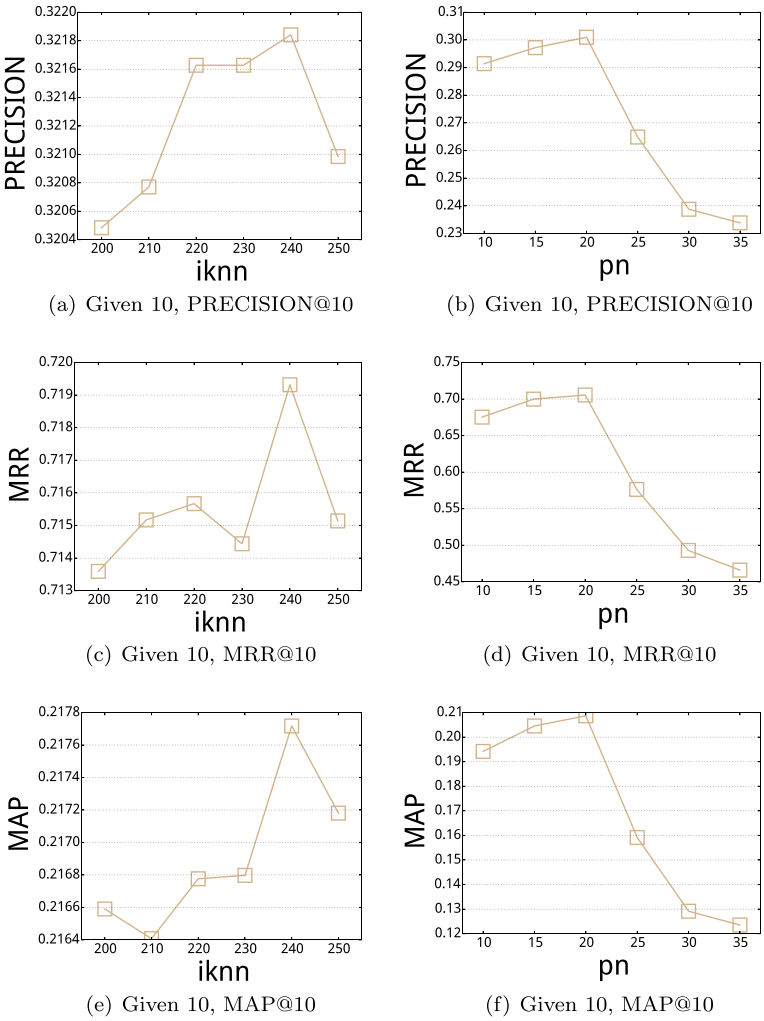
- With the parameters carefully adjusted, the unified framework, i.e., CPL can outperform its two single units, i.e., pointwise prediction method and L2R approach. For example, almost all CPL-dy, CPL-st, and CPL-no approaches can achieve better performance than single components, i.e., FSLIM, GAPfm, FunkSVD, and BPR. It proves the effectiveness of CPL framework.
- Generally, the performance of CPL-dy is better than that of CPL-st and CPL-no. Compared to the "static sampling" and "without sampling" strategies, the "dynamic sampling" strategy can enhance the interaction between the pointwise method and L2R approach, which might be the reason for performance improvements.
- Although CPLdb, including CPLdb-dy, CPLdb-st, and CPL-no, can improve the performance of FunkSVD and BPR, its corresponding performance is not as good as CPLmg's. One reason for this result might be as follows: the performance of the components of CPLmg, i.e., FSLIM and GAPfm, is generally better than that of CPLdb, i.e., FunkSVD and BPR.
- Furthermore, CPLmg-dy outperforms CPLdb-dy. It might be not only caused by the performance of the components, but also caused by the sample division mechanism of GAPfm. CPLmg-dy has a more elaborate process to evaluate potential positive examples.

5.6 Analysis of CPLmg parameters

In this section, we describe the experiments conducted to explore the influences of the main parameters on CPLmg, such as the number of neighborhoods $iknn$ for the feature selection algorithm, the size of the candidate potential positive examples pn, the learning rate parameters η_1 and η_2. It is also worth pointing out, when we change the value of one of these parameters, the others are set to their optimal values, i.e., $iknn = 245$, $pn = 22$, $\eta_1 = 0.05$, $\eta_2 = 0.00001$. Based on the optimal settings, all the experiment results given in this section are under the condition that "Given 10". Without loss of generality and due to the space limitation, we only report the parameter influences for the top-10 recommendations on AppData. Similar results are observed on the other datasets.

5.6.1 *iknn*

We first conduct an experiment to investigate the influence of the number of the nearest neighborhoods $iknn$. This parameter influences the experiment results in two phases: (1) updating the gradient of FSLIM, and (2) selecting the candidate potential positive examples for GAPfm. To simplify the analysis, we set the same value of $iknn$ for these two phases. The results are shown in Figure 3a, c, and e. 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 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.



(a) Given 10, PRECISION@10 (b) Given 10, PRECISION@10

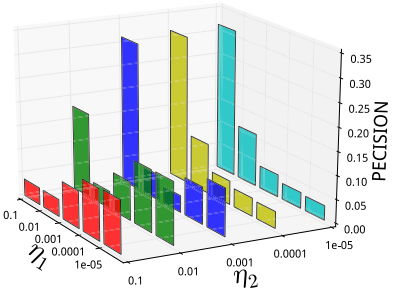
(c) Given 10, MRR@10 (d) Given 10, MRR@10

(e) Given 10, MAP@10 (f) Given 10, MAP@10

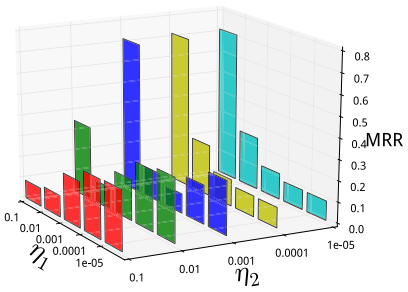
Figure 3 Experimental results on different parameters *iknn* and *pn*

5.6.2 *pn*

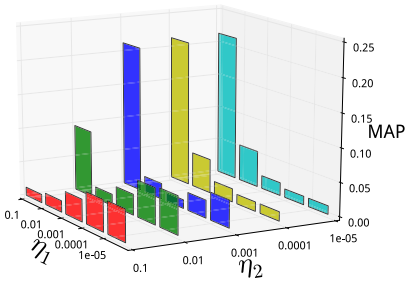
The value of *pn* controls the number of candidate potential preferred items in the sampling process of GAPfm. The influence of *pn* on the recommendation performance is shown in Figure 3b, d, and f. We can observe that precision, MRR, and MAP performance can be improved by properly increasing *pn*. However, when the increase 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 feedbacks, since the selected missing values can alleviate the overfitting risk.



(a) Given 10, PRECISION@10



(b) Given 10, MRR@10



(c) Given 10, MAP@10

Figure 4 Experimental results on different parameters η_1 and η_2

5.6.3 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, where η_1 is from (10) and η_2 is from (20). η_1 and η_2 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 influences of η_1 and η_2 are shown in Figure 4. We observe that all criteria show the same trends 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 increase of divergence between these two parameters.

6 Conclusions and future work

In this paper, we first analyzed the strengths and limitations of traditional pointwise prediction and L2R methods. Then, we proposed a new framework, CPL, where pointwise prediction and L2R are inherently combined to improve the performance of top-N recommendations. Moreover, to verify the effectiveness of CPL, we implemented two instantiations of CPL: CPLmg and CPLdb. The former takes FSLIM and GAPfm as its

two components, where FSLIM is a variant of SLIM by infusing denser representations. Whereas, the latter takes FunkSVD and BPR as its components. The components reinforce each other through information interchange based on the denser representations and aggregation coefficients. The experimental results show that CPLmg significantly outperforms the compared methods, and CPL framework can help improve the performance of the pointwise and L2R components in CPLmg and CPLdb.

There are also some potential research topics. Firstly, the combination approach between two components of CPLmg can be extended. For example, we would like to explore a more complex combination motivated by NCF [8], which fuses two components based on the neural network framework. Secondly, other models of pointwise prediction and L2R methods can be tried in the framework. Thirdly but not lastly, we can make use of other side information, such as the number of times users' visited items or the social relationships among users to infer the connection strengths between users and items.

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

References

1. Chapelle, O., Metlzer, D., Zhang, Y., Grinspan, P.: Expected reciprocal rank for graded relevance. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management, pp. 621–630. ACM (2009)
2. Chen, H., Niu, D., Lai, K., Xu, Y., Ardakani, M.: Separating-plane factorization models: scalable recommendation from one-class implicit feedback. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, pp. 669–678. ACM (2016)
3. Cheng, Y., Yin, L., Yu, Y.: Lorslim: low rank sparse linear methods for top-n recommendations. In: 2014 IEEE International Conference on Data Mining (ICDM), pp. 90–99. IEEE (2014)
4. Christakopoulou, E., Karypis, G.: Local item-item models for top-n recommendation. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 67–74. ACM (2016)
5. Dong, X., Yu, L., Wu, Z., Sun, Y., Yuan, L., Zhang, F.: A hybrid collaborative filtering model with deep structure for recommender systems. In: AAAI, pp. 1309–1315 (2017)
6. Friedrich, G., Zanker, M.: A taxonomy for generating explanations in recommender systems. AI Mag. **32**(3), 90–98 (2011)
7. Guo, G., Zhang, J., Sun, Z., Yorke-Smith, N.: Librec: a java library for recommender systems. In: UMAP Workshops (2015)
8. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.-S.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web, pp. 173–182. International World Wide Web Conferences Steering Committee (2017)
9. Jamali, M., Lakshmanan, L.: Heteromf: recommendation in heterogeneous information networks using context dependent factor models. In: Proceedings of the 22nd International Conference on World Wide Web, pp. 643–654. International World Wide Web Conferences Steering Committee (2013)
10. Kabbur, S., Ning, X., Karypis, G.: Fism: factored item similarity models for top-n recommender systems. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 659–667. ACM (2013)
11. Kim, D., Park, C., Oh, J., Lee, S., Yu, H.: Convolutional matrix factorization for document context-aware recommendation. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 233–240. ACM (2016)
12. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 426–434. ACM (2008)
13. Larraín, S., Parra, D., Soto, A.: Towards improving top-n recommendation by generalization of slim. In: RecSys Posters (2015)
14. Lee, J., Kim, S., Lebanon, G., Singer, Y.: Local low-rank matrix approximation. ICML (2) **28**, 82–90 (2013)

15. Lee, J., Bengio, S., Kim, S., Lebanon, G., Singer, Y.: Local collaborative ranking. In: Proceedings of the 23rd International Conference on World Wide Web, pp. 85–96. ACM (2014)
16. Levy, M., Jack, K.: Efficient top-n recommendation by linear regression. In: RecSys Large Scale Recommender Systems Workshop (2013)
17. 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)
18. Ning, X., Karypis, G.: Slim: sparse linear methods for top-n recommender systems. In: 2011 IEEE 11th International Conference on Data Mining (ICDM), pp. 497–506. IEEE (2011)
19. Qiu, S., Cheng, J., Yuan, T., Leng, C., Lu, H.: Item group based pairwise preference learning for personalized ranking. In: Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 1219–1222. ACM (2014)
20. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-fifth Conference on Uncertainty in Artificial Intelligence, pp. 452–461. AUAI Press (2009)
21. Robertson, S.E., Kanoulas, E., Yilmaz, E.: Extending average precision to graded relevance judgments. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 603–610. ACM (2010)
22. Salakhutdinov, R., A.Mnih.: Bayesian probabilistic matrix factorization using Markov Chain Monte Carlo. In: Proceedings of the 25th International Conference on Machine Learning, pp. 880–887. ACM (2008)
23. Sedhain, S., Menon, A.K., Sanner, S., Braziunas, D.: On the effectiveness of linear models for one-class collaborative filtering. In: AAAI, pp. 229–235 (2016)
24. 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 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 155–164. ACM (2012)
25. Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Oliver, N., Hanjalic, A.: Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In: Proceedings of the Sixth ACM Conference on Recommender Systems, pp. 139–146. ACM (2012)
26. Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Hanjalic, A.: Gapfm: optimal top-n recommendations for graded relevance domains. In: Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management, pp. 2261–2266. ACM (2013)
27. 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 7th ACM Conference on Recommender Systems, pp. 431–434. ACM (2013)
28. Yue, Y., Finley, T., Radlinski, F., Joachims, T.: A support vector method for optimizing average precision. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 271–278. ACM (2007)
29. Zheng, Y., Mobasher, B., Burke, R.: Cslim: contextual slim recommendation algorithms. In: Proceedings of the 8th ACM Conference on Recommender Systems, pp. 301–304. ACM (2014)
30. Zhu, N., Cao, J.: Gtrm: a top-n recommendation model for smartphone applications. In: 2017 IEEE International Conference on Web Services (ICWS), pp. 309–316. IEEE (2017)
31. Zhu, N., Cao, J.: Cpl: a combined framework of pointwise prediction and learning to rank for top-n recommendations with implicit feedback. In: Proceedings of the International Conference on Web Information Systems Engineering, pp. 259–273. Springer (2019)