Collaborative List-and-Pairwise Filtering from Implicit Feedback

Article *in* IEEE Transactions on Knowledge and Data Engineering · August 2020

DOI: 10.1109/TKDE.2020.3016732

6 authors, including:

Runlong Yu
University of Science and Technology of China

Mingyue Cheng
University of Science and Technology of China

Enhong Chen
University of Science and Technology of China

Collaborative List-and-Pairwise Filtering From Implicit Feedback

Runlong Yu, Qi Liu, *Member, IEEE*, Yuyang Ye, Mingyue Cheng, Enhong Chen, *Senior Member, IEEE*, and Jianhui Ma

Abstract—The implicit feedback based collaborative filtering (CF) has attracted much attention in recent years, mainly because users implicitly express their preferences in many real-world scenarios. The current mainstream pairwise methods optimize the Area Under the Curve (AUC) and are empirically proved to be helpful to exploit binary relevance data, but lead to either not address the ranking problem, or not specifically focus on top-k recommendation. Although there exists the listwise method maximizes the Mean Reciprocal Rank (MRR), it has low efficiency and is not particularly adequate for general implicit feedback situations. To that end, in this paper, we propose a new framework, namely *Collaborative List-and-Pairwise Filtering (CLAPF)*, which aims to introduce pairwise thinking into listwise methods. Specifically, we smooth another well-known rank-biased measure called Mean Average Precision (MAP), and respectively combine two rank-biased metrics (MAP, MRR) with the pairwise objective function to capture the performance of top-k recommendation. Furthermore, the sampling scheme for CLAPF is discussed to accelerate the convergence speed. Our CLAPF framework is a new hybrid model that provides an idea of utilizing rank-biased measures in a pairwise way on implicit feedback. Empirical studies demonstrated CLAPF outperforms state-of-the-art approaches on real-world datasets.

Index Terms—Recommender systems, collaborative filtering, implicit feedback, top-k recommedation

1 Introduction

Collaborative filtering (CF) has been widely used techniques in recommender systems [1], [2], [3], [4]. It generates recommendations by leveraging the user-item interactions derived from historical data. Previously, most researches on collaborative filtering focus on explicit feedback [5], like the numerical ratings. However, in some real-world scenarios, explicit feedback is not always available [6]. Contrarily, there are many types of data in the one-class form [7], e.g., transactions in E-commerce platforms, thumb-ups in online social networks, and watch records in online video platforms. Such data do not contain the scoring (ratings) between users and items, which are usually called one-class [8] or implicit feedback [6]. Implicit feedback differs from explicit feedback: the latter explicitly expresses users' positive and negative preferences through the rating scores, while the former contains only positive feedback. Therefore, huge unobserved item feedbacks cannot be simply considered as negative preferences, in views of the items which may not be seen by users before [8].

As aforementioned, the implicit feedback problem usually poses challenges of lacking negative feedback, especially in cases of sparse data [9]. A lot of negative examples and missing positive examples are mixed together and cannot be distinguished, which makes many existing classification algorithms not directly applicable to the problem [10]. In general, previous methods for dealing with implicit feedback can be divided into two groups [11], [12], [13]: (1) *pointwise regression methods*, and (2) *pairwise ranking methods*. Pointwise methods take implicit feedback as absolute preference scores and minimize a pointwise square loss to approximate the absolute rating scores [6], [8], while pairwise methods train recommendation models by optimizing the Area Under the Curve (AUC) measure, which is essentially based on pairwise comparisons between a sample of relevant items and a sample of irrelevant items. For example, Bayesian Personalized Ranking (BPR) [14] is one of the most popular approaches that adopt such pairwise preference assumption. Given an observed user-item interaction (u, i) and an unobserved user-item interaction (u, j), BPR assumes that a user u has a higher preference on item i than on item j.

Research shows that the pairwise methods are significantly preferable to the pointwise ones [15], and have been the preferred solutions for implicit feedback problem. Many pairwise methods improve over BPR, e.g., Multiple Pairwise Ranking (MPR) [16] further taps the connections among items with multiple pairwise ranking criteria. However, the AUC measure optimized by these pairwise methods does not well reflect the quality of recommendation lists because it is not a rank-biased measure [17]. That means most of the pairwise methods may not perform well

- *Runlong Yu, Qi Liu, Enhong Chen, and Jianhui Ma are with the Anhui Province Key Laboratory of Big Data Analysis and Application (BDAA), School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China. E-mail: yrunl@mail.ustc.edu.cn, {qiliuql, cheneh, jianhui}@ustc.edu.cn.*
- *Yuyang Ye is with the Management Science and Information Systems Department, Rutgers University, Newark, NJ 07102. E-mail: yuyang.ye@rutgers.edu.*
- *Mingyue Cheng is with the Anhui Province Key Laboratory of Big Data Analysis and Application (BDAA), School of Data Science, University of Science and Technology of China, Hefei, Anhui 230027, China. E-mail: mycheng@mail.ustc.edu.cn.*

Manuscript received 11 Oct. 2019; revised 20 June 2020; accepted 31 July 2020. Date of publication 0 . 0000; date of current version 0 . 0000.
(Corresponding authors: Qi Liu and Jianhui Ma.)
Recommended for acceptance by G. Koutrika.
Digital Object Identifier no. 10.1109/TKDE.2020.3016732

in terms of top-k recommendation, which is becoming more critical in personalized recommendation [18]. Although there exists some work that generalizes pairwise ranking to listwise ranking via direct optimization of rank-biased measure, it is difficult to model the inter list loss and has low efficiency [10], e.g., Collaborative Less-is-More Filtering (CLiMF) [17] maximizes a rank-biased metric called Mean Reciprocal Rank (MRR) [20] for a few historical items given to the individual user. In addition, research shows that such listwise methods can commonly improve the performance based on multi-classification datasets significantly, like explicit data, but not adequate for accurate characterization of binary-classification datasets, like implicit data [19].

In this paper, we propose a new hybrid CF framework, namely *Collaborative List-and-Pairwise Filtering (CLAPF)*, to solve the problem. We first summarize and categorize the existing work on collaborative filtering from implicit feedback. Then we optimize another well-known rank-biased measure called Mean Average Precision (MAP) [21], which calculates the precision at the position of every correct item in the ranked resulting lists of the recommender. Compared with the AUC, MAP is a listwise measure and usually provides users with the more valuable top-ranked recommendation; Compared with the MRR, MAP is more applicable to multiple correct responses (hits) in the resulting lists [22]. After that, we combine the objective functions of optimizing the above two rank-biased metrics (MAP, MRR) with the pairwise objective function and propose our CLAPF. CLAPF framework can be regarded as a new hybrid model that presents a new perspective to utilizing rank-biased measures in a pairwise way on implicit feedback. As many negative sampling strategies used by pairwise methods sampling from the unobserved items of each user are not suitable for CLAPF, we design a new sampling strategy, namely *Double Sampling Strategies (DSS)*, which places more emphasis on both the rank information of positive and negative items for each gradient step, to further focus on the model convergence. Experiments on real-world datasets clearly validate the effectiveness of our CLAPF framework and DSS sampler compared with several baselines. Three contributions of the paper include:

- We propose an approach for smoothing MAP. As MAP is an important rank-biased measure, studying the smooth form of MAP is of great significance for understanding item ranking in recommendations.
- For implicit feedback problem, we provide a novel idea of combining the listwise and pairwise objective functions, which not only digs users' implied preferences on items from huge unobserved data, but also achieves an efficient method of addressing the ranking problem.
- We propose a sampling strategy, which involves the rank information of both positive and negative items. Experiments demonstrate the sampling strategy accelerates the convergence speed of CLAPF.

Overview. The rest of this paper is organized as follows. In Section 2, we will summarize some related work of our study. Section 3 will introduce the notations, problem definition, and briefly give some previous optimization criteria, which will be used later. Then, the formulation of our proposed CLAPF and the learning process will be detailed in Section 4. Afterward, we will discuss the sampling problem and propose a new sampler in Section 5. Section 6 comprehensively evaluates the model performance in real-world datasets. Finally, conclusions will be drawn in Section 7.

2 Related Work

The related work of our study can be grouped into two categories, namely Pairwise Methods and Ranking-oriented CF.

2.1 Pairwise Methods

For solving implicit feedback problem, pairwise methods have been the mainstream solutions. Most pairwise methods are the improvement of BPR algorithm and can be categorized into six classes which will be respectively introduced below. (1) *Relaxing the two fundamental assumptions in BPR.* Some studies argue that the two fundamental assumptions made in BPR, namely individual preference assumption over two items and independence assumption between two users, may not always hold in practice [23], [24]. MPR relaxes the individual preference assumption by tapping the connections among items with multiple pairwise ranking criteria [16], while Group Bayesian Personalized Ranking (GBPR) relaxes the independence assumption among users by considering that users' preferences are influenced by other users with the same interests [23]. (2) *Improving the sampling strategies in BPR.* BPR samples negative items from the unobserved items with equal probabilities for every user. However, some researchers have found that uniform sampler is highly ineffective, especially for long-tail or large-scale datasets. Therefore, Dynamic Negative Sampling (DNS) [25], Adaptive Oversampling Bayesian Personalized Ranking (AoBPR) [26] and Alpha-Beta Sampling (ABS) [27] are proposed which dynamically pick negative training samples from a ranking list produced by the current prediction model and iteratively update the list containing all unobserved items. (3) *Improving the objective function in BPR.* The AUC metric is not for quantifying such a recommender list where positive items placed on the top, negative items placed at the bottom, and unknown items in between. To address this issue, Song, *et al.* [28] introduce a generalized AUC (GAUC) that measures both head and tail of a ranking list. (4) *Mining implicit information via additional data.* For example, Ding, *et al.* focus on the purchase feedback and propose a sampler for BPR with probabilistic weights based on the additional view data of the E-commerce domain. Moreover, Yu, *et al.* leverage view data to classify the uncertainly negative items [16]. (5) *Introducing transfer learning to BPR.* Since most of the pairwise methods are confined to one domain of data source, some work has concerned the question of modeling preferences across distinct domains. CroRank [29] is a typical approach that bridges users' inclinations transferred from the auxiliary domain to the target domain for a better recommendation. (6) *Combining BPR with specific application issues.* Because pairwise methods have achieved success in solving implicit feedback problem, some studies apply BPR to practical applications and find that it can greatly improve performance and productivity, e.g., teaching path recommendation [30], [31], technology forecasting [32], [33], talent recommendation [34], etc.

To learn pairwise objective functions, most approaches are implemented by matrix factorization. Nowadays, since deep neural networks (DNNs) have shown success in computer vision, natural language processing, and so on [35], some work attempts to leverage neural networks to learn pairwise objective functions instead of matrix factorization. Specifically, Xiangnan He, *et al.* [36] propose a general framework called Neural Collaborative Filtering (NCF), which models users and items as feature embeddings, to be fed into neural layers for learning interactions. An advanced instantiation of NCF is NeuMF which consists of generalized matrix factorization and multi-layer perceptron to model latent feature interactions. NeuPR proposes an alternative approach so that the negative sampler in NCF is unnecessary [37]. In addition to neural networks, there is also some work that leverages graphs to model user-item interactions, while its pairwise objective function is the same as BPR but optimized by graph learning algorithms [38]. It is worth mentioning that, DNNs are not only used to learn pairwise ranking, but also to learn pointwise regression in some work [39]. However, there are a number of empirical studies showing deep models do not always generate better recommendations [40]. Therefore, it can be considered that matrix factorization based models are still the mainstream way for handling implicit feedback problem, which leads us to adopt matrix factorization to design our algorithm and sampler in this paper.

2.2 Ranking-Oriented CF

As aforementioned, the criteria of pairwise methods do not well reflect the quality of the recommendation lists, as mistakes at different positions are penalized equally, which is not the expected behavior in a ranking list. As top-k recommendation has become a common choice in scenarios, the goal of recommending a satisfying sequential list for users becomes even more important. Several prior ranking-oriented CF algorithms typically use ranking-oriented objective functions to learn potential factors of users and items. Earlier, researches focus on probabilistic Latent Semantic Analysis (pLSA) for statistical modeling user preferences from ratings [41]. [42] further improves the traditional pLSA by directly modeling user preferences with a set of items rather than individual items. Later on, [43] proposes a similarity-based approach to leverage the ranks of items in the ranking list rather than the rating values, so does OrdRec [44] while it further put forward a pointwise regression of ranks by ratings. Collaborative Competitive Filtering (CCF) employs a multiplicative latent factor model to exploit the interactive choice process in recommender systems [45]. Some work addresses item ranking by labeling, e.g., [46] proposes a top-k labeling strategy based on context information and it outperforms five-graded feedback ("bad", "fair", "good", "excellent", "perfect"). Recently, more and more work pays attention to metric space. LCR [47] assumes that the rank matrix is low-rank in certain neighborhoods of the metric space defined by user-item pairs, and proposes to minimize a general empirical risk of ranking loss. Along this line, l-Injection [48] further adopts pre-use preferences of users to address the sparsity problem. Nowadays, there are methods leverages which listwise measures to design a ranking-oriented CF, e.g., ListCF [49] optimizes similar users' probability distributions over permutations of the items to estimate a preference ranking based on ratings. However, most of these methods are not specially designed for general recommendation scenarios with implicit no-graded relevance scores from users to items [50], [51]. Later on, Shi, *et al.* [17] propose CLiMF to deal with one-class data by directly maximizing the MRR and achieve better ranking results for implicit feedback problem, which makes CLiMF become one of the most popular listwise approaches, but it has low efficiency.

Since our paper mainly addresses the smoothing and optimization process of MAP and MRR, here we discuss previous CF methods which attempt to optimize another ranking metric, namely NDCG, for making a distinction. In general, we can roughly divide them into two categories. The first category is to optimize NDCG in an explicit and interpretable way, like CoFiRank [52], the authors design a loss function to directly optimize NDCG, however, it is of extremely high time complexity due to sophisticated computation of NDCG and optimization processes. The second category is more common today, it aims to optimize NDCG in an implicit fashion without a smoothing objective function, like CRMF [53] and DNS [25], while it makes the approaches lack interpretability to some extent. Consequently, we intend to optimize ranking metrics in an explicit and efficient manner, which seems to be difficult to achieve by optimizing NDCG.

In summary, although there is some work that generalizes pairwise ranking to listwise via direct optimization of ranking measure [17], [54], it is difficult to model the inter list loss and has low efficiency. In addition, research shows that such listwise methods all adopt learning method based on structured estimation [19], which can commonly improve the performance based on multi-classification datasets significantly, like explicit data, and is not adequate for accurate characterization of binary-classification datasets, like implicit data [51], resulting in that such listwise methods are inferior to some pairwise methods on implicit feedback.

To solve the problem mentioned above, we consider linking the pairwise thinking and the listwise framework, and propose a new hybrid CF called CLAPF. Specifically, the listwise framework is designed for addressing the ranking problem, while the pairwise thinking can be effectively helpful to tap the implicit feedback information from data. In detail, we follow some outstanding ideas in CLiMF [17] and Multiple Pairwise Ranking (MPR) [16] to optimize the MAP and formulate the objective functions as multiple pairs. Besides, the computation complexity of CLAPF is acceptable. In particular, the convergence speed of learning the CLAPF can be further accelerated by a new sampler designed in this paper.

3 Preliminaries

In this section, we first introduce some notations and the definition of implicit feedback problem. Then the optimization criteria of pairwise methods and CLiMF which will be used in later sections are given briefly.

3.1 Notation and Problem Definition

We first give the notations and problem definition. $U = \{u\}_{u=1}^{n}$ is defined as the set of users and $I = \{i\}_{i=1}^{m}$ is defined as the set of items, where n and m represent the number of users and items, respectively. Each $u \in U$ has expressed her positive feedbacks on items $I_u^+ \subset I$. The number of observed items for user u in the given data collection is n_u^+. Y_{ui} denotes the binary relevance score of item i to user u, i.e., $Y_{ui} = 1$ if item i is relevant to user u, 0 for irrelevant. $\mathbb{I}(x)$ is an indicator function that is equal to 1, if x is true, and 0 for false. $\sigma(x)$ is the Sigmoid function, where $\sigma(x) = 1/(1+e^{-x})$.

R_{ui} denotes the rank of item i in the ranking list for user u, and the items are ranked in a descending order based on their predicted relevance scores for user u, which means that the higher the relevance score of the prediction, the smaller the rank of the item. f_{ui} denotes the predictor function that maps the parameters from user u and item i to a predicted relevance score. The predictor function is modeled by widely used matrix factorization as $f_{ui} = U_u V_i^T + b_i$, where U_u is a latent factor vector describing user u, V_i is a latent factor describing item i, and b_i is the bias of item i. The goal of implicit feedback problem is to recommend a personalized ranking list of items for user u from the unobserved item set $I \backslash I_u^+$ based on the predicted score f_{ui}.

3.2 Optimization Criteria of Pairwise Methods

Most of the optimization criteria of pairwise methods directly adopt the BPR criterion, which is fundamentally based on pairwise comparisons between an observed item and an unobserved item [14]. This criterion is mainly to optimize the AUC. The definition of AUC for user u is given by

$$AUC_u = \frac{1}{|I_u^+||I \backslash I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \backslash I_u^+} \mathbb{I}(R_{ui} < R_{uj}). \tag{1}$$

In BPR, researchers derive the approximation of $\mathbb{I}(R_{ui} < R_{uj})$ by using the differentiable loss as

$$\mathbb{I}(R_{ui} < R_{uj}) \approx \ln \sigma(f_{ui} - f_{uj}). \tag{2}$$

When neglecting the constant, we can obtain the objective function of BPR as

$$L_{BPR}(U_u, I) = \sum_{i \in I_u^+} \sum_{j \in I \backslash I_u^+} \ln \sigma(f_{ui} - f_{uj}). \tag{3}$$

In BPR, researchers point out that optimizing the objective function L_{BPR} means maximizing the individual probability that user prefers item i to item j, which contributes to i should rank higher than j, and can be expressed as

$$L_{BPR}(U_u, I) = \prod_{i \in I_u^+} \prod_{j \in I \backslash I_u^+} \Pr(R_{ui} < R_{uj}). \tag{4}$$

3.3 Optimization Criterion of CLiMF

Shi, *et al.* [17] propose CLiMF for dealing with implicit feedback by directly maximizing the Mean Reciprocal Rank (MRR) and achieve better ranking results on some usage scenarios, which makes CLiMF become one of the most popular listwise approaches. The definition of Reciprocal Rank of a recommendation list for user u, as defined in information retrieval [20], can be given by

$$RR_u = \sum_{i=1}^{m} \frac{Y_{ui}}{R_{ui}} \prod_{k=1}^{m} (1 - Y_{uk} \mathbb{I}(R_{uk} < R_{ui})). \tag{5}$$

Obviously, RR_u is dependent on the ranking of the observed items. In CLiMF, researchers smooth Reciprocal Rank in the same way as in BPR, and the smooth version of RR_u can be given by

$$RR_u = \sum_{i=1}^{m} Y_{ui} \sigma(f_{ui}) \prod_{k=1}^{m} (1 - Y_{uk} \sigma(f_{uk} - f_{ui})). \tag{6}$$

Although Eq. (6) is a smooth function with respect to the predicted relevance scores, optimizing this function could still be practically intractable, due to its multiplicative nature. The computational cost grows quadratically with the number of items, which is very large for most recommender systems. To solve the problem, a lower bound of the smooth version of RR_u can be derived and we finally have the objective function of CLiMF as

$$\begin{aligned} L_{CLiMF}(U_u, I) &= \sum_{i \in I_u^+} \ln \sigma(f_{ui}) + \sum_{i,k \in I_u^+} \ln(1 - \sigma(f_{uk} - f_{ui})) \\ &= \sum_{i \in I_u^+} \ln \sigma(f_{ui}) + \sum_{i,k \in I_u^+} \ln \sigma(f_{ui} - f_{uk}). \end{aligned} \tag{7}$$

Notice that we use $1 - \sigma(x) = \sigma(-x)$ to get the above formula. Through the objective function, we can find that the optimization criteria of listwise methods only focus on the observed items. Unlike the mainstream pairwise methods digging users' preference through pairs of the observed item and the unobserved item, the current listwise objective functions have no positive-unlabeled pairs and no unobserved items. However, in implicit feedback situations, users usually see fewer items and most items are unobserved, so we argue such an objective function exists limitations on the exploitation of huge unobserved information.

Overall, both pairwise and listwise methods optimize some kind of metrics and utilize informative observed items of users, but the only kind pairs of an observed item and an unobserved item in pairwise methods lead to insufficient ability on ranking performance, while the only kind pairs of two observed items in listwise methods lack ability for mining implicit information. In the following section, we will introduce the technical details of our CLAPF model for addressing the above problem.

4 Collaborative List-and-Pairwise Filtering

We will introduce CLAPF in the following three steps: smoothing the MAP, CLAPF formulation, and learning the CLAPF.

Specifically, we first smooth the Mean Average Precision (MAP) as a low bound version to make it can be optimized in a comparable time to pairwise methods. MAP is a listwise measure and usually provides users more valuable top-ranked recommendation. Some researchers try to maximize MAP in some application scenarios [55] but not in implicit feedback situations. Then, we respectively combine the smooth MAP and aforementioned MRR with the pairwise objective function to make these listwise methods more effective in top-k recommendation from implicit feedback. Finally, we illustrate the learning process of CLAPF using matrix factorization and Stochastic Gradient Descent (SGD) in detail.

4.1 Smoothing the MAP

MAP is defined as the average of AP across all the users [21]. The definition of AP of a ranked list for user u can be given by

$$AP_u = \frac{1}{\sum_{l=1}^{m} Y_{ul}} \sum_{i=1}^{m} \frac{Y_{ui}}{R_{ui}} \sum_{k=1}^{m} Y_{uk} \mathbb{I}(R_{uk} \le R_{ui}). \tag{8}$$

Obviously, AP_u is dependent on the rankings of the observed items. The rankings of the items change in a non-smooth way concerning predicted relevance scores, and therefore AP_u is a non-smooth function with respect to the model parameters. Thus we cannot use standard optimization methods to optimize AP_u. Based on insights in CLiMF, we approximate $\mathbb{I}(R_{uk} \le R_{ui})$ by using a Sigmoid function $\mathbb{I}(R_{uk} \le R_{ui}) \approx \sigma(f_{uk} - f_{ui})$, and approximate $\frac{1}{R_{ui}}$ by using another Sigmoid function $\frac{1}{R_{ui}} \approx \sigma(f_{ui})$, which makes the relationship that the higher the relevance score of the predict, the smaller the rank of the item. Then based on this trick, we reach a smoothed approximation of AP_u as

$$AP_u = \frac{1}{\sum_{l=1}^{m} Y_{ul}} \sum_{i=1}^{m} Y_{ui}\sigma(f_{ui}) \sum_{k=1}^{m} Y_{uk}\sigma(f_{uk} - f_{ui}). \tag{9}$$

Eq. (9) is a smooth function over the model parameters, but optimizing the function still has low efficiency. For example, the complexity of the gradient of Eq. (9) concerning the item feature parameter V_i is $O(m^2)$, so the computation complexity grows quadratically with the number of item m. Next, we propose a lower bound of Eq. (9) to make it can be optimized in a comparable time to pairwise methods.

The model parameters U_u, V_i can be obtained via maximizing Eq. (9) as

$$\begin{aligned} U_u, I &= \arg\max_{U_u, I}\{AP_u\} = \arg\max_{U_u, I}\{\ln(AP_u)\} \\ &= \arg\max_{U_u, I}\Bigg\{\ln\Bigg(\frac{1}{\sum_{l=1}^{m} Y_{ul}} \sum_{i=1}^{m} Y_{ui}\sigma(f_{ui}) \\ &\qquad \sum_{k=1}^{m} Y_{uk}\sigma(f_{uk} - f_{ui})\Bigg)\Bigg\}. \end{aligned} \tag{10}$$

Notice that $\sum_{l=1}^{m} Y_{ul} = n_u^+$, according to Jensen's inequality and the concavity of the Sigmoid function, then we have

$$\begin{aligned} &\ln(AP_u) \\ &= \ln\left(\sum_{i=1}^{m} \frac{Y_{ui}}{\sum_{l=1}^{m} Y_{ul}} \sigma(f_{ui}) \sum_{k=1}^{m} Y_{uk}\sigma(f_{uk} - f_{ui})\right) \\ &\ge \frac{1}{n_u^+} \sum_{i=1}^{m} Y_{ui} \ln\left(\sigma(f_{ui}) \sum_{k=1}^{m} Y_{uk}\sigma(f_{uk} - f_{ui})\right) \\ &= \frac{1}{n_u^+} \sum_{i=1}^{m} Y_{ui}\left(\ln\sigma(f_{ui}) + \ln\left(\sum_{k=1}^{m} Y_{uk}\sigma(f_{uk} - f_{ui})\right)\right) \\ &\ge \frac{1}{n_u^+} \sum_{i=1}^{m} Y_{ui}\left(\ln\sigma(f_{ui}) + \ln\left(\sum_{k=1}^{m} \frac{Y_{uk}}{n_u^+}\sigma(f_{uk} - f_{ui})\right)\right) \\ &\ge \frac{1}{n_u^+} \sum_{i=1}^{m} Y_{ui}\left(\ln\sigma(f_{ui}) + \frac{1}{n_u^+}\sum_{k=1}^{m} Y_{uk}\ln\sigma(f_{uk} - f_{ui})\right) \\ &= \frac{1}{n_u^+} \sum_{i\in I_u^+}\left(\ln\sigma(f_{ui}) + \frac{1}{n_u^+}\sum_{k\in I_u^+} \ln\sigma(f_{uk} - f_{ui})\right) \\ &\ge \frac{1}{(n_u^+)^2} \sum_{i\in I_u^+}\left(\ln\sigma(f_{ui}) + \sum_{k\in I_u^+} \ln\sigma(f_{uk} - f_{ui})\right) \\ &= \frac{1}{(n_u^+)^2}\left(\sum_{i\in I_u^+}\ln\sigma(f_{ui}) + \sum_{i\in I_u^+}\sum_{k\in I_u^+} \ln\sigma(f_{uk} - f_{ui})\right). \end{aligned} \tag{11}$$

The constant $\frac{1}{(n_u^+)^2}$ in the lower bound can be neglected. Then we can obtain a new objective function of optimizing the MAP measure as

$$L_{MAP}(U_u, I) = \sum_{i\in I_u^+} \ln\sigma(f_{ui}) + \sum_{i,k\in I_u^+} \ln\sigma(f_{uk} - f_{ui}). \tag{12}$$

We can take a close look at the two terms within the first summation. The maximization of the first term is the same as in Eq. (7), which contributes to learning latent factors that promote the observed item i. However, maximizing the second term turns to learn latent factors of the other observed items in order to increase their relevance scores, which is very different from the criterion of CLiMF given by Eq. (7). In summary, CLiMF leads to promote one observed item and scatter the others, while Eq. (12) makes a better balance of promoting and scattering two observed items at once.

4.2 CLAPF Formulation

As we have the objective functions of optimizing MAP and MRR measures in Eqs. (7) and (12), we can next analyze the functions from an individual probabilistic perspective and bring the pairwise thinking into listwise methods. Here, we just start with the MAP described by Eq. (12).

Similar to BPR, we respectively analyze the two terms in L_{MAP} function. Optimizing the first term $\sum_{i\in I_u^+}\ln\sigma(f_{ui})$ means maximizing the individual probability that user u prefers item i, which contributes to promoting the observed items as

$$\sum_{i\in I_u^+} \ln\sigma(f_{ui}) = \prod_{i\in I_u^+} \Pr(R_{ui}). \tag{13}$$

Optimizing the second term $\sum_{i,k\in I_u^+} \ln\sigma(f_{uk} - f_{ui})$ means maximizing the individual probability that user u prefers item k to item i, which contributes to k should rank higher than i as

$$\sum_{i,k\in I_u^+} \ln\sigma(f_{uk} - f_{ui}) = \prod_{i,k\in I_u^+} \Pr(R_{uk} < R_{ui}). \tag{14}$$

Similar to CLiMF, we can find that L_{MAP} is only dependent on the observed items, not exploring rich interactions in the unobserved items. In implicit feedback situations, users usually see fewer items and most items are unobserved items, so such an objective function poses insufficiency to a certain degree. Motivated by pairwise thinking represented as Eq. (4), we can inject the unobserved items into our objective function L_{MAP}. Based on Eq. (13), we can relax the criterion of promoting the observed item i, assuming that the promotion of the observed item i should rank higher than the unobserved item j, which is similar to Eq. (4). Using this trick, we can introduce pairwise ranking into our model and further exploit the hidden richer interactions in the unobserved items, expecting to further improve the recommendation performance.

We make a summary and derive our final objective function. Optimizing the second term Eq. (14) means maximizing the individual probability that user u prefers the observed item k to the other observed item i; Optimizing the first term Eq. (13) can be relaxed to maximizing the individual probability that user u prefers item i to item j, expressed as $\prod_{i\in I_u^+}\prod_{j\in I\backslash I_u^+} \Pr(R_{ui} < R_{uj})$. Now we have two different ranking targets described by individual probability related to two pairs of items. Facing the ranking problem about multiple pairs, inspired by MPR framework [16], we can maximize both of these two targets by maximizing their joint distribution probability of two ranking pairs. Then we have a new criterion called CLAPF-MAP, showing the overall likelihood for all users and items as

$$CLAPF-MAP = \prod_{u\in U}\prod_{i,k\in I_u^+}\prod_{j\in I\backslash I_u^+} \Pr(R_{uk} < R_{ui}, R_{ui} < R_{uj}). \tag{15}$$

We can represent the ranking pairs $R_{uk} < R_{ui}, R_{ui} < R_{uj}$ to be optimized for user u as follows:

$$\lambda(f_{uk} - f_{ui}) + (1-\lambda)(f_{ui} - f_{uj}), \tag{16}$$

where $0 \le \lambda \le 1$ is a tradeoff parameter used to fuse their relation, which can be determined via empirically testing a validation set. Following BPR, we use $\sigma(x)$ to approximate the probability $\Pr(\cdot)$ to make the objective function differentiable. Then the objective function of CLAPF-MAP can be represented as follows:

$$\min_{\Theta} -\ln CLAPF - MAP + \frac{1}{2}\mathcal{R}(\Theta), \tag{17}$$

where $\Theta = \{U_u \in \mathbb{R}^{1\times d}, V_i \in \mathbb{R}^{1\times d}, b_i \in \mathbb{R}, u \in U, i \in I\}$ is set of model parameters to be learned, and d is the number of latent factors

$$\begin{aligned}\ln CLAPF - MAP = &\sum_{u\in U}\sum_{i,k\in I_u^+}\sum_{j\in I\backslash I_u^+} \ln\sigma(\lambda(f_{uk} - f_{ui}) \\ &+ (1-\lambda)(f_{ui} - f_{uj})).\end{aligned} \tag{18}$$

Eq. (18) is the log-likelihood of CLAPF-MAP. $\mathcal{R}(\Theta) = \sum_{u\in U}\sum_{t\in S}[\alpha_u||U_u||^2 + \alpha_v||V_t||^2 + \beta_v||b_t||^2]$ is a regularization term to prevent overfitting in the learning process, and $S = \{i,k,j\}$ is a group of sampled items, where $i,k \in I_u^+$, and $j \in I\backslash I_u^+$.

Next, we formulate the MRR measure Eq. (7) with pairwise ranking in the same way. Optimizing the second item $\sum_{i,k\in I_u^+} \ln\sigma(f_{ui} - f_{uk})$ means maximizing the individual probability that user u prefers the observed item i to the other observed item k, expressed as $\prod_{i,k\in I_u^+} \Pr(R_{ui} < R_{uk})$; Optimizing the first term the same as Eq. (13) can be relaxed to maximizing the individual probability that user u prefers item i to item j, expressed as $\prod_{i\in I_u^+}\prod_{j\in I\backslash I_u^+} \Pr(R_{ui} < R_{uj})$. We maximize both of these targets by maximizing their joint distribution probability of two ranking pairs. By this mean, we can represent the ranking pairs as in the new criterion called CLAPF-MRR as follows:

$$\lambda(f_{ui} - f_{uk}) + (1-\lambda)(f_{ui} - f_{uj}), \tag{19}$$

where $0 \le \lambda \le 1$ is a tradeoff parameter used to fuse their relation. Then the objective function of CLAPF-MRR can be represented as

$$\min_{\Theta} -\ln CLAPF - MRR + \frac{1}{2}\mathcal{R}(\Theta). \tag{20}$$

Here, we directly give the log-likelihood of CLAPF-MRR in the same way as

$$\begin{aligned}\ln CLAPF - MRR = &\sum_{u\in U}\sum_{i,k\in I_u^+}\sum_{j\in I\backslash I_u^+} \ln\sigma(\lambda(f_{ui} - f_{uk}) \\ &+ (1-\lambda)(f_{ui} - f_{uj})).\end{aligned} \tag{21}$$

4.3 Learning the CLAPF

For CLAPF, when we learned the model parameters Θ, we can predict the user u's preference on an unobserved item j via commonly used matrix factorization $f_{uj} = U_u V_j^T + b_j$. Then the personalized ranking list for user u can be obtained via picking up the top-k largest preference scores of items which are the mostly relevant to the user.

The optimization problem of the objective functions in Eqs. (17) & (20) can be solved by employing the widely used Stochastic Gradient Descent (SGD) algorithm. The main process of SGD is to randomly select a record, which includes a user u, three items containing i, k, j, and iteratively update model parameters based on the sampled feedback records. Here we abbreviate Eqs. (16) or (19) as $R_{\succ u}$ and sampled items as S, then the tentative objective function of CLAPF-MAP or CLAPF-MRR can be written as $f(u,S) = -\ln\sigma(R_{\succ u}) + \frac{\alpha_u}{2}||U_u||^2 + \frac{\alpha_v}{2}\sum_{t\in S}||V_t||^2 + \frac{\beta_v}{2}\sum_{t\in S}||b_t||^2 = \ln[1+\exp(-R_{\succ u})] + \frac{\alpha_u}{2}||U_u||^2 + \frac{\alpha_v}{2}\sum_{t\in S}||V_t||^2 + \frac{\beta_v}{2}\sum_{t\in S}||b_t||^2$. We can update the corresponding parameters Θ by walking along the descending gradient direction

$$\Theta = \Theta - \gamma \frac{\partial f(u,S)}{\partial \Theta}, \tag{22}$$

where Θ can be $U_u, V_t, b_t, t \in S = \{i,k,j\}$, and $\gamma > 0$ is the learning rate.

Compared with BPR, the extra computational cost of CLAPF algorithm is mainly due to the calculation of gradient update for newly introduced one item k. The time complexity of the update rule in Eq. (22) is $O(d)$, where d is the number of latent features. Then the total time complexity of CLAPF is $O(Tnd)$, where T is the number of iterations and n is the number of users. Meanwhile, the time complexity for predicting a user's preference on an item is $O(d)$, the same as that in BPR. Thus, the computation complexity of our proposed approach CLAPF and the seminal approach BPR are comparable in terms of efficiency, which is much faster than the existing listwise methods.

5 Improving the CLAPF

We have introduced a new hybrid CF framework called CLAPF and two instantiations of CLAPF called CLAPF-MAP and CLAPF-MRR. Compared with pairwise methods, CLAPF makes a comparison of two observed items, which contributes a lot to the ranking problem in top-k recommendation; Compared with listwise methods, CLAPF deep taps the connection in the observed items and the unobserved items, which can exploit the hidden rich interactions among users and the unobserved items. In this section, we discuss the sampling problem under the objective functions of CLAPF and design a new sampling strategy for CLAPF.

5.1 The Sampling Problem

Sampling strategies play an important role in learning from data. Especially in CF areas, researches on pairwise ranking methods focus on building an adaptive sampler for the unobserved items. Among the samplers, Dynamic Negative Sampling (DNS) [25] and Adaptive Oversampling Bayesian Personalized Ranking (AoBPR) [26] have become the most popular ones by dynamically picking negative training samples from a ranking list produced by the current prediction model and iteratively updating the list containing all unobserved items. However, these negative sampling strategies are designed for the gradient vanish problem in the pairwise ranking field. As for ranking oriented CLAPF, we not only deal with the pair of the observed item and the unobserved item to make an accurate recommendation, but also focus on the pair of two observed items to address the ranking problem, so a sampling strategy containing all of the observed items and the unobserved items is much needed.

Similar to AoBPR, we first analyze a gradient of model parameter Θ of our CLAPF as

$$\frac{\partial f(u,S)}{\partial \Theta} = (1 - \sigma(R_{\succ_u})) \frac{\partial (R_{\succ_u})}{\partial \Theta}. \tag{23}$$

Learning the model parameter with CLAPF is done by looping over Eq. (22). As can be seen in Eq. (23), each gradient step has a multiplicative scalar $(1 - \sigma(R_{\succ_u}))$, which depends on how the scoring model (using current model parameters Θ) would discriminate between the pairs of a user u. Notice that, if $(1 - \sigma(R_{\succ_u}))$ is close to 0, nothing can be learned from the sample case S because its gradient vanishes, i.e., Θ is not changed by Eq. (22).

Thus, for given (u,i), we could choose (k,j) pair s.t. $R_{\succ_u}$ is small to increase $(1 - \sigma(R_{\succ_u}))$ and effectively update the model parameters. For CLAPF-MAP, $R_{\succ_u} = \lambda(f_{uk} - f_{ui}) + (1-\lambda)(f_{ui} - f_{uj})$, so instead of using a large f_{uk}, it is better to choose an item k with small predicted relevance score from the observed items; and instead of using a small f_{uj}, it is better to choose an item j with large predicted relevance score from the unobserved items. As for CLAPF-MRR algorithm, $R_{\succ_u} = \lambda(f_{ui} - f_{uk}) + (1-\lambda)(f_{ui} - f_{uj})$, so the item k and the item j both with large predicted relevance score from the observed items and the unobserved items can be considered as good sample case. To sample such cases, a ranking list is first generated according to the predicted relevance score to help probability-driven sample from the global data. As most of the real-world data follow long-tail distributions, the geometric sampler is adopted to sample from the ranking lists.

5.2 Double Sampling Strategy

Here, we propose a new sampler for CLAPF, namely Double Sampling Strategy (DSS), and give an illustration of DSS in Fig. 1.

To speed up the learning convergence of CLAPF, the sampler consists of two parts where the first part is a negative sampler for the item j, while the second part is a positive sampler for the item k. In addition, we uniformly sample the item i from the observed items of user u. In detail, for the instantiation CLAPF-MAP, we sample the item k and the item j by the following steps.

- *Step (1):* Model the users and the items by matrix factorization and get the latent factor representation of users and items.
- *Step (2):* Randomly pick a factor f_q, and rank the items by descending order according to the latent factor values, then get the ranking list.
- *Step (3):* For current user u and random factor f_q, return $\mathrm{sgn}(U_{u,q})$, where U is the latent representation of users, $U_{u,q}$ is the value in factor f_q related to user u, and $\mathrm{sgn}(\cdot)$ is the sign function.
- *Step (4):* If $\mathrm{sgn}(U_{u,q}) \geq 0$, return the item k from the observed items by geometric sampling the bottom items in the ranking list; and the item j from the unobserved items by geometric sampling the top items in the ranking list; Otherwise, if $\mathrm{sgn}(U_{u,q}) < 0$, reverse the ranking list and then do the same thing.

As for CLAPF-MRR, Step (4) changes as

- *Step (4):* If $\mathrm{sgn}(U_{u,q}) \geq 0$, return the item k from the observed items by geometric sampling the top items in the ranking list; and the item j from the unobserved items by geometric sampling the top items in the ranking list; Otherwise, if $\mathrm{sgn}(U_{u,q}) < 0$, reverse the ranking list and then do the same thing.

Based on the above steps, DSS gives two sampled items k, j from the observed items and the unobserved items. Compared with uniform sampling, the extra computational cost of DSS sampler is mainly due to the ranking process in Step (2). Thus we can easily follow AoBPR and DNS and reset the ranking lists every $\log(|m|)$ iterations, where m is

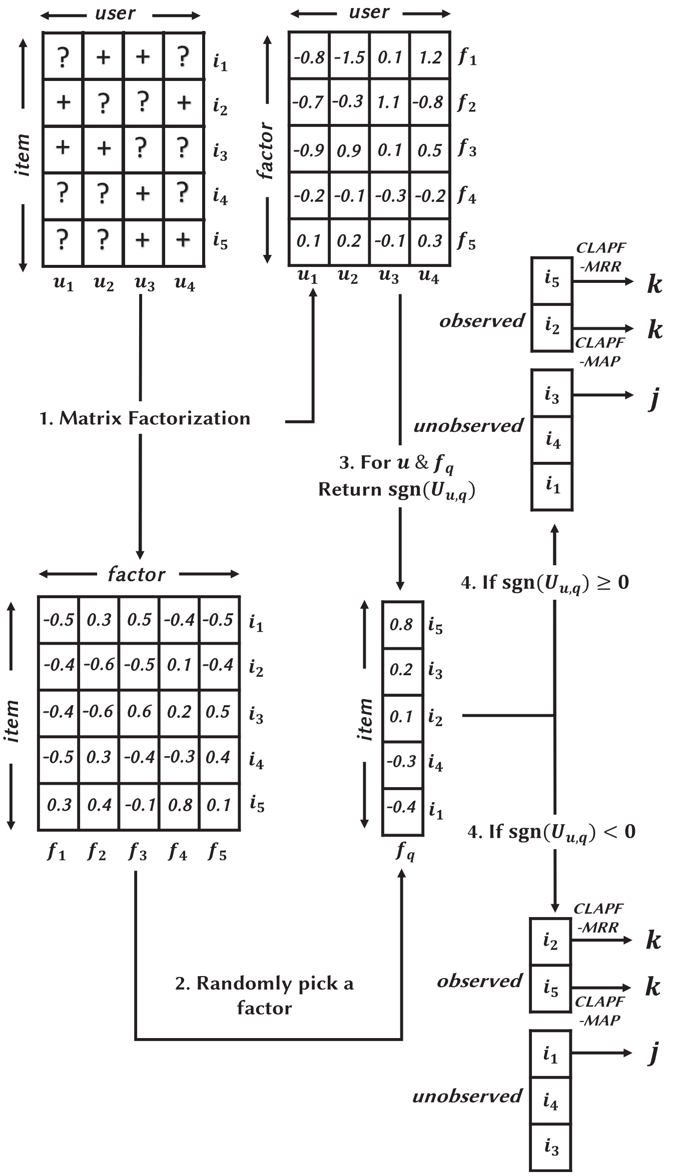


Fig. 1. Illustration of double sampling strategy.

the number of items, to make DSS can be used in a comparable time to uniform sampling. For simplicity, we abbreviate CLAPF with DSS algorithm as CLAPF+.

6 Experimental Evaluation

In this section, we mainly evaluate CLAPF and CLAPF+ on six real-world datasets from different perspectives. Specifically, we first describe the datasets, baselines, and parameter settings used in the experiments. Then, we compare the recommendation performance of CLAPF and CLAPF+ with baseline approaches in terms of many evaluation metrics. Finally, we analyze the effectiveness of the proposed DSS sampler in CLAPF+ on learning convergence.

6.1 Datasets

We use six real-world datasets in our empirical studies, including three general datasets, i.e., MovieLens100K,[1] MovieLens1M, UserTag, and three large datasets, i.e., MovieLens20M, Flixter,[2] Netflix.[3] Specifically, MovieLens100K (ML100K) contains 100,000 ratings annotated by 943 users on 1,682 movies; MovieLens1M (ML1M) contains 1,000,209 ratings annotated by 6,040 users on 3,952 movies; UserTag contains 246,436 user-tag pairs from 3,000 users and 2,000 tags; MovieLens20M (ML20M) contains 20,000,263 ratings annotated by 138,493 users on 26,744 items; Flixter contains 8,196,077 ratings annotated by 147,612 users on 48,794 items; and Netflix contains 99,072,112 ratings annotated by 480,189 users on 17,770 items. We use "item" to denote movie (for ML100K, ML1M, ML20M, Flixter, and Netflix) or tag (for UserTag). For ML100K, ML1M, ML20M, Flixter, and Netflix, we take a pre-processing step mentioned in [56], which only keeps the ratings larger than 3 as the observed positive feedback (to simulate the implicit feedback). The final datasets are shown in Table 1.

TABLE 1
Description of the Experimental Datasets, Including the Number of Users (n), the Number of Items (m), the Number of User-Item Pairs in the Training Data ($\mathcal{P}$), the Number of User-Item Pairs in the Test Data ($\mathcal{P}^{te}$), and the Density of Each Data, i.e., $(\mathcal{P}+\mathcal{P}^{te})/n/m$

Datasets	n	m	$\mathcal{P}$	$\mathcal{P}^{te}$	$(\mathcal{P}+\mathcal{P}^{te})/n/m$
ML100K	943	1,682	27,688	27,687	3.49%
ML1M	6,040	3,952	287,641	287,640	2.41%
UserTag	3,000	3,000	123,218	123,218	4.11%
ML20M	138,493	26,744	579,741	580,093	0.11%
Flixter	147,612	48,794	318,353	318,671	0.02%
Netflix	480,189	17,770	4,556,347	4,558,506	0.23%

For all the six datasets, following the previous common training/test split strategy [10], [23], we randomly split half of the observed user-item pairs as training data, and the rest as test data; we then randomly take one user-item pair for each user from the training data to construct a validation set. We repeat the above procedure for five times, so we have five copies of training data and test data. The experimental results are averaged over the performance of those five copies of test data.

6.2 Evaluation Metrics

To study the recommendation performance, we adopt several metrics for distinct perspectives. As for top-k recommendation, we adopt commonly used top-k evaluation metrics, including $Precision$, $Recall$, $F1$, and $1-Call$. In addition, we also adopt ranking-aware evaluation metrics, including MAP, MRR, and $NDCG$.

6.3 Baselines and Parameter Settings

In order to demonstrate the effectiveness of our model, we compare it with several methods,[4] i.e., PopRank, RandomWalk, WMF, BPR, MPR, CLiMF, NeuMF, NeuPR, and DeepICF. We describe the baselines below:

1. https://grouplens.org/datasets/movielens/.

2. https://www.cs.ubc.ca/jamalim/datasets/.
3. http://www.netflix.com/.
4. We release the source code at https://github.com/bigdata-ustc/CLAPF-MPR.

- PopRank ranks the items according to their popularity in training data.
- Random Walk (denoted as RandomWalk) estimates the user's preference on an item via a weighted average of all reachable users' preferences on that item.
- WMF [6] is a typical pointwise method based on matrix factorization. It defines a weight distribution for each $(u, i) \in U \times I$, then employs a matrix factorization model to solve a regression problem by optimizing a square loss function.
- BPR [14] is a seminal pairwise method as mentioned above.
- MPR [16] is a state-of-the-art pairwise ranking method, which taps the connections among items with the multiple pairwise criteria.
- CLiMF [17] is a typical listwise method, which explores the optimization of Mean Reciprocal Rank (MRR).
- NeuMF [36] is a pairwise neural-based model, and is an advanced instantiation of NCF which consists both of generalized matrix factorization and multilayer perceptron to model latent feature interactions.
- NeuPR [37] is a pairwise neural-based model and is a more efficient deep CF model without negative sampling.
- DeepICF [39] is a typical pointwise neural-based model.

We use "CLAPF (-MAP, -MRR)" to represent CLAPF (-MAP, -MRR) with the uniform sampler, and "CLAPF+ (-MAP, -MRR)" to represent CLAPF (-MAP, -MRR) with our DSS sampler. For a fair comparison, all the matrix factorization based CF methods are implemented in the same code framework. For all CLAPF methods, the regularization parameters are searched as $\alpha_u = \alpha_v = \beta_v \in \{0.001, 0.002, 0.01, 0.02, 0.1\}$, and the tradeoff parameter $\lambda \in \{0.0, 0.1, \ldots, 1.0\}$, and the iteration number is chosen from $T \in \{1\,000, 10\,000, 100\,000\}$. The $NDCG@5$ performance on the validation data is used to select all the best parameters of CLAPF. The learning rate is chosen from $\gamma \in \{0.0001, 0.001, 0.01\}$ and the number of latent is fixed as $d = 20$ in BPR, MPR and CLAPF, and the initialization value of U_u, V_i, b_i are set the same as in [57]. For RandomWalk, the walk length is searched from $\{20, 40, 60, 80\}$, and the reachable threshold is searched from $\{2, 5, 10, 20\}$, as showing huge time cost on large datasets, we make some tradeoffs between efficiency and effectiveness. For WMF, the number of latent is chosen from $\{10, 20\}$, the weighted parameter is searched from $\{10, 20, 40, 100\}$, the learning rate is chosen from $\{0.0001, 0.001, 0.01\}$, and the regularization parameters are searched from $\{0.001, 0.01, 0.1\}$. For MPR, the tradeoff parameter is searched from $\{0.0, 0.1, \ldots, 1.0\}$. For CLiMF, regularization parameters are searched from $\{0.001, 0.01, 0.1\}$, the latent dimensionality is fixed as 20, and the learning rate is searched from $\{0.0001, 0.001, 0.01\}$. For each deep model, we implement it using TensorFlow, the embedding size is searched from $\{4, 8, 16, 32\}$, the learning rate is chosen from $\{0.0001, 0.001, 0.01\}$, and we keep the structure as reported in [36], [37], [39] containing four layers in MLP component. For the above and other model parameters, the optimal values are tuned according to $NDCG@5$ performance on validation data. Noted that, unlike the evaluate protocol in [36], where only 100 unobserved items are sampled to evaluate the final ranking performance, we rank all the unobserved items based on the predicted scores as adopted in common recommender systems.

6.4 Summary of Experimental Results

6.4.1 Main Results

The experimental results and the training time of all algorithms on six datasets are shown in Table 2, and the numbers in boldface are the best results (with DSS sampler or not). In addition, top-k ($k = 3, 5, 10, 15, 20$) recommendation performance is shown in terms of two most concerned metrics, $Recall$ and $NDCG$, in Fig. 2. We use "$-$ " to denote the cases that do not produce results within 200 hours. From the table and the figure, we have the following observations:

- CLAPF (-MAP, -MRR) and CLAPF+ (-MAP, -MRR) perform better than the other baselines in terms of $Precision@k$, $Recall@k$, $F1@k$, $1 - Call@k$, and $NDCG@k$ on six datasets, which shows that our proposed algorithms can recommend better top-k items for users. Besides, CLiMF is inferior to the pairwise ranking methods, indicating that the typical listwise method works on datasets where only a few historical items are given to the individual user as in [17]. Moreover, we observe that neural-based models are not superior to matrix factorization based models on some datasets, which mainly because deep models are possibly to overfit under various conditions of data sparseness.
- CLAPF (-MAP, -MRR) and CLAPF+ (-MAP, -MRR) perform better than the other baselines in terms of $NDCG$, MAP, and MRR on six datasets, which proves that our proposed algorithms really address the ranking problem by optimizing the observed item pairs, and propose a more accurate rank-biased list for users. More precisely, CLAPF-MAP overall performs better than CLAPF-MRR in terms of MAP with DSS sampler or not, while CLAPF-MRR overall performs better than CLAPF-MAP in terms of MRR with DSS sampler or not, confirming our proposed algorithms are optimizing what they intend to optimize.
- As to the training time, CLAPF and CLAPF+ are comparable to BPR in terms of efficiency even for large datasets, far faster than CLiMF, which indicates that our proposed algorithm does not increase the computation complexity. To some extent, our proposed DSS sampler works efficiently in CLAPF framework, indicated that CLAPF is a basic method with extensive applicability.

6.4.2 Impact of Tradeoff Parameters

To have a deep understanding of the objective functions in CLAPF, we adjust the tradeoff parameter as $\lambda \in \{0.0, 0.1, \ldots, 1.0\}$ and show the results in terms of $Prec@5$, $Recall@5$, $F1@5$, $NDCG@5$, MAP, and MRR in Fig. 3. It is worth mentioning that, since CLAPF-MAP and CLAPF-MRR respectively have two-pair objective functions (one is of listwise and the other is of pairwise), we can remove one of two pairs to study their performance on datasets by setting

TABLE 2
Performance Comparisons of CLAPF (-MAP, -MRR) and Baselines on ML100K, ML1M, UserTag, ML20M, Flixter, and Netflix

Dataset	Method	$Prec@5$	$Recall@5$	$F1@5$	$1-Call@5$	$NDCG@5$	MAP	MRR	time
ML100K	PopRank	0.272±0.009	0.054±0.002	0.082±0.003	0.652±0.020	0.291±0.007	0.140±0.001	0.443±0.005	136s
	RandomWalk	0.298±0.006	0.061±0.001	0.089±0.001	0.683±0.010	0.316±0.004	0.149±0.002	0.455±0.006	162s
	WMF	0.359±0.008	0.086±0.004	0.121±0.005	0.792±0.019	0.375±0.009	0.239±0.002	0.563±0.017	1189s
	BPR	0.364±0.006	0.094±0.001	0.130±0.002	0.813±0.002	0.379±0.010	0.247±0.002	0.587±0.012	256s
	MPR	0.372±0.004	0.098±0.002	0.135±0.002	0.826±0.006	0.384±0.009	0.254±0.003	0.598±0.010	485s
	CLiMF	0.278±0.003	0.055±0.001	0.084±0.003	0.667±0.022	0.301±0.005	0.162±0.009	0.499±0.006	521s
	NeuMF	0.365±0.009	0.094±0.005	0.130±0.005	0.806±0.018	0.379±0.009	0.251±0.002	0.590±0.012	753s
	NeuPR	0.337±0.003	0.082±0.003	0.115±0.002	0.784±0.006	0.347±0.005	0.220±0.003	0.545±0.016	685s
	DeepICF	0.355±0.003	0.090±0.002	0.122±0.003	0.791±0.009	0.368±0.005	0.247±0.002	0.576±0.010	1096s
	CLAPF ($\lambda=0.4$) -MAP	**0.432±0.005**	**0.120±0.003**	**0.163±0.003**	0.858±0.011	0.454±0.006	**0.294±0.002**	0.664±0.010	264s
	CLAPF ($\lambda=0.2$) -MRR	0.395±0.004	0.109±0.002	0.146±0.001	0.850±0.012	0.417±0.009	0.270±0.002	**0.669±0.009**	266s
	CLAPF+ ($\lambda=0.4$) -MAP	**0.432±0.003**	0.110±0.001	0.159±0.002	**0.869±0.027**	**0.456±0.008**	0.289±0.001	0.655±0.020	282s
	CLAPF+ ($\lambda=0.2$) -MRR	0.410±0.004	0.102±0.002	0.142±0.002	0.851±0.019	0.439±0.010	0.264±0.003	**0.669±0.007**	286s
ML1M	PopRank	0.282±0.002	0.040±0.001	0.063±0.001	0.667±0.001	0.293±0.001	0.151±0.001	0.444±0.002	1174s
	RandomWalk	0.296±0.002	0.044±0.001	0.068±0.001	0.688±0.001	0.308±0.001	0.151±0.001	0.459±0.002	7633s
	WMF	0.441±0.004	0.074±0.004	0.113±0.001	0.857±0.003	0.452±0.005	0.249±0.001	0.639±0.001	10654s
	BPR	0.438±0.001	0.073±0.001	0.112±0.001	0.850±0.009	0.452±0.002	0.255±0.001	0.648±0.002	5688s
	MPR	0.440±0.002	0.075±0.001	0.117±0.001	0.849±0.005	0.460±0.002	0.262±0.001	0.655±0.002	9736s
	CLiMF	0.270±0.002	0.039±0.001	0.061±0.001	0.664±0.006	0.277±0.002	0.139±0.001	0.464±0.002	10105s
	NeuMF	0.399±0.010	0.066±0.002	0.101±0.003	0.818±0.011	0.415±0.010	0.224±0.001	0.593±0.002	8249s
	NeuPR	0.349±0.009	0.053±0.004	0.083±0.005	0.763±0.010	0.362±0.009	0.202±0.001	0.554±0.003	7697s
	DeepICF	0.387±0.006	0.064±0.001	0.096±0.003	0.799±0.002	0.411±0.005	0.217±0.001	0.583±0.004	14014s
	CLAPF ($\lambda=0.4$) -MAP	0.474±0.002	0.081±0.001	0.123±0.001	**0.877±0.009**	0.490±0.003	0.265±0.001	0.686±0.003	5747s
	CLAPF ($\lambda=0.8$) -MRR	0.478±0.002	0.082±0.001	0.120±0.001	0.864±0.003	0.491±0.002	0.261±0.001	**0.692±0.006**	5724s
	CLAPF+ ($\lambda=0.4$) -MAP	**0.487±0.002**	**0.087±0.001**	**0.133±0.001**	0.876±0.004	**0.508±0.002**	**0.269±0.001**	0.674±0.003	6120s
	CLAPF+ ($\lambda=0.8$) -MRR	0.470±0.002	0.079±0.001	0.124±0.001	0.873±0.003	0.481±0.003	0.261±0.001	0.678±0.004	6213s
UserTag	PopRank	0.264±0.001	0.037±0.001	0.061±0.001	0.522±0.006	0.263±0.001	0.125±0.001	0.396±0.003	543s
	RandomWalk	0.271±0.004	0.038±0.001	0.064±0.001	0.533±0.006	0.277±0.001	0.126±0.001	0.398±0.003	4035s
	WMF	0.273±0.004	0.041±0.001	0.064±0.001	0.570±0.004	0.280±0.004	0.134±0.001	0.399±0.006	4365s
	BPR	0.287±0.003	0.042±0.001	0.066±0.001	0.572±0.006	0.283±0.003	0.141±0.001	0.402±0.006	1826s
	MPR	0.282±0.003	0.045±0.001	0.067±0.001	0.590±0.005	0.280±0.003	0.151±0.001	0.411±0.005	3144s
	CLiMF	0.263±0.002	0.039±0.001	0.063±0.001	0.540±0.008	0.270±0.003	0.145±0.001	0.422±0.005	6428s
	NeuMF	0.294±0.008	0.046±0.001	0.073±0.001	0.605±0.010	0.302±0.009	0.157±0.001	0.440±0.005	6759s
	NeuPR	0.269±0.007	0.040±0.002	0.064±0.002	0.574±0.013	0.276±0.007	0.131±0.001	0.389±0.005	6173s
	DeepICF	0.285±0.005	0.041±0.001	0.067±0.002	0.582±0.012	0.293±0.005	0.150±0.009	0.429±0.006	8592s
	CLAPF ($\lambda=0.3$) -MAP	0.296±0.003	0.047±0.001	0.073±0.001	0.593±0.009	0.305±0.002	0.161±0.001	0.457±0.004	1907s
	CLAPF ($\lambda=0.2$) -MRR	0.267±0.002	0.041±0.001	0.064±0.001	0.578±0.008	0.276±0.003	0.149±0.001	0.460±0.006	1927s
	CLAPF+ ($\lambda=0.3$) -MAP	**0.307±0.002**	**0.049±0.001**	**0.080±0.001**	**0.639±0.009**	**0.322±0.003**	**0.166±0.001**	0.461±0.004	2128s
	CLAPF+ ($\lambda=0.2$) -MRR	0.291±0.002	0.047±0.001	0.069±0.001	0.584±0.008	0.306±0.002	0.160±0.001	**0.469±0.005**	2137s
ML20M	PopRank	0.063±0.001	0.083±0.001	0.059±0.001	0.256±0.001	0.089±0.001	0.035±0.001	0.096±0.001	2h
	RandomWalk	0.069±0.001	0.086±0.003	0.063±0.002	0.281±0.008	0.102±0.003	0.040±0.001	0.126±0.001	94h
	WMF	0.077±0.001	0.096±0.001	0.071±0.001	0.305±0.002	0.104±0.001	0.045±0.001	0.189±0.001	48h
	BPR	0.089±0.001	0.114±0.003	0.083±0.002	0.346±0.005	0.121±0.003	0.054±0.001	0.204±0.001	29h
	MPR	0.093±0.001	0.116±0.002	0.087±0.002	0.352±0.003	0.126±0.003	0.058±0.001	0.207±0.001	44h
	CLiMF	–	–	–	–	–	–	–	>200h
	NeuMF	0.080±0.001	0.101±0.002	0.074±0.002	0.327±0.008	0.110±0.003	0.048±0.001	0.192±0.001	71h
	NeuPR	0.075±0.001	0.090±0.002	0.067±0.002	0.299±0.005	0.104±0.003	0.044±0.001	0.183±0.001	67h
	DeepICF	0.077±0.001	0.095±0.002	0.071±0.002	0.315±0.007	0.106±0.002	0.046±0.001	0.188±0.001	102h
	CLAPF ($\lambda=0.3$) -MAP	0.112±0.001	**0.145±0.002**	**0.104±0.001**	0.411±0.004	**0.157±0.002**	0.080±0.001	0.235±0.001	33h
	CLAPF ($\lambda=0.9$) -MRR	0.105±0.001	0.140±0.002	0.097±0.001	0.392±0.004	0.146±0.001	0.073±0.001	**0.238±0.001**	35h
	CLAPF+ ($\lambda=0.3$) -MAP	**0.113±0.001**	0.141±0.002	0.102±0.001	**0.421±0.005**	0.153±0.002	**0.082±0.001**	0.232±0.001	33h
	CLAPF+ ($\lambda=0.9$) -MRR	0.109±0.001	0.133±0.002	0.095±0.001	0.401±0.004	0.139±0.001	0.069±0.001	0.228±0.001	35h
Flixter	PopRank	0.048±0.001	0.075±0.001	0.043±0.001	0.197±0.001	0.078±0.001	0.032±0.001	0.104±0.001	2h
	RandomWalk	3.0E-5±4.3E-6	2.0E-5±0.001	1.7E-5±8.8E-6	1.4E-4±2.5E-5	4.9E-5±8.8E-6	2.3E-4±1.8E-6	8.2E-4±4.4E-6	108h
	WMF	0.058±0.001	0.102±0.001	0.055±0.001	0.233±0.001	0.100±0.001	0.039±0.001	0.167±0.001	20h
	BPR	0.062±0.001	0.100±0.001	0.056±0.001	0.252±0.002	0.107±0.001	0.043±0.001	0.175±0.001	12h
	MPR	0.064±0.001	0.107±0.001	0.058±0.001	0.266±0.002	0.110±0.001	0.049±0.001	0.192±0.001	22h
	CLiMF	–	–	–	–	–	–	–	>200h
	NeuMF	0.062±0.001	0.093±0.002	0.056±0.001	0.260±0.003	0.109±0.001	0.045±0.001	0.185±0.001	45h
	NeuPR	0.052±0.001	0.085±0.002	0.050±0.001	0.221±0.002	0.088±0.001	0.036±0.001	0.163±0.001	38h
	DeepICF	0.059±0.001	0.091±0.002	0.053±0.001	0.247±0.003	0.100±0.001	0.040±0.001	0.175±0.001	62h
	CLAPF ($\lambda=0.3$) -MAP	0.064±0.001	0.104±0.002	0.057±0.001	0.264±0.001	0.110±0.001	0.050±0.001	0.194±0.001	14h
	CLAPF ($\lambda=0.2$) -MRR	**0.073±0.001**	**0.121±0.005**	**0.069±0.002**	**0.284±0.003**	**0.119±0.001**	0.053±0.001	**0.207±0.001**	16h
	CLAPF+ ($\lambda=0.3$) -MAP	0.065±0.001	0.108±0.002	0.058±0.001	0.268±0.002	0.110±0.001	**0.055±0.001**	0.196±0.001	15h
	CLAPF+ ($\lambda=0.2$) -MRR	0.071±0.001	0.117±0.002	0.065±0.001	0.277±0.002	0.108±0.001	0.053±0.001	0.201±0.001	16h
Netflix	PopRank	0.048±0.001	0.032±0.001	0.030±0.001	0.197±0.001	0.052±0.001	0.031±0.001	0.087±0.001	3h
	RandomWalk	–	–	–	–	–	–	–	>200h
	WMF	0.101±0.001	0.068±0.001	0.063±0.001	0.361±0.002	0.117±0.001	0.053±0.001	0.181±0.001	89h
	BPR	0.109±0.001	0.076±0.001	0.069±0.001	0.388±0.001	0.126±0.001	0.060±0.001	0.199±0.001	64h
	MPR	0.114±0.001	0.080±0.001	0.073±0.002	0.397±0.002	0.132±0.004	0.063±0.001	0.205±0.001	103h
	CLiMF	–	–	–	–	–	–	–	>200h
	NeuMF	0.098±0.001	0.070±0.001	0.066±0.001	0.355±0.002	0.120±0.001	0.054±0.001	0.182±0.001	122h
	NeuPR	0.088±0.001	0.063±0.001	0.055±0.001	0.339±0.002	0.108±0.001	0.050±0.001	0.171±0.001	104h
	DeepICF	0.095±0.001	0.068±0.001	0.062±0.001	0.345±0.001	0.114±0.001	0.052±0.001	0.176±0.001	165h
	CLAPF ($\lambda=0.3$) -MAP	0.134±0.001	**0.090±0.001**	0.085±0.001	0.450±0.002	0.158±0.001	0.075±0.001	0.220±0.001	72h
	CLAPF ($\lambda=0.2$) -MRR	0.119±0.001	0.083±0.001	0.076±0.001	0.433±0.002	0.139±0.001	0.068±0.001	0.213±0.001	75h
	CLAPF+ ($\lambda=0.3$) -MAP	**0.139±0.001**	0.089±0.001	**0.087±0.001**	**0.453±0.001**	**0.162±0.001**	**0.081±0.001**	0.228±0.001	71h
	CLAPF+ ($\lambda=0.2$) -MRR	0.122±0.001	0.085±0.001	0.080±0.001	0.446±0.002	0.148±0.001	0.073±0.001	**0.232±0.001**	75h

Numbers in boldface are the best results.

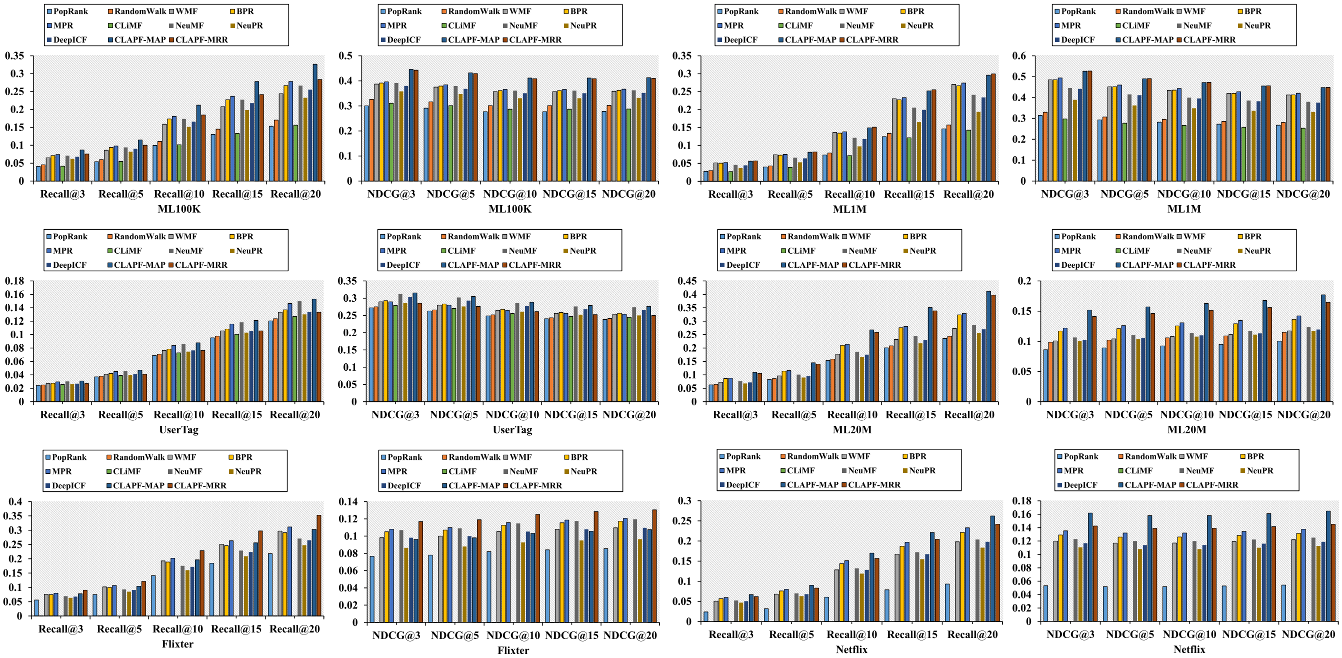


Fig. 2. Top-k ($k = 3, 5, 10, 15, 20$) recommendation performance of CLAPF (-MAP, -MRR) and baselines on ML100K, ML1M, UserTag, ML20M, Flixter, and Netflix.

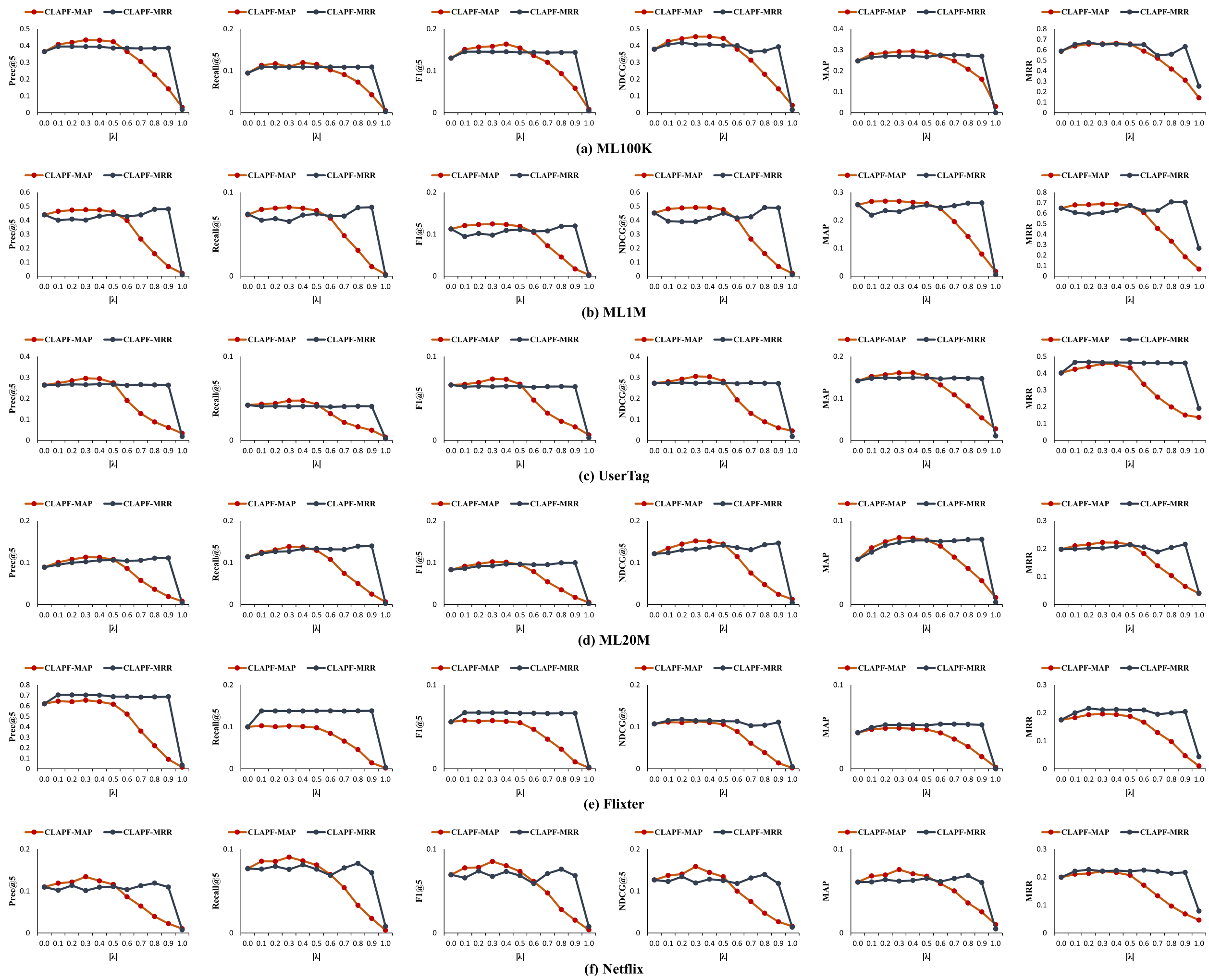


Fig. 3. Recommendation performance of CLAPF (-MAP, -MRR) with different tradeoff parameters (from top row to bottom row: ML100K, ML1M, UserTag, ML20M, Flixter, and Netflix).

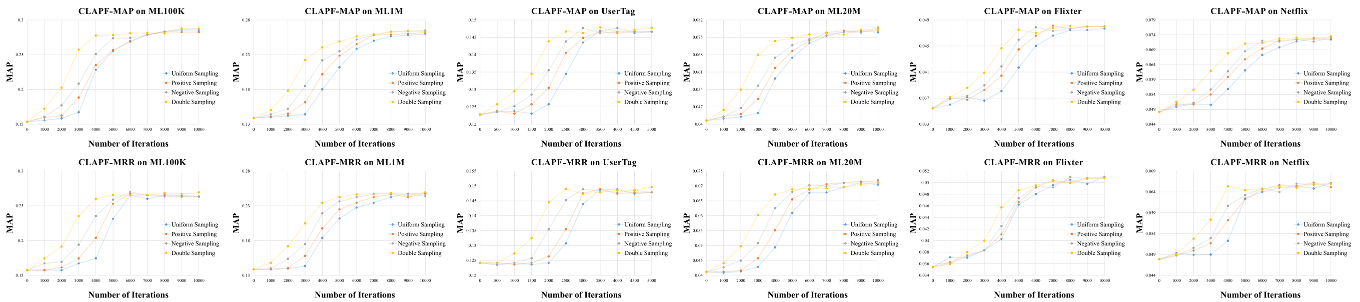


Fig. 4. The learning convergence of CLAPF with different samplers on training iterations (from left column to right column: ML100K, ML1M, UserTag, ML20M, Flixter, and Netflix).

the tradeoff parameter $\lambda = 0$ or 1. From the figure, we can see that using different tradeoff parameters effects the recommendation performance of CLAPF, but there is some difference between CLAPF-MAP and CLAPF-MRR. CLAPF-MAP responds more gently to changes in parameters, while CLAPF-MRR responds very strongly to changes in certain parameters. Specifically, in terms of some metrics, like $F1@5$, $NDCG@5$, and MAP, a flexible trade-off parameter overall help CLAPF-MAP get better performance than CLAPF-MRR, which indicates that CLAPF-MAP has more potential in top-k or rank-aware recommendation, and our smoothing approach preserves aforementioned good properties of MAP measure. Notice that when $\lambda = 0$, CLAPF reduces to BPR.

6.4.3 Convergence Analysis

We also conduct supplementation experiments on six datasets to further demonstrate the effectiveness of our proposed DSS sampler for CLAPF in Fig. 4. As DSS not only samples the negative item (item j in CLAPF) from the unobserved items, but also samples the positive item (item k in CLAPF) from the observed items each time, we remove one or both of the sampling functions in DSS to build three comparative sampling strategies:

- Uniform Sampling picks the positive items (the item k and the item i in CLAPF) and the negative item (the item j in CLAPF) from the observed items and unobserved items with equal probabilities each time.
- Positive Sampling picks the positive item (the item k) in the same way as DSS, and picks the other items (the item j and the item i) in the same way as Uniform Sampling each time.
- Negative Sampling picks the negative item (the item j) in the same way as DSS, and picks the other items (the item k and the item i) in the same way as Uniform Sampling each time.

Fig. 4 shows that DSS sampler helps converge much faster than the other samplers in terms of MAP, which indicates that DSS is a more effective sampler for CLAPF by drawing informative positive and negative items in a fine-grained way. In addition, all non-uniform samplers help converge faster than Uniform Sampling. Meanwhile, Positive Sampling does not perform as well as Negative Sampling, which mainly because the observed items are much fewer than the unobserved items. Moreover, DSS sampler helps converge faster at early iterations, which mainly because such fine-grain utilizing of rank information on positive and negative items is significant for learning the unstable model. Finally, all algorithms almost converge after some iterations, then fluctuate in a tiny range around.

All the analyses show that our CLAPF algorithm and DSS sampler are indeed superior to the previous methods for implicit feedback problem.

7 Conclusion

In summary, this paper presents a new hybrid ranking model, namely Collaborative List-and-Pairwise Filtering (CLAPF), for improving top-k recommendation from implicit feedback. We combined the objective functions of optimizing the two rank-biased metrics (MAP, MRR) with the pairwise objective function and formalized two instantiations of CLAPF called CLAPF-MAP and CLAPF-MRR. On the one hand, CLAPF brings the ranking measure into pairwise methods, which contributes a lot to the ranking problem in the top-k recommendation. On the other hand, CLAPF introduces pairwise thinking into listwise objective functions, which can exploit the hidden rich unobserved information and reduce the computation complexity. We conducted extensive experiments on six real-world datasets, and proved that our methods significantly outperform state-of-the-art implicit feedback recommenders regarding various evaluation metrics. The main contribution of our approach is to provide a new idea of utilizing rank-biased measures by combining the pairwise objective function on implicit feedback. The CLAPF framework is a hybrid listwise and pairwise model that helps us understand the ranking essence in top-k item recommendation, and is not limited to the instantiations in this paper. We encourage more smoothed listwise metrics to be optimized with our CLAPF framework.

Acknowledgments

This work was partially supported by grants from the National Key Research and Development Program of China (No. 2016YFB1000904), and the National Natural Science Foundation of China (No.s 61922073, 61672483 and U1605251). Qi Liu gratefully acknowledges the support of the Youth Innovation Promotion Association of CAS (No. 2014299).

References

[1] Q. Liu, Y. Ge, Z. Li, E. Chen, and H. Xiong, "Personalized travel package recommendation," in *Proc. 11th IEEE Int. Conf. Data Mining*, 2011, pp. 407–416.
[2] L. Wu, E. Chen, Q. Liu, L. Xu, T. Bao, and L. Zhang, "Leveraging tagging for neighborhood-aware probabilistic matrix factorization," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, 2012, pp. 1854–1858.

[3] L. Wu *et al*., "Modeling the evolution of users' preferences and social links in social networking services," *IEEE Trans. Knowl. Data Eng*., vol. 29, no. 6, pp. 1240–1253, Jun. 2017.

[4] M. Hou, L. Wu, E. Chen, Z. Li, V. W. Zheng, and Q. Liu, "Explainable fashion recommendation: A semantic attribute region guided approach," in *Proc. 28th Int. Joint Conf. Artif. Intell*., 2019, pp. 4681–4688.

[5] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Proc. Int. Conf. Neural Inf. Process. Syst*., 2008, pp. 1257–1264.

[6] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proc. 8th IEEE Int. Conf. Data Mining*, 2008, pp. 263–272.

[7] R. Pan and M. Scholz, "Mind the gaps: Weighting the unknown in large-scale one-class collaborative filtering," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2009, pp. 667–676.

[8] R. Pan *et al*., "One-class collaborative filtering," in *Proc. 8th IEEE Int. Conf. Data Mining*, 2008, pp. 502–511.

[9] J. Bobadilla, F. Ortega, A. Hernando, and J. Bernal, "A collaborative filtering approach to mitigate the new user cold start problem," *Knowl.-Based Syst*., vol. 26, pp. 225–238, 2012.

[10] W. Pan and L. Chen, "Group Bayesian personalized ranking with rich interactions for one-class collaborative filtering," *Neurocomputing*, vol. 207, pp. 501–510, 2016.

[11] Y. He, C. Wang, and C. Jiang, "Correlated matrix factorization for recommendation with implicit feedback," *IEEE Trans. Knowl. Data Eng*., vol. 31, no. 3, pp. 451–464, Mar. 2019.

[12] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2011, pp. 448–456.

[13] G. Zeno *et al*., "MyMediaLite: A free recommender system library," in *Proc. 5th ACM Conf. Recommender Syst*., 2011, pp. 305–308.

[14] S. Rendle *et al*., "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. 25th Conf. Uncertainty Artif. Intell*., 2009, pp. 452–461.

[15] L. Du, X. Li, and Y.-D. Shen, "User graph regularized pairwise matrix factorization for item recommendation," in *Proc. Int. Conf. Adv. Data Mining Appl*., 2011, pp. 372–385.

[16] R. Yu *et al*., "Multiple pairwise ranking with implicit feedback," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage*., 2018, pp. 1727–1730.

[17] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic, "CLIMF: Learning to maximize reciprocal rank with collaborative less-is-more filtering," in *Proc. 6th ACM Conf. Recommender Syst*., 2012, pp. 139–146.

[18] M. Deshpande and G. Karypis, "Item-based top-N recommendation algorithms," *ACM Trans. Inf. Syst*., vol. 22, no. 1, pp. 143–177, 2004.

[19] Q. Le and A. Smola, "Direct optimization of ranking measures," 2007, *arXiv:0704.3359*. [Online]. Available: https://arxiv.org/abs/0704.3359

[20] E. M. Voorhees, "The TREC-8 question answering track report," in *Proc. 8th Text Retrieval Conf*., 1999, pp. 77–82.

[21] Y. Liu, P. Zhao, A. Sun, and C. Miao, "A boosting algorithm for item recommendation with implicit feedback," in *Proc. 24th Int. Joint Conf. Artif. Intell*., 2015, pp. 1792–1798.

[22] V. Mahant, "Improving Top-N evaluation of recommender systems," Master thesis, Texas State Univ., San Marcos, Texas, 2016. [Online]. Available: https://digital.library.txstate.edu/handle/10877/6309

[23] W. Pan and L. Chen, "GBPR: Group preference based Bayesian personalized ranking for one-class collaborative filtering," in *Proc. 23rd Int. Joint Conf. Artif. Intell*., 2013, pp. 2691–2697.

[24] S. Ouyang, L. Li, W. Pan, and Z. Ming, "Asymmetric Bayesian personalized ranking for one-class collaborative filtering," in *Proc. 5th ACM Conf. Recommender Syst*., 2019, pp. 373–377.

[25] W. Zhang, T. Chen, J. Wang, and Y. Yu, "Optimizing top-n collaborative filtering via dynamic negative item sampling," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2013, pp. 785–788.

[26] S. Rendle and C. Freudenthaler, "Improving pairwise learning for item recommendation from implicit feedback," in *Proc. 7th ACM Int. Conf. Web Search Data Mining*, 2014, pp. 273–282.

[27] M. Cheng *et al*., "Alpha-beta sampling for pairwise ranking in one-class collaborative filtering," in *Proc. 19th IEEE Int. Conf. Data Mining*, 2019, pp. 1000–1005.

[28] D. Song, D. A. Meyer, and D. Tao, "Efficient latent link recommendation in signed networks," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2015, pp. 1105–1114.

[29] Y. Guo, X. Wang, and C. Xu, "CroRank: Cross domain personalized transfer ranking for collaborative filtering," in *Proc. 15th IEEE Int. Conf. Data Mining Workshop*, 2015, pp. 1204–1212.

[30] Q. Liu, Z. Huang, Y. Yin, E. Chen, Y. Su, and G. Hu, "EKT: Exercise-aware knowledge tracing for student performance prediction," *IEEE Trans. Knowl. Data Eng*., to be published, doi: 10.1109/TKDE.2019.2924374.

[31] Q. Liu *et al*., "Exploiting cognitive structure for adaptive learning," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2009, pp. 627–635.

[32] H. Wu *et al*., "Deep technology tracing for high-tech companies," in *Proc. 19th IEEE Int. Conf. Data Mining*, 2019, pp. 1396–1401.

[33] Q. Liu, H. Wu, Y. Ye, H. Zhao, C. Liu, and D. Du, "Patent litigation prediction: A convolutional tensor factorization approach," in *Proc. 27th Int. Joint Conf. Artif. Intell*., 2018, pp. 5052–5059.

[34] Y. Ye, H. Zhu, T. Xu, F. Zhuang, R. Yu, and H. Xiong, "Identifying high potential talent: A neural network based dynamic social profiling approach," in *Proc. 19th IEEE Int. Conf. Data Mining*, 2019, pp. 718–727.

[35] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[36] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.

[37] B. Song, X. Yang, Y. Cao, and C. Xu, "Neural collaborative ranking," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage*., 2018, pp. 1353–1362.

[38] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2019, pp. 165–174.

[39] F. Xue, X. He, X. Wang, J. Xu, K. Liu, and R. Hong, "Deep item-based collaborative filtering for top-n recommendation," *ACM Trans. Inf. Syst*., vol. 37, no. 3, pp. 1–25, 2019.

[40] S. Rendle, W. Krichene, L. Zhang, and J. Anderson, "Neural collaborative filtering vs. matrix factorization revisited," 2020, *arXiv: 2005.09683*. [Online]. Available: https://arxiv.org/abs/2005.09683

[41] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Trans. Inf. Syst*., vol. 22, no. 1, pp. 89–115, 2004.

[42] N. N. Liu, M. Zhao, and Q. Yang, "Probabilistic latent preference analysis for collaborative filtering," in *Proc. 18th ACM Conf. Inf. Knowl. Manage*., 2009, pp. 759–766.

[43] N. N. Liu and Q. Yang, "EigenRank: A ranking-oriented approach to collaborative filtering," in *Proc. 31st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2008, pp. 83–90.

[44] Y. Koren and J. Sill, "OrdRec: An ordinal model for predicting personalized item rating distributions," in *Proc. 5th ACM Conf. Recommender Syst*., 2011, pp. 117–124.

[45] S.-H. Yang, B. Long, A. Smola, H. Zha, and Z. Zheng, "Collaborative competitive filtering: Learning recommender using context of user choice," in *Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2011, pp. 295–304.

[46] S. Niu, J. Guo, Y. Lan, and X. Cheng, "Top-k learning to rank: Labeling, ranking and evaluation," in *Proc. 35th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2012, pp. 751–760.

[47] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer, "Local collaborative ranking," in *Proc. 23rd Int. Conf. World Wide Web*, 2014, pp. 85–96.

[48] J. Lee, W.-S. Hwang, J. Parc, Y. Lee, S.-W. Kim, and D. Lee, "*l*-Injection: Toward effective collaborative filtering using uninteresting items," *IEEE Trans. Knowl. Data Eng*., vol. 31, no. 1, pp. 3–16, Jan. 2019.

[49] S. Wang, S. Huang, T.-Y. Liu, J. Ma, Z. Chen, and J. Veijalainen, "Ranking-oriented collaborative filtering: A listwise approach," *ACM Trans. Inf. Syst*., vol. 35, no. 2, pp. 1–28, 2016.

[50] J. Bu, X. Shen, B. Xu, C. Chen, X. He, and D. Cai, "Improving collaborative recommendation via user-item subgroups," *IEEE Trans. Knowl. Data Eng*., vol. 28, no. 9, pp. 2363–2375, Sep. 2016.

[51] C.-C. Hsu, M.-Y. Yeh, and S.-de Lin, "A general framework for implicit and explicit social recommendation," *IEEE Trans. Knowl. Data Eng*., vol. 30, no. 12, pp. 2228–2241, Dec. 2018.

[52] M. Weimer, A Karatzoglou, and Q. V. Le, "COFI RANK-Maximum margin matrix factorization for collaborative ranking," in *Proc. Int. Conf. Neural Inf. Process. Syst*., 2008, pp. 1593–1600.

[53] S. Balakrishnan and S. Chopra, "Collaborative ranking," in *Proc. 5th ACM Int. Conf. Web Search Data Mining*, 2012, pp. 143–152.

[54] Y. Shi, M. Larson, and A. Hanjalic, "List-wise learning to rank with matrix factorization for collaborative filtering," in *Proc. 4th ACM Conf. Recommender Syst*., 2010, pp. 269–272.

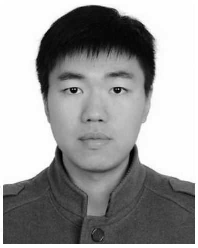
[55] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver, "TFMAP: Optimizing MAP for top-n context-aware recommendation," in *Proc. 35th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2012, pp. 155–164.

[56] V. Sindhwani, S. S. Bucak, J. Hu, and A. Mojsilovic, "One-class matrix completion with low-density factorizations," in *Proc. 10th IEEE Int. Conf. Data Mining*, 2010, pp. 1055–1060.

[57] W. Pan, E. W. Xiang, and Q. Yang, "Transfer learning in collaborative filtering with uncertain ratings," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012, pp. 662–668.

Runlong Yu received the BEng degree in computer science and technology from the University of Science and Technology of China (USTC), Hefei, China, in 2017. He is currently working toward the PhD degree in the Anhui Province Key Laboratory of Big Data Analysis and Application (BDAA), School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, China. His research interests include recommender systems, data mining, machine learning, and computational intelligence. He was the recipient of KDD CUP' 19 Special Award. He was also the recipient of China National Scholarship, in 2019.

Qi Liu (Member, IEEE) received the PhD degree from the University of Science and Technology of China (USTC), Hefei, China, in 2013. He is currently a professor with the Anhui Province Key Laboratory of Big Data Analysis and Application (BDAA), School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, China. His research interests include data mining, machine learning, and recommender systems. He was the recipient of KDD' 18 Best Student Paper Award and ICDM' 11 Best Research Paper Award. He was also the recipient of China Outstanding Youth Science Foundation, in 2019.



Yuyang Ye received the matser's degree in computer science and technology from the University of Science and Technology of China (USTC), Hefei, China, in 2020. He is currently working toward the PhD degree in the Management Science and Information Systems Department, Rutgers University, Newark, New Jersey. His research interests include representation learning, data mining, and recommender systems. He was the recipient of KDD CUP' 19 Special Award.

Mingyue Cheng received the BArts degree in business administration from the Hefei University of Technology, Hefei, China, in 2017. He is currently working toward the PhD degree in the Anhui Province Key Laboratory of Big Data Analysis and Application (BDAA), School of Data Science, University of Science and Technology of China (USTC), Hefei, China. His research interests include positive unlabeled learning, and recommender systems.

Enhong Chen (Senior Member, IEEE) received the PhD degree from the University of Science and Technology of China (USTC), Hefei, China, in 1996. He is currently a director (professor) of the Anhui Province Key Laboratory of Big Data Analysis and Application (BDAA), University of Science and Technology of China (USTC), Hefei, China. He is an executive dean of the School of Data Science, USTC, and vice dean of the School of Computer Science and Technology, USTC. He is a CCF fellow. His research interests include data mining, machine learning, and recommender systems. He has published more than 200 refereed international conference and journal papers. He was the recipient of KDD' 08 Best Application Paper Award and ICDM' 11 Best Research Paper Award.

Jianhui Ma is currently a lecturer with the Anhui Province Key Laboratory of Big Data Analysis and Application (BDAA), School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, China. His research interests include data mining, machine learning, recommender systems, and blockchain. He has published several papers in refereed conference proceedings, such as ICDM' 19.