Neural Collaborative Preference Learning with Pairwise Comparisons

Zhaopeng Li, Qianqian Xu*, *Senior Member, IEEE,* Yangbangyan Jiang, Ke Ma, *Member, IEEE,* Xiaochun Cao, *Senior Member, IEEE,* and Qingming Huang*, *Fellow, IEEE*

***Abstract*—Collaborative Ranking (CR), as an effective recommendation framework, has attracted increasing attention in recent years. Most CR methods simply adopt the inner product between user/item embeddings as the rating score function, with an assumption that the interacted items are preferred to non-interacted ones. However, such fixed score functions and assumptions might not be sufficient to capture the real preference ranking list from the complicated interactions in real-world data. To alleviate this issue, we develop a novel collaborative ranking framework that learns an arbitrary utility function for item ranking with user preference concerned. In the core of our framework, a neural network is employed to model the utility function for personalized ranking with the strength of its nonlinearity. On top of this, we further adopt a pairwise ranking loss for user-item pairs to preserve the preference order of items for users. Besides, such a utility function enables us to generate the final top-K preference list in a much easier way. Finally, extensive experiments on four real-world datasets show the validity of our proposed method.**

***Index Terms*—Recommender system, collaborative ranking, neural networks, preference ranking.**

This work was supported in part by the National Key R&D Program of China under Grant 2018AAA0102003, in part by National Natural Science Foundation of China: 61620106009, U1636214, 61931008, 61836002, 61971016, U1936208, 61672514 and 61976202, in part by Key Research Program of Frontier Sciences, CAS: QYZDJ-SSW-SYS013, in part by Beijing Education Committee Cooperation Beijing Natural Science Foundation (No.KZ201910005007), in part by the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No. XDB28000000, in part by Beijing Natural Science Foundation (4182079), and in part by Youth Innovation Promotion Association CAS. *(*Corresponding author: Qianqian Xu, Qingming Huang.)*

Z. Li and Y. Jiang are with State Key Laboratory of Information Security (SKLOIS), Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China, and also with School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: lizhaopeng@iie.ac.cn, jiangyangbangyan@iie.ac.cn).

Q. Xu is with the Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China (e-mail: xuqianqian@ict.ac.cn).

K. Ma is with the School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China, and with the Artificial Intelligence Research Center, Peng Cheng Laboratory, Shenzhen 518055, China (e-mail: make@ucas.ac.cn).

X. Cao is with State Key Laboratory of Information Security (SKLOIS), Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China, also with Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518055, China, and also with School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: caoxiaochun@iie.ac.cn).

Q. Huang is with the School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 101408, China, also with the Key Laboratory of Big Data Mining and Knowledge Management (BDKM), University of Chinese Academy of Sciences, Beijing 101408, China, also with the Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China, and also with Peng Cheng Laboratory, Shenzhen 518055, China (e-mail: qmhuang@ucas.ac.cn).

I. Introduction

With the prevalence of web applications and online multimedia, people often feel troublesome to select what they really desire. To offer people some advice tailored to personal interests, Recommender System (RS) has been widely adopted. So far, RS has brought convenience to all aspects of our lives, $e.g.$, social recommendation [1]–[3], news recommendation [4], travel recommendation [5], and video recommendation [6]–[8].

A common framework for RS is Collaborative Filtering (CF) [9], [10], which only requires historical user-item interactions to model users' preference. Specifically, the underlying assumption of CF is that users with similar historical interactions tend to share similar preference patterns. Among the various CF algorithms [11]–[13], Matrix Factorization (MF), which achieved great success in the Netflix Prize, has been the most popular one [14]. In short, original MF methods decompose user-item rating matrix to vectors in a shared latent space and calculate the inner product between their vectors to predict ratings. Representative variants along this line include Bayesian Personalized Ranking [15], Weighted Regularized Matrix Factorization [16], [17], SVD-based models [18], [19], and Factorization Machines [20], [21]. Moreover, the above methods still apply the inner product as score function, and combines the multiplication of latent vectors linearly, which might be insufficient to model the complicated interactions. To address this issue, many research efforts are devoted to modeling the nonlinear relations between users and items, including autoencoder-based models [22], [23], multi-layer perceptron-based models [24]–[26], convolutional neural networks-based models [27], graph-based models [28], and translation-based models [29].

Unfortunately, many CF methods either focus on predicting the score of unobserved user-item interaction, or classifying the unseen items based on the assumption that non-interacted items are negative, which ignores the relative ranking between items for each user. For example, AutoRec [22] adopts autoencoders to explicitly predict the score of unseen user-item pairs, while NCF [24] uses point-wise loss to classify positive and negative user-item interactions with implicit feedback. Recent CF models ($e.g.$, NeuBPR [26], ONCF [27], and NCGF [28].) take BPR's assumption as an improvement, $i.e.$, the non-observed items are not necessarily negative but just less preferred to the interacted ones [15], with a pairwise BPR loss to predict the relative orders between interactions. However, they still suffer from the calibration problem [30].

Specifically, simply considering the user would prefer his observed items to non-observed ones is harmful to the item ranking performance, as the user might be dissatisfied with some items he interacted with, and there might exist some items that would be liked but just have not been found by the user. In order to accurately predict the top-K recommended items, many researchers turn to the Collaborative Ranking (CR) [31] framework. Recent CR variants [32]–[35] focus on improving top-K recommendations by utilizing the naturally existing item ranking information and optimizing the ranking of items directly. Under such a CR framework, users' preference order for different items could be modeled more thoroughly. However, most existing CR methods either depend on a fixed handcraft score function, or model the interactions in a linear manner, thus may lack the ability to learn underlying relationships from complex data. Although NCR [36] adopts neural networks for item ranking, it suffers from two aspects. On one hand, NCR still takes the BPR's assumption, which might harm the ranking performance. On the other hand, the model could only predict the possibility that user i prefers item j to item k. Consequently, it needs to scan the candidate item set K times to infer the final top-K item ranking list, which is time-consuming in practice.

From the perspective of item ranking, in this paper, we develop a new CR framework called Neural Collaborative Preference Learning (NCPL), which learns a utility function for personalized ranking from naturally existing pairwise comparisons between interacted items. Specifically, we employ deep neural networks to learn two kinds of embeddings, $i.e.$, shallow/deep embeddings. The shallow embeddings employ a linear operation to learn simple relations, while the deep embeddings model the complex relations with its nonlinearity. Equipped with the two embeddings, we could obtain an arbitrary continuous real utility function for user-item interactions. Taking into account the user-specific item ranking problem, we further adopt a pairwise ranking loss, which is optimized for a personalized ranking list directly. Consequently, the learned utility function could predict a rank-preserved score for top-K recommendations. More specifically, our main contributions are listed as follows:

- We propose a novel deep learning framework for collaborative preference learning with the pairwise comparisons between interacted items for each user, which combines the shallow and deep embeddings to learn a real utility function for personalized ranking.
- In the core of the framework lies a pairwise ranking loss to optimize the score difference margin of user-item pairs, which enables our model to predict a rank-preserved score for each user-item pair.
- The experimental results on four real-world datasets show the superior performance of our methods compared with other algorithms.

The rest of this paper is organized as follows. We discuss related work in Section II. In Section III, we present a CR framework that learns a utility function for personalized ranking. We conduct comprehensive experiments to show the effectiveness of our design in Section IV. Concluding remarks are made in Section V.

II. Related Work

A. Collaborative Filtering

Matrix Factorization (MF) has been one of the most popular Collaborative Filtering (CF) algorithms in the past few years. Traditional MF models [37] decompose the user-item interaction matrix ($e.g.$, the user-item rating matrix) into the product of the user feature matrix and item feature matrix. In other words, they adopt the inner product of user feature vector and item feature vectors as a score function for user-item interaction. To alleviate overfitting and improve the generalization ability, Weighted Regularized Matrix Factorization (WRMF) [16] introduces a set of varying confidence levels to construct a regularized least-square optimization framework, where interacted items tend to have higher weights than non-interacted ones. Followed by SVD-based model [18], Factorization Machines [20], Max-margin MF [38], Nonnegative MF [39] and Bayesian logistic MF [40], [41]. Meanwhile, to alleviate the data sparsity and cold start problem, some researchers integrate rich side information [42] into CF as additional valuable features, and use them to estimate a users' preference [5], [7]. Although using such additional information does improve the performance, it might limit the wide application of the algorithm, as it needs additional computing power and storage [43], and it is hard to guarantee that information would be available in other situations. Thereby, in this paper we aim to accurately learn users' preference order from the user-item historical interactions without any additional information.

Most of the above methods use an inner product as the score function. However, recent studies [44] figure out that the inner product does not satisfy the triangle inequality, and may limit the expressiveness of MF. To address this issue, many methods are proposed to model the nonlinear relations between users and items. For example, AutoReC [22] designs a autoencoder framework as score function, and MultiVAE [23] extends variational autoencoders to CF. Besides, NCF [24] proposes a general CF framework based on neural networks, which first integrates a linear kernel (generalized matrix factorization) and a nonlinear kernel (multi-layer perceptron) as the score functions of user-item interactions. Similarly, NeuPR [26] incorporates generalized matrix factorization with BPR for image recommendation. NeuRec [25] combines a multilayer perceptron and a shallow latent factor to learn dense vector representations for users and items, with an inner product as the score function. Moreover, TransRec [29] embeds items and users as points and translation vectors in the same latent space, with a nearest-neighbor search to predict users' sequential behavior. ONCF [27] leverages CNN as a score function to learn high order relations from the outer product between user and item embeddings. NGCF [28] considers the user-item graph structure, propagates the collaborative signal on it by embedding propagation layer, and uses the inner product to model the users' preference towards the target item.

B. Collaborative Ranking

In order to estimate user preference for unseen items, most CF methods try to precisely predict the rating of unseen items,

or distinguish positive (preferred) items from negative (less-preferred) ones. Under such circumstances, these methods obtain good performance on point-wise error metrics ($e.g.$, mean-square error (MSE)) or classification metrics ($e.g.$, precision). However, the ranking performance of top-K recommendations for each user could not be guaranteed. In real-world recommender systems, the item ranking of recommendation list is vital to user experience and business benefit. To this end, researchers propose Collaborative Ranking (CR) [31] framework to recommend items in better order with users' preference concerned. Early CR methods simply add a pairwise ranking loss after their point-wise rating prediction counterparts. In other words, they still directly optimize the rating values. Based on the idea of maximum margin matrix factorization (MMMF), CoFiRank [45] presents the first effort to optimize the normalized discounted cumulative gain (NDCG) measure instead of rating measures, and becomes a common baseline for CR task. Borrowing the idea from learning to rank community for web search, Balakrishnan and Chopra [31] approximately optimize NDCG for the recommendation task. Volkovs and Zemel [46] present a method for CR which takes advantage of both neighborhood- and model-based CF approaches. Different from the above methods which assume that the rating matrix is low-rank, Lee et al. [47] assume that the rating matrix is globally high-rank but locally low-rank, and optimize a pairwise ranking loss. Hu and Li [48] consider ratings as ordinal rather than real values or categorical labels, and emphasize those items of higher rating scores to improve the performance for top-K recommendations. Besides, NCR [36] combines the BPR assumption with NCF [24] framework. LRML [49] learns latent relations with deep neural networks to model users and items in metric space.

However, the above CR methods still suffer from the calibration drawback, as it is probable that different users have different interpretations of the same score or score difference. To tackle this problem, some studies focus on exploiting pairwise comparisons more thoroughly, and directly optimizing the ranking metrics [32] without previous point-wise rating prediction in those early CR methods. In order to infer the ranking list of items for a specific user, Yi et al. [33] propose a crowd ranking framework based on matrix completion theory, and minimizes a convex objective using the hinge loss on a low-rank matrix. After that, Park et.al [34] provide a non-convex algorithm called Alternating Support Vector Machine (AltSVM), which is more practical than those existing convex programs in a large-scale setting. Besides, to improve the scalability and practicability of the CR algorithm, Wu et al. [35] try to reduce the time complexity by carefully designing and speeding up the computation of gradient and Hessian vector product, which brings to competitive ranking performance as well as better theoretical time complexity. Different from above CR methods with a fixed score function such as inner product, or only taking into account the relative order between observed interactions and non-observed ones, we aim to learn an arbitrary utility function which digs out the naturally existing pairwise comparison data from the interacted items for each user, so as to achieve better personalized ranking.

III. Methodology

In this section, we first introduce the mathematical notations used throughout this paper. Then we elaborate on the architecture of our model, where an arbitrary continuous real utility function can be learned by optimizing a ranking loss. Finally, we systematically show the learning procedure of our method.

A. Notations

Given a set of users $\mathcal{U}$ and a set of items $\mathcal{V}$, let $|\mathcal{U}|$ and $|\mathcal{V}|$ denote the number of users and items, respectively. Since we focus on the recommendation under collaborative ranking framework in this paper, the initial comparison matrix of user i is denoted as $C^i \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, where C^i_{jk} records user i's preference for item j and item k. For example, $C^i_{jk} = 1$ means that user i prefers item j to item k, and $C^i_{jk} = 0$ for else. Therefore, it is natural to transform users' comparison matrix into a set of pairwise comparison graph. Consequently, all the observed comparisons form the training set $\mathcal{T} = \{(i,j,k)\,|i \in \mathcal{U}, j,k \in \mathcal{V}, j \neq k\}$. Also, the label for each comparison (i,j,k) could be denoted as follows:

$$y^i_{jk} = \begin{cases} 1, & \text{if user } i \text{ prefers item } j \text{ to item } k, \\ -1, & \text{otherwise.} \end{cases}$$

Therefore, $\mathcal{Y} = \left\{y^i_{jk}\right\}$ is the label set of all triplets.

B. Framework

Borrowing the wisdom from existing methods, we propose our NCPL as an improvement. On one hand, we take the naturally existing pairwise comparisons between observed interactions into consideration. On the other hand, we adopt neural networks to obtain a utility function for personalized ranking. As shown in Figure 1, we first construct a pairwise comparison graph based on users' comparison matrix. Then we adopt a neural network to learn an arbitrary continuous real utility function for personalized ranking. After that, a pairwise ranking loss is optimized to preserve the ranking between different items of certain users. Finally, the relative ranking score of each pair is used for top-K recommendation. Specifically, our model mainly consists of the following modules:

- **User/Item embedding**: As our model only takes the index of users and items as the input, we adopt linear transformations to project their ids to dense vectors before further modeling.
- **Ranking utility function modeling**: In general, we aim to obtain a utility function for personalized ranking. However, only using the shallow linear modeling as in traditional MF or metric learning might fail to capture those intricate relations. To this end, we model the utility function for user-item ranking with a shallow and a deep structure implemented, respectively. As a result, we obtain two different vectors to imply the latent embedding for each user-item pair.
- **Pairwise ranking**: Based on the two embedding vectors, we could calculate the final relative ranking scores for

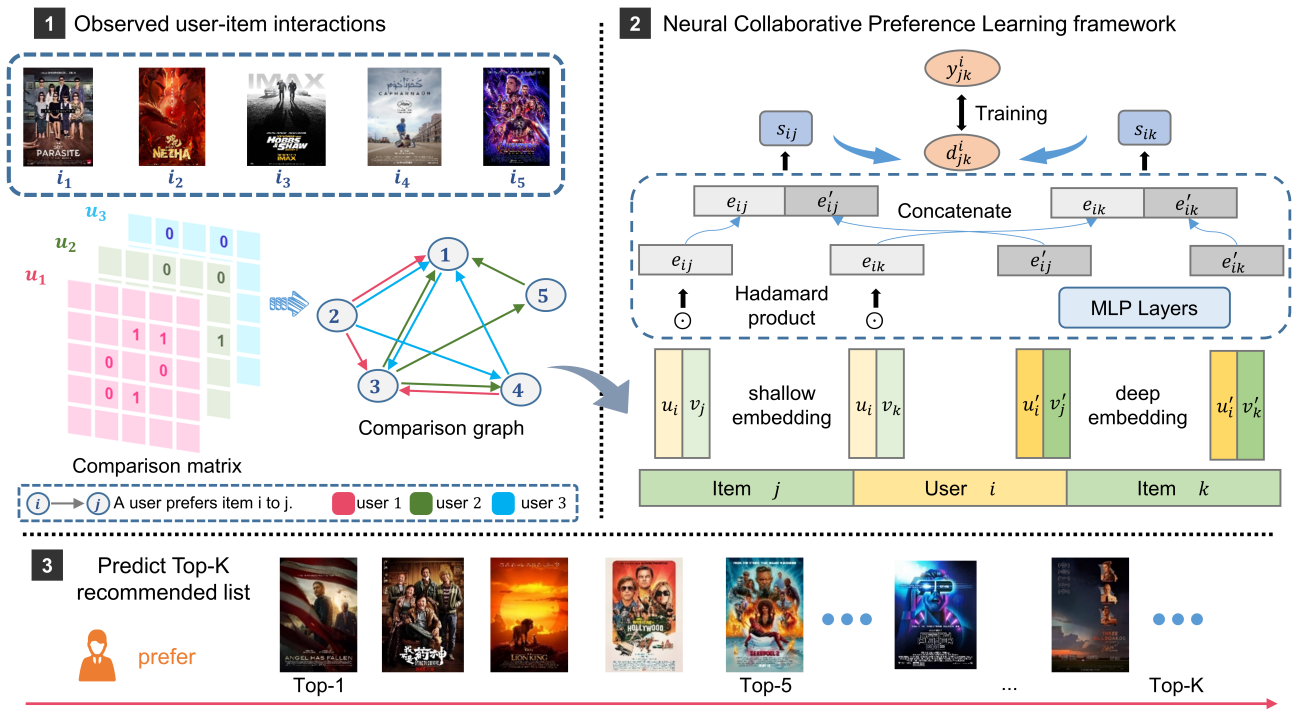

Fig. 1. The framework of our method. (1) Our model employs users' comparison matrix as the initial input, and projected them to a pairwise comparison graph. (2) Then we develop a neural collaborative preference learning framework called NCPL to approximate an arbitrary utility function for personalized ranking. A pairwise ranking loss is adopted here to optimize the margin of score difference between different interacted items of certain users. (3) With the learned score function, NCPL could predict a relative ranking score for each unseen user-item pair and derive the top-K preference list.

user-item pairs. Then a pairwise ranking loss is optimized to preserve the ranking of different items for each user.

Next, we will introduce how our framework models utility function for ranking from the shallow and deep perspective.

1) Utility Function Modeling with Shallow Linear Structure: Given a triplet (i, j, k), we first separately encode the index of users and items into one-hot vectors. To exploit the latent information, we further learn two linear transformations to map the sparse vectors of users and items to dense representations. Hence, for user i and item j, we obtain their latent representations $\boldsymbol{u}_i, \boldsymbol{v}_j \in \mathbb{R}^d$. Borrowing the wisdom of MF and metric learning, we hope the latent representations for users and items lie in a joint latent space after the mapping. Then the preference for a user could be measured with the distance between the embeddings of the user and item. Considering the issues including portability and user privacy, in this paper we focus on the pure collaborative preference learning setting, $i.e.$, we only take the index of users and items as the input. Actually, we could adopt additional information into our framework by concatenating the feature vectors with corresponding user or item latent embeddings.

With these latent embeddings, we then learn from traditional MF methods to define a linear interaction operation for users and items. In MF models, the inner product is adopted as the rating function for latent user and item embeddings. Similarly, we employ the Hadamard product to model the utility function in a linear manner. Different from the inner product, we do not implement the final sum operation in our model since the purpose of our interaction operation is to obtain a shallow embedding vector. For the triplet, the operation could be written as:

$$\boldsymbol{e}_{ij} = \boldsymbol{u}_i \odot \boldsymbol{v}_j, \tag{1}$$

where $\boldsymbol{e}_{ij} \in \mathbb{R}^d$ denotes the shallow embedding vector of the user-item pair (i, j).

2) Utility Function Modeling with Deep Nonlinear Structure: Intuitively, the shallow linear modeling might fail to capture those complex patterns between users and items. To find complement information, we also leverage the strength of neural networks to integrate non-linear user-item embeddings.

Similar to shallow modeling, firstly we transform users and items into dense embeddings with another two dense mappings. Note that the rich side information of users or items, $e.g.$, users' tag information, visual features of items' images, or textual features of items' descriptions, could be code into embeddings here as the complement information. In this structure we implement different mappings to separate the process of latent coding for shallow and deep modeling such that they could focus on their own modeling.

Let $\boldsymbol{u}_i^{'}, \boldsymbol{v}_j^{'} \in \mathbb{R}^{d^{'}}$ denote the obtained new embeddings for user i and item j. In order to utilize neural networks to model and encode those intricate user-item relations, we concatenate the embeddings of user and item, and then feed it to a stack of fully-connected layers to generate the non-linear user-item embedding vector for ranking:

$$\boldsymbol{e}_{ij}^{'} = f_L \circ \cdots f_2 \circ f_1 \left(\left[\boldsymbol{u}_i^{'}, \boldsymbol{v}_j^{'} \right] \right), \tag{2}$$

where $[\cdot, \cdot]$ denotes the concatenate operator, f_l denotes the l-th hidden layer with ReLU [50] as activation function, L

denotes the total number of hidden layers, and $\boldsymbol{e}_{ij}^{'}$ denotes the generated deep embedding vector of the user-item pair (i, j).

3) Embedding Fusion for Ranking: So far we have obtained two kinds of embeddings, $i.e.$, shallow/deep embedding, involving the simple and complex preference information, respectively. Intuitively, combining two kinds of embeddings may lead to a better modeling utility function for ranking. To achieve comprehensive modeling, we concatenate them as the final embedding vector for each user-item pair. Finally, we implement another fully-connected layer to measure user preference from such user-item embedding vector with a score:

$$s_{ij} = f_{out}\left(\left[\boldsymbol{e}_{ij}, \boldsymbol{e}_{ij}^{'}\right]\right) \tag{3}$$

where f_{out} is the output fully-connected layer, and s_{ij} is the preference score.

Equipped with the structure above, our model could eventually learn an arbitrary continuous real utility function for personalized ranking with both linearity and nonlinearity. In what follows, we move one step further to optimize the preference ranking list for each user in our framework.

Unlike point-wise methods distinguish positive items from negative items, we want to use the difference of predicted scores to optimize the preference ranking of interacted items for each user. To this end, we first calculate the score difference d_{jk}^{i} between user-item pair (i, j) and (i, k):

$$d_{jk}^{i} = s_{ij} - s_{ik}. \tag{4}$$

Obviously, $d_{jk}^{i} > 0$ means that the user i prefers item j to k, otherwise item k is preferred by the user.

In particular, the predicted score of a user and his/her preferred item should be higher than that of a less preferred one. Thus, with the users and items in our training set $\mathcal{T}$, we could apply pairwise comparisons to design an effective ranking loss. For example, we could adopt a hinge loss to optimize the score difference:

$$\mathcal{L}_{hinge} = \sum_{(i,j,k)\in\mathcal{T}} \left[y_{jk}^{i}\left(m - d_{jk}^{i}\right)\right]_{+}, \tag{5}$$

where $[z]_{+} = max\left(z, 0\right)$ denotes the hinge loss, and $m > 0$ is a predefined score difference margin.

However, in order to make the score difference more discriminative, and reduce the workload of predefining the score difference margin m, we design and adapt following cross-entropy loss as the final ranking loss:

$$\begin{aligned}\mathcal{L}_{cross} = \sum_{(i,j,k)\in\mathcal{T}} & -\left[y_{jk}^{i}\right]_{+} ln\left(\sigma\left(d_{jk}^{i}\right)\right) \\ & -\left(1 - \left[y_{jk}^{i}\right]_{+}\right) ln\left(1 - \sigma\left(d_{jk}^{i}\right)\right),\end{aligned} \tag{6}$$

where $\sigma\left(\cdot\right)$ denotes the sigmoid function.

Furthermore, the above loss functions enable our model to learn a rank-preserved score, which is beneficial for the top-K recommendation.

C. Model Learning

In this section, we introduce the regularization and training procedure for our method.

TABLE I
STATISTICS OF THE DATASETS.

	ML-100K	ML-1M	Netflix	Last.fm
#Users	943	6,040	10,000	2,100
#Items	1,682	3,952	17,770	18,475
#Interactions	100,000	1,000,209	8,930,336	92,834
%Density	6.30	4.47	5.03	0.24

1) Regularization: To reduce overfitting of the training dataset and improve the generalization of our model, we apply L2 regularization to restrict the scale of embeddings and weights in our neural networks:

$$\begin{aligned}\mathcal{L}_{reg}\left(\theta_*, \boldsymbol{u}_*, \boldsymbol{v}_*, \boldsymbol{u}_*^{'}, \boldsymbol{v}_*^{'}\right) = \theta_{out}^2 + \sum_{l\in[L]} \theta_l^2 \\ + \sum_{i\in[N]} \left(\boldsymbol{u}_i^2 + \boldsymbol{u}_i^{'2}\right) + \sum_{j\in[M]} \left(\boldsymbol{v}_j^2 + \boldsymbol{v}_j^{'2}\right),\end{aligned} \tag{7}$$

where $\theta_{out}, \theta_l, \theta_*$ denote the weights in the final output layer, l-th hidden layer, as well as all the weights in our model, $\boldsymbol{u}_i$, $\boldsymbol{u}_i^{'}$, $\boldsymbol{v}_j$, $\boldsymbol{v}_j^{'}$ denote the latent representations for user i and item j, and $\boldsymbol{u}_*$, $\boldsymbol{u}_*^{'}$, $\boldsymbol{v}_*$, $\boldsymbol{v}_*^{'}$ denote the latent representations for all the users and items, respectively.

2) Training Procedure: The complete objective function of the proposed model is as follows:

$$\min_{\theta_*, \boldsymbol{u}_*, \boldsymbol{v}_*, \boldsymbol{u}_*^{'}, \boldsymbol{v}_*^{'}} \quad \mathcal{L}_{cross} + \lambda\mathcal{L}_{reg}, \tag{8}$$

where λ is a hyperparameter that controls the weight of regularization. Our training procedure is listed as follows:

- Transform users' comparison matrix into a comparison graph, and construct the training set $\mathcal{T}$.
- Compute gradients and update parameters of the complete objective function, and apply *Adam* [51] to control the learning rate.
- Repeat the second procedure until convergence.

IV. EXPERIMENTS

In this section, we conduct extensive experiments to evaluate the performance of NCPL on many public benchmark datasets, including movie rating datasets MovieLens [52] (MovieLens-100K, MovieLens-1M), Netflix [53], and a user and listened music artists dataset Last.fm [54]. The main statistics of all the datasets are summarized in Table I.

A. MovieLens-100K

1) Dataset description: The full MovieLens dataset was collected from the MovieLens website by GroupLens Researchers. MovieLens-100K is a small subset of the full MovieLens dataset, where each user has at least 20 ratings on a 5-star scale (whole-star ratings only). We first transform the user ratings to a pairwise comparison graph, and then exploit it to evaluate the performance of various algorithms.

As we solve the top-K recommendations in a collaborative ranking framework, we following the same setting of [34] to process the data. Specifically, we randomly select N interacted items as the training samples for each user, and compare the ratings for each pair of items to construct the comparison

matrix, which is further transformed to the training set. For each user, we select at most M items from the rest interacted items, and sort them by ratings to construct the test set, where the top-K items with their ranking are the ideal item ranking result.

2) Evaluation metrics: To evaluate the top-K recommendations of each method, we adopt the following standard performance measures [55], [56]: Hit Ratio (HR@K), Precision@K (P@K), Normalized Discounted Cumulative Gain (NDCG@K) and Area Under ROC Curve (AUC@K).

- **Hit Ratio (HR@K)** intuitively measures whether the preferred items are present in top-K lists.
- **Precision@K** is the ratio of the number of preferred items in top-K recommended list.
- **Normalized DCG (NDCG@K)** measures the average performance of a ranking algorithm by assigning higher scores to the hits at top ranks, with the order of each item in top-K recommended list concerned. Specifically, we calculate the NDCG@K as follows:

$$NDCG@K = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{DCG_u@K}{IDCG_u@K}.$$

For user u, $DCG_u@K$ (Discounted Cumulative Gain) and $IDCG_u@K$ (Ideal Discounted Cumulative Gain) are as follows:

$$DCG_u@K = \sum_{i=1}^{I_u}\frac{1\cdot y_{ui}}{log_2\,(i+1)},$$

$$IDCG_u@K = \sum_{i=1}^{min(K,|I_u^*|)}\frac{1}{log_2\,(i+1)},$$

where I_u is the predicted top-K preference list of user u, and I_u^* is the list of user u's preferred items in the test set. Moreover, y_{ui} is a binary indicator that $y_{ui}=1$ if the predicted item lies in I_u^*, and $y_{ui}=0$ for else.

- **Area Under ROC Curve (AUC@K)** is the probability that the algorithm will rank a randomly chosen preferred item higher than another randomly chosen less preferred one.

$$AUC@K = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}AUC_u@K,$$

$$AUC_u@K = \frac{\sum_{i\in I_u^*}(1+N-rank_i)-\frac{N(1+N)}{2}}{N\,(N-K)},$$

3) Competitors: In this paper, we compare our algorithm with 18 competitors. There are nine CF methods from classical MF model to recent CF method with neural networks:

- **RegularizedSVD** [37] is a classic MF model that optimizes mean-square error with regularization to learn the latent representation for both users and items.
- **GMF** [24] uses the sigmoid function to transform the element-wise product of user and item's latent factor to a probability, and optimize a binary cross-entropy loss.
- **MLP** [24] adopts a multilayer perceptron to learn a nonlinear score function to predict the user-item score. Specifically, it adds hidden layers on the concatenated vector of user and item's latent features, and uses ReLU as the activation function.
- **NeuMF** [24] allows GMF and MLP to learn separate embeddings, and combine the two models by concatenating their last hidden layer to predict a user-item score.
- **IRGAN** [57] is the first model that introduces adversarial learning for CF. In detail, it regards item recommendation as a generalized retrieval problem, and constructs the user profile from their past item consumption. After that, the generative retrieval focuses on predicting relevant documents given a query, and the discriminative retrieval focuses on predicting relevancy given a query-document pair.
- **AutoRec** [22] uses the partial observed vectors to represent users or items, followed by a one-layer autoencoder to reconstruct the input and predict missing ratings. Here we record the performance of Item-based AutoRec since it outperforms the User-based AutoRec.
- **NeuRec** [25] adopts a multilayer perceptron and a latent factor to learn dense vector representations of users and items. Here we record the performance of Item-based NeuRec due to the same reason as for AutoRec.
- **ONCF** [27] uses an outer product to obtain a two-dimensional interaction map for a user-item pair, and employs a convolutional neural network to capture high-order information.
- **NGCF** [28] transforms the user-item interactions into a user-item graph, and explicitly propagates embeddings on it to model the high-order connectivity.

In addition, we compare with nine representative CR models from traditional CR methods to recent neural networks based models as well:

- **CofiRank** [45] is an early CR algorithm based on the idea of MMMF, but directly optimizes a ranking metric NDCG instead of rating measures. It combines the application of bundle methods to convex optimization problems, MMMF, as well as a structured estimation to achieve the goals.
- **RankBasedSVD** [47] is a CR method which handles the problem of top-K recommendations by minimizing a smooth surrogate loss function that forms a convex upper bound on the zero-one loss function.
- **LCR** [47] takes the assumption that the rating matrix is globally high-rank but locally low-rank. Based on this, it generalizes LLORMA [58] from least squares to ranked loss minimization, and combines multiple local low-rank matrices each of which is trained to minimize a ranking loss to achieve competitive performance.
- **AltSVM** [34] is a large-scale non-convex implementation of fitting a rank score matrix to the pairwise data. It trains a factored form of the matrix via alternatively optimizing users and items' latent vector by stochastic dual coordinate descent algorithm in a non-convex framework.
- **Global Ranking** [34] solves the CR problem by removing the personalization in AltSVM and only optimizing the items' latent vector. In detail, it fixes users' latent vectors to all ones and solves for items' latent vector.

- **Primal-CR** [35] aims to tackle the poor scalability problem of existing CR algorithms. It applies Newton's method and calculation of the gradient and Hessian matrix to reduce time complexity significantly.
- **Primal-CR++** [35]exploits the fact that most data is in the form of numerical ratings instead of pairwise comparisons, and further accelerates the gradient and Hessian calculation to improve the performance of Primal-CR.
- **NCR** [36] develops a classification strategy on the framework of NeuMF [24], with an assumption that the non-interacted items are less preferred than interacted items. Specifically, NCR takes one user and two items as input, and predicts the possibility that the user prefers the former item to the latter one.
- **LRML** [49] is a metric learning approach for recommendation, which learns latent relations that describe each user-item interaction, and design a score function considering the latent relation vector.

4) Implementation details: To better evaluate the effectiveness of both shallow and deep parts in our algorithm, we implement our proposed method together with two ablated variants:

- **Shallow-CPL** only uses our shallow linear structure to obtain a shallow utility function for personalized ranking, and adopt a fully-connected layer to obtain the ranking score.
- **Deep-CPL** only uses our deep nonlinear structure to learn a nonlinear utility function, followed by the same fully-connected layer to predict the ranking score.
- **NCPL** takes advantage of the fusion of shallow and deep structures to achieve more comprehensive utility function modeling as described in our paper.

We implement our model with TensorFlow [59], and adopt Adaptive Moment Estimation (Adam) [51] as the mini-batch gradient descent optimizer to adapt the learning rate for faster convergence. Specifically, we set the dimension of shallow user and item embeddings $d = 10$, learning rate $\alpha = 0.05$ and the hyperparameter $\lambda = 0.5$ in Shallow-CPL. We adopt the same setting in Deep-CPL and NCPL except that dimension of deep user and item embeddings is $d^{'} = 64$ and the number of neurons in each hidden layer is $128 \rightarrow 64 \rightarrow 32 \rightarrow 8$ in our deep nonlinear structure. We set $N = 50$ and $M = 70$ for this dataset.

As the implementation of competitors, we directly adopt their released source code, except that for RegularizedSVD, RankBasedSVD and LCR, we adopt the implementations in Personalized Recommendation Algorithms Toolkit (PREA) [60]. Besides, for all competitors with neural networks, the embedding size is set to 64, and the shallow embedding size is set to 20 if the model includes a shallow part. Besides, the network structure, weight decay strategy, learning rate, and other hyperparameters are the same as in their papers or released codes. Moreover, we set $K = 10$ to evaluate the ranking performance at the top of the recommended list.

5) Experimental results: As shown in Table II, we could draw the following conclusions:

a) Our proposed methods outperform all competitors on all the metrics, which verifies the superior performance of our proposed algorithm. Specifically, our method improves the best competitors by 4.50% and 3.62% on P@10 and NDCG@10, respectively.

b) Compared to MLP and NeuMF which adopt neural networks to solve a binary classification problem and obtain quite competitive performance, our methods outperform them on all the metrics. The reason lies in that our methods optimize the items' order for each user, rather than simply treat observed interactions as positive, which neglects the naturally existing item ranking for observed items.

c) Our methods obtain a clear margin in performance gain over NCR that also adopts neural networks for item ranking. This is because our methods optimize the order of different items for each user more thoroughly, while NCR only assumes the observed items are more preferred than non-observed ones.

d) Despite the good performance of LCR and Primal-CR++, our NCPL still outperforms them on all the metrics. A key reason lies in that LCR and Primal-CR++ only use a fixed handcraft score function and suffer from insufficient expression ability, while our method benefits from a more powerful utility function for item ranking learned by the deep neural networks.

e) When it comes to the comparison between CR and CF methods, we observe that some CR methods, even without neural networks, could still achieve comparable ranking performance compared with CF methods included MLP, GMF and IRGAN. This is attributed to the difference between CR and CF methods. Besides, it is obvious that LRML obtains relatively poor performance, which is similar on the following datasets. According to the explanation of their authors [1], perhaps it is because their released code is a barebones version.

f) Now we focus on the results of Shallow-CPL, Deep-CPL and NCPL. By taking advantage of collaborative ranking,

TABLE II
EXPERIMENTAL RESULTS ON MOVIELENS-100K. THE BEST PERFORMANCE IS IN BOLDFACE AND THE SECOND BEST IS UNDERLINED.

Type	Method	HR@10↑	P@10↑	NDCG@10↑	AUC@10↑
CF	RegularizedSVD	0.9557	0.3493	0.3756	0.6481
	MLP	0.9537	0.3837	0.3944	0.6893
	GMF	0.9537	0.3841	0.3945	0.6894
	NeuMF	0.9738	0.4115	0.4419	0.6953
	IRGAN	0.9557	0.2948	0.2932	0.5706
	AutoRec	0.9738	0.3799	0.3974	0.6832
	NeuRec	0.9437	0.3041	0.3145	0.5998
	ONCF	0.9759	0.3918	0.4049	0.7021
	NGCF	0.9716	0.3931	0.3979	0.6797
CR	CoFiRank	0.9659	0.3620	0.3839	0.6640
	RankBasedSVD	0.9759	0.3732	0.3904	0.6726
	LCR	0.9758	0.3755	0.3945	0.6808
	AltSVM	0.9456	0.3614	0.3663	0.6611
	Global Ranking	0.9515	0.3591	0.3666	0.6645
	Primal-CR	0.9738	0.3730	0.3964	0.6763
	Primal-CR++	0.9738	0.3743	0.3969	0.6775
	NCR	0.9698	0.3641	0.3778	0.6314
	LRML	0.8893	0.2455	0.2547	0.5161
Ours	Shallow-CPL	0.9698	0.4010	0.4278	<u>0.6894</u>
	Deep-CPL	<u>0.9799</u>	<u>0.4185</u>	<u>0.4506</u>	**0.7050**
	NCPL	**0.9839**	**0.4300**	**0.4579**	0.6982

[1] https://github.com/vanzytay/WWW2018_LRML

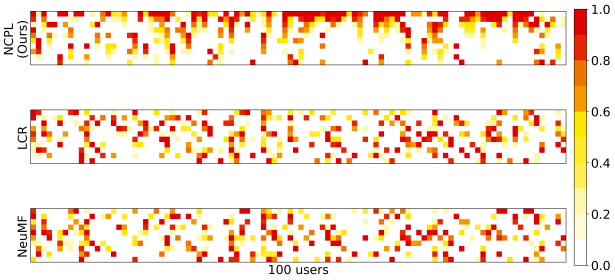

Fig. 2. Visualization of the top-10 recommended list for NeuMF, LCR and our NCPL on MovieLens-100K. The grid in a deeper color represents that the item gets a higher predicted score. Our method possesses more grids in deep colors on the top half of the figure, which proves that we obtain better ranking results compared to competitors.

Shallow-CPL could obtain a competitive result with its simple shallow linear structure. Deep-CPL could make better recommendations with the help of neural networks. With these two parts together, NCPL further improves the accuracy and ranking performance of top-K recommendations.

Except for quantitative results, we also visualize the predictions in Figure 2 to intuitively investigate the ranking performance. First, we visualize the top-10 ranking results of our NCPL with LCR and NeuMF, which are the methods with the best performance in CR and CF competitors on Movielens-100K. For each heatmap, the horizontal axis denotes 100 randomly selected users, and the vertical axis represents the top-10 items for each user from top to bottom. The grid in a deeper color means that the item gets a higher predicted score given by the algorithm, while a lighter color indicates a lower predicted score, and the lightest color (white) represents that a positive item is not included in top-10 predicted list. Consequently, the heatmap of the ideal situation should be filled with grids, and the color of the grid gradually changes from dark to light from the top to the end. As shown in these heatmaps, it is obvious that our NCPL possesses more grids in deeper color on the top half of the figure, which demonstrates we achieve better ranking performance compared to LCR and NeuMF.

Moreover, we calculate the normalized rating of all items for the chosen users, and plot the result of NCPL and LCR in Figure 3. It can be simply seen as averaging the heatmap in Figure 2 along the horizontal axis. Obviously, the ideal curve is monotonically decreasing, as an item of a higher ranking in real should obtain a higher predicted score. In addition, we record the id of items that is not in real top-10 list but obtains a higher predicted score than top-10 items, and mark them as abnormal points. As shown in Figure 3, the curve of NCPL decreases more consistently than LCR in the positive sample area, and there are no abnormal points for NCPL in the negative sample area. Consequently, NCPL better predicts the preference ranking than LCR.

B. MovieLens-1M

1) Dataset description: MovieLens-1M is a larger subset of the full MovieLens dataset, which contains 1,000,209 ratings of 3,900 movies rated by 6,040 users. It is more sparse than MovieLens-100K. We preprocess this subset in the same way as MovieLens-100K.

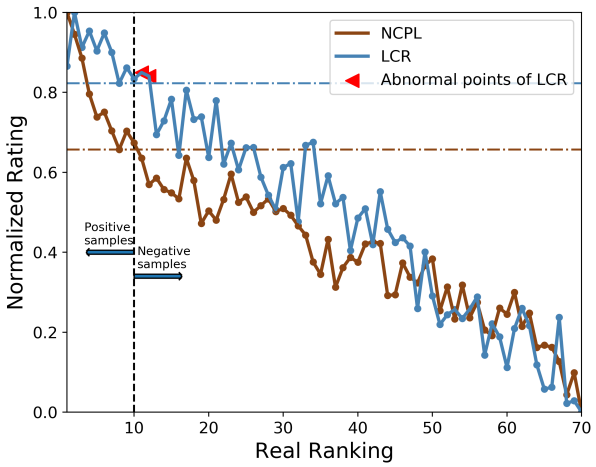

Fig. 3. Normalized rating for test items on MovieLens-100K. The ideal normalized rating should be monotonically decreasing w.r.t. the real ranking. The curve of our NCPL is much more smooth than LCR, especially in the positive sample area. Besides, NCPL has no abnormal point while LCR gets two.

TABLE III
EXPERIMENTAL RESULTS ON MOVIELENS-1M. THE BEST PERFORMANCE IS IN BOLDFACE AND THE SECOND BEST IS UNDERLINED.

Type	Method	HR@10↑	P@10↑	NDCG@10↑	AUC@10↑
CF	RegularizedSVD	0.9822	0.3749	0.3947	0.6844
	MLP	0.9672	0.3733	0.3804	0.7061
	GMF	0.9570	0.3733	0.3806	0.7059
	NeuMF	0.9673	0.3733	0.3805	0.7061
	IRGAN	0.9309	0.3425	0.3632	0.6690
	AutoRec	0.9820	0.3779	0.4001	0.6956
	NeuRec	0.9413	0.3064	0.3137	0.6063
	ONCF	0.9723	0.3867	0.3932	0.7161
	NGCF	0.9457	0.3056	0.3205	0.6026
CR	CoFiRank	0.9744	0.3477	0.3628	0.6655
	RankBasedSVD	0.9802	0.3680	0.3932	0.6642
	LCR	0.9814	0.3745	0.3979	0.6810
	AltSVM	0.9532	0.3441	0.3663	0.6452
	Global Ranking	0.9525	0.3419	0.3627	0.6446
	Primal-CR	0.9673	0.3513	0.3714	0.6596
	Primal-CR++	0.9655	0.3504	0.3706	0.6599
	NCR	0.9535	0.3449	0.3618	0.6419
	LRML	0.8702	0.2285	0.2290	0.5044
Ours	Shallow-CPL	0.9771	0.3652	0.4024	0.6680
	Deep-CPL	<u>0.9904</u>	<u>0.4261</u>	<u>0.4541</u>	<u>0.7167</u>
	NCPL	**0.9906**	**0.4390**	**0.4618**	**0.7251**

2) Implementation details: The training strategy and hyper-parameters are the same as on MovieLens-100K, except that the dimension of shallow users and items embedding d is set to 20.

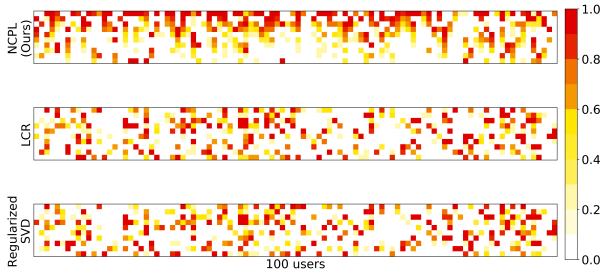

Fig. 4. Visualization of the top-10 recommended list for RegularizedSVD, LCR and our NCPL on MovieLens-1M. Our method still obtains more grids in deep colors on the top half of the figure, which again proves that we achieve better ranking results compared to competitors.

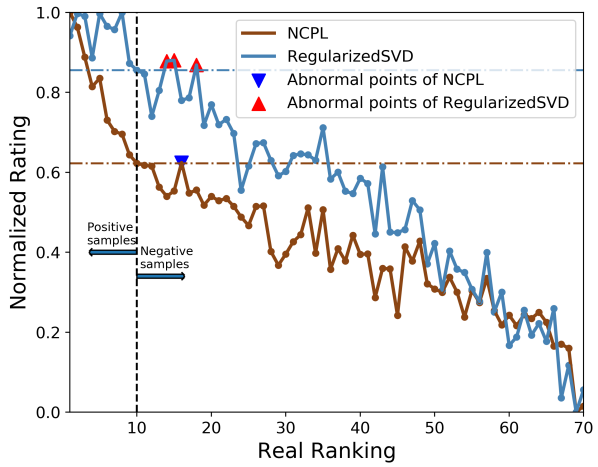

Fig. 5. Normalized rating for test items on MovieLens-1M. The ideal rating is monotonically decreasing w.r.t. the real ranking. The curve of our NCPL is much more smooth than RegularizedSVD, especially in the positive sample area. Moreover, NCPL has fewer abnormal points than RegularizedSVD.

3) Experimental results: As shown in Table III, our proposed method outperforms all the competitors on all the metrics, which is the same on MovieLens-100K. In detail, NCPL improves the best competitors by 13.50% and 15.42% on P@10 and NDCG@10, respectively. Compared to CF methods, our NCPL achieves better performance as we directly optimize the ranking order of different items for each user, which is more effective when facing insufficient user-item interactions. Also, Shallow-CPL itself could achieve comparable or even better performance compared with competitors, while Deep-CPL and NCPL further improve the recommendation quality, leading to the best experimental result among all the competitors. Besides, ONCF obtains competitive performance on this dataset, showing the effectiveness of convolutional neural networks in preference modeling.

Similar to MovieLens-100K, here we draw the heatmaps to compare our ranking performance with LCR and RegularizedSVD on Movielens-1M. As shown in Figure 4, it is obvious that NCPL obtain more grids in deeper color on the top half and in lighter color on the lower half, which again proves that the recommended list predicted by NCPL could better preserve the order of users preference.

Besides, we plot the normalized rating of NCPL and RegularizedSVD on this dataset as well. As shown in Figure 5, the curve of NCPL is more like a monotonically decreasing curve than RegularizedSVD, especially in the positive sample area. It indicates that the score of test items predicted by NCPL could better preserve the actual top-10 ranking result than that of RegularizedSVD. At the same time, there are fewer abnormal points for NCPL compared to RegularizedSVD, which again demonstrates that our model could better predict the item ranking for users.

C. Netflix

1) Dataset description: Netflix dataset is constructed to support participants in the famous open competition Netflix Prize. It includes more than 100,000,000 ratings from 480,000 anonymous customers over 17,000 movie titles. The ratings are also on a scale from 1 to 5 (integral) stars. As the dataset is much larger than MovieLens datasets, we randomly sample 10,000 users with all their ratings to construct a subset, and use this subset for the evaluation.

TABLE IV
EXPERIMENTAL RESULTS ON NETFLIX. THE BEST PERFORMANCE IS IN BOLDFACE AND THE SECOND BEST IS UNDERLINED.

Type	Method	HR@10↑	P@10↑	NDCG@10↑	AUC@10↑
CF	RegularizedSVD	0.9754	0.3185	0.3406	0.6937
	MLP	0.9675	0.2985	0.3189	0.6683
	GMF	0.9463	0.2598	0.2762	0.6226
	NeuMF	0.9646	0.2891	0.3073	0.6612
	IRGAN	0.9584	0.2863	0.3147	0.6382
	AutoRec	0.9736	0.3108	0.3361	0.6809
	NeuRec	0.9501	0.3056	0.3116	0.6046
	ONCF	0.9599	0.3363	0.3475	0.6939
	NGCF	0.9425	0.3065	0.3138	0.6118
CR	CoFiRank	0.9510	0.2705	0.2742	0.6323
	RankBasedSVD	0.9523	0.2785	0.3045	0.6463
	LCR	0.9556	0.2787	0.3072	0.6323
	AltSVM	0.9662	0.3053	0.3312	0.6788
	Global Ranking	0.9692	0.2308	0.3252	0.6741
	Primal-CR	0.9481	0.2809	0.3023	0.6484
	Primal-CR++	0.9482	0.2811	0.3024	0.6484
	NCR	-	-	-	-
	LRML	0.8846	0.2148	0.2361	0.5606
Ours	Shallow-CPL	0.9682	0.3388	0.3495	**0.7007**
	Deep-CPL	0.9813	0.3477	0.3744	0.6981
	NCPL	**0.9843**	**0.3708**	**0.3935**	0.6995

2) Implementation details: All the training strategy and hyperparameters are the same as on MovieLens-1M, except that the learning rate α is 0.01 and λ is 0.01.

3) Experimental results: As shown in Table IV, our proposed methods outperform all the competitors on all the metrics except for AUC@10. Specifically, NCPL improves the best competitors by 10.26% and 13.24% on P@10 and NDCG@10, respectively. Considering that ONCF achieves the best performance compared with other baselines on such a large dataset, the wisdom of applying convolutional neural networks for user-item modeling is worth learning. Besides, here we exclude NCR because it is too time-consuming for us to find its best performance on this large dataset based on NCR's recommendation algorithm.

As Netflix each user has more rated items than in MovieLens-100K or MovieLens-1M, we conduct experiments and record the results for several models to compare the performance with different sizes of test data. Here M is increased from 70 to 200 by 10. As shown in Figure 6, our method almost consistently obtains the highest performance with different test data size M. What's more, the performance gap between NCPL and competitors becomes larger as M grows up, which indicates that our method still predicts a much more reasonable top-10 ranking list compared to competitors with a larger test data size.

D. Last.fm

1) Dataset description: The Last.fm dataset contains a set of 2,100 users and 18,745 artists from Last.fm online music system[2]. Different from the above datasets where users would give an explicit score for each interacted item, we only have user and their listening count of interacted music artists here.

[2]http://www.last.fm

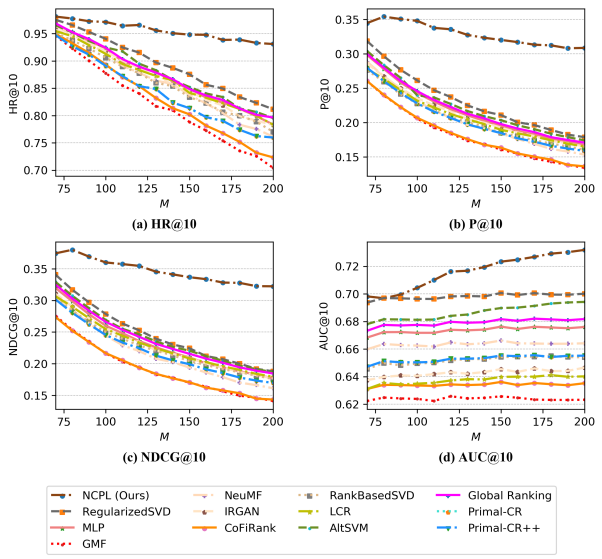

Fig. 6. Comparisons of all the metrics on Netflix w.r.t the different maximum test items M for each user.

However, the listening count could be seen as an indicator of users' preference, so we exploit the user and listened-artist relations to conduct our experiment. Specifically, this dataset is much more sparse (only has a density of 0.0024) than above three datasets, so predicting user preference on this dataset would be a more challenging task.

2) Implementation details: All the training strategy and hyperparameters are the same as on MovieLens-100K, except that λ is set as 0.1. Besides, we set $N = 20$ and sort all the rest interacted artists to construct the test set. Notably, we sort the listened artist for each user by their listening count, *i.e.*, the listened artists with larger listening count will obtain a higher ranking position as ground truth in the test set.

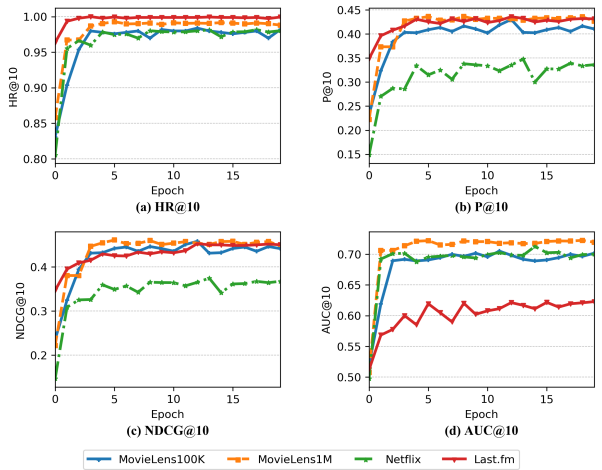

Fig. 7. All the metrics of our NCPL w.r.t. the number of train epochs.

3) Experimental results: As shown in Table V, NCPL outperforms all the competitors on all the metrics. With a view to the performance of other competitors, we find that most CR methods outperform most CF methods especially on NDCG@10 and AUC@10. There are two possible reasons for this phenomenon. On one hand, the absence of explicit rating information limits the performance of CF methods. While most

TABLE V
EXPERIMENTAL RESULTS ON LAST.FM. THE BEST PERFORMANCE IS IN BOLDFACE AND THE SECOND BEST IS UNDERLINED.

Type	Method	HR@10↑	P@10↑	NDCG@10↑	AUC@10↑
CF	RegularizedSVD	0.8946	0.3333	0.3588	0.5071
	MLP	0.9978	0.3847	0.3868	0.5611
	GMF	0.9935	0.3566	0.3645	0.5185
	NeuMF	0.9989	0.3811	0.3876	0.5593
	IRGAN	0.9973	0.3792	0.3825	0.5581
	AutoRec	0.9978	0.4027	0.4121	0.5807
	NeuRec	0.9978	0.4295	0.4335	0.6019
	ONCF	0.9903	0.3605	0.3527	0.5249
	NGCF	0.9984	0.4296	0.4503	0.6037
CR	CoFiRank	0.9953	0.3891	0.3991	0.5586
	RankBasedSVD	0.9965	0.3932	0.4033	0.5616
	LCR	0.9969	0.3941	0.4055	0.5622
	AltSVM	0.9978	0.3947	0.4067	0.5670
	Global Ranking	0.9985	0.4023	0.4252	0.5762
	Primal-CR	0.9989	0.4115	0.4306	0.5897
	Primal-CR++	0.9989	0.4115	0.4306	0.5897
	NCR	0.9957	0.4283	0.4459	0.6047
	LRML	0.9935	0.3368	0.3415	0.5023
Ours	Shallow-CPL	0.9978	0.4294	0.4417	0.6189
	Deep-CPL	**0.9994**	0.4379	0.4529	0.6291
	NCPL	0.9978	**0.4391**	**0.4535**	**0.6298**

CF methods focus on accurately predicting the actual rating of unobserved user-item interaction, in this dataset, there does not exist explicit rating for each user-artist interaction, and almost every user-artist listening count is different. As a result, it is hard to accurately predict the unseen user-artist listening count. On the other hand, the sparseness of this dataset leading to insufficient user-item historical interaction information, which poses a challenge to CF methods. However, these problems could be alleviated by collaborative ranking, which directly optimizes the item ranking for each user. Besides, NGCF achieves better performance on this sparse dataset than on the other three denser datasets, indicating that exploiting high-order connectivity might be promising to tackle the sparsity issue in recommender systems. In addition, our proposed NCPL consistently outperforms all the CR methods, which again validates the effectiveness of such an arbitrary utility function learned by neural networks as well as our shallow and deep fusion strategy.

From the experimental results of our proposed methods on all datasets, we find that the Deep-CPL achieves better performance than shallow-CPL with the representability of the neural networks. Besides, NCPL's performance gain over Deep-CPL on a large dataset (Netflix) is more obvious than on relatively small datasets including MovieLens-100K, MovieLens-1M and Last.fm. The possible reason lies in that Deep-CPL might be overfitting with a large amount of users, while NCPL benefits from the linearity of shallow modeling.

To investigate the learning procedure of NCPL, we plot the change of all metrics on all datasets in Figure 7. It is obvious that most metrics increase rapidly to a high level over only several epochs and then tend to be stable, which benefits from the strong expression ability of neural networks.

E. Study of NCPL

In this section, we evaluate the different aspects of our NCPL, including the efficiency comparisons, the impact of different λ and different ranking loss function, as well as the performance under different K.

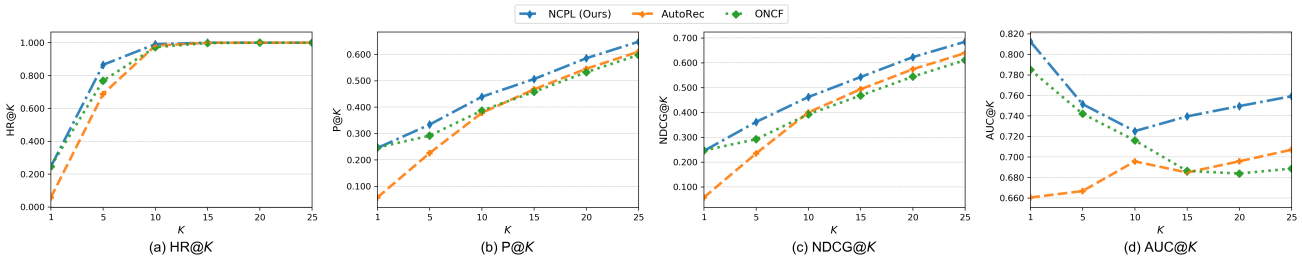

Fig. 8. HR@K, P@K, NDCG@K and AUC@K for top-K recommendations for MovieLens-1M.

TABLE VI
EXPERIMENTAL RESULTS OF NCPL WITH DIFFERENT λ. THE BEST PERFORMANCE IS IN BOLDFACE AND THE SECOND BEST IS UNDERLINED.

Dataset	λ	HR@10↑	P@10↑	NDCG@10↑	AUC@10↑
MovieLens-100K	0.01	0.9496	0.3865	0.3964	0.6837
	0.05	0.9537	0.3837	0.3944	0.6892
	0.1	0.9637	0.3877	0.4036	0.6935
	0.5	**0.9839**	**0.4300**	**0.4579**	**0.6982**
	1.0	0.9839	0.3917	0.4263	0.6730
MovieLens-1M	0.01	0.9672	0.3732	0.3804	0.7060
	0.05	0.9873	0.4101	0.4355	0.6972
	0.1	**0.9916**	0.4183	0.4495	0.7114
	0.5	0.9906	**0.4390**	**0.4618**	**0.7251**
	1.0	0.9867	0.4165	0.4419	0.7040
Netflix	0.01	0.9843	**0.3708**	**0.3935**	0.6995
	0.05	0.9839	0.3634	0.3861	0.6963
	0.1	**0.9846**	0.3676	0.3874	**0.7051**
	0.5	0.9815	0.3598	0.3790	0.6855
	1.0	0.9777	0.3364	0.3625	0.6742
Last.fm	0.01	0.9983	0.4275	0.4299	0.6211
	0.05	0.9983	0.4268	0.4308	0.6211
	0.1	0.9978	**0.4391**	0.4535	**0.6298**
	0.5	0.9984	0.4316	0.4529	0.6237
	1.0	**0.9989**	0.4328	**0.4563**	0.6030

1) The impact of different λ: As mentioned in subsection III-C, we use $\mathcal{L}_{reg}$ to reduce overfitting and improve the generalization ability, with the hyperparameter λ to control its weight. To investigate how λ should be set for different datasets, we apply a grid search and record their performance in Table VI. From the experimental results, on relatively small datasets($e.g.$, MovieLens-100K, MovieLens-1M, and Last.fm), a small λ such as 0.01 fails to provide a satisfactory performance, while a medium or relatively large λ($e.g.$, 0.1, 0.5, 1.0) usually leads to a better performance. However, on a relatively large dataset (Netflix), a small $\lambda = 0.01$ is better for the model's generalization ability.

2) The impact of different ranking loss function: As described in subsection III-B, we design two different ranking loss functions and adopt $\mathcal{L}_{cross}$ to reduce the workload of parameter adjustment. However, here we conduct experiments to better compare its performance with $\mathcal{L}_{hinge}$. As shown in Table VII, with extra effort to find a proper hyperparameter m as the score margin, applying $\mathcal{L}_{hinge}$ could obtain comparable or even better performance, while $\mathcal{L}_{hinge}$ achieves satisfactory performance without hyperparameter m.

3) The performance under different K: To fully study the performance under different K, here we test $K \in \{1, 5, 10, 15, 20, 25\}$ on MovieLens-1M for NCPL, AutoRec, and ONCF, which are the three models with best performance of top-10 on this dataset. As shown in Figure 8, with the increasing of K, HR@K, P@K and NDCG@K increase as well. Our NCPL consistently outperforms AutoRec and ONCF with different K. Besides, the difference of AUC@K even becomes larger for $K > 10$, indicating that our utility function is more likely to rank the real preferred items higher.

TABLE VII
EXPERIMENTAL RESULTS OF NCPL WITH DIFFERENT WITH DIFFERENT RANKING LOSS FUNCTIONS. THE BEST PERFORMANCE IS IN BOLDFACE AND THE SECOND BEST IS UNDERLINED.

Dataset	Ranking loss		HR@10↑	P@10↑	NDCG@10↑	AUC@10↑
MovieLens-100K	$\mathcal{L}_{hinge}$	$(m=1)$	0.9859	0.4303	0.4546	0.7040
		$(m=2)$	**0.9959**	**0.4329**	0.4550	**0.7115**
		$(m=3)$	0.9597	0.3903	0.4031	0.6810
	$\mathcal{L}_{cross}$		0.9839	0.4300	**0.4579**	0.6982
MovieLens-1M	$\mathcal{L}_{hinge}$	$(m=1)$	**0.9916**	0.4220	0.4456	0.7073
		$(m=2)$	0.9870	0.4071	0.4327	0.7002
		$(m=3)$	0.9888	0.4021	0.4335	0.7029
	$\mathcal{L}_{cross}$		0.9906	**0.4390**	**0.4618**	**0.7251**
Netflix	$\mathcal{L}_{hinge}$	$(m=1)$	0.9654	0.3267	0.3399	0.6601
		$(m=2)$	0.9651	0.3140	0.3314	0.6437
		$(m=3)$	0.9613	0.3317	0.3400	**0.7166**
	$\mathcal{L}_{cross}$		**0.9843**	**0.3708**	**0.3935**	0.6995
Last.fm	$\mathcal{L}_{hinge}$	$(m=1)$	**0.9983**	0.4275	0.4299	0.6211
		$(m=2)$	0.9973	**0.4402**	**0.4536**	**0.6301**
		$(m=3)$	0.9978	0.4389	0.4532	0.6093
	$\mathcal{L}_{cross}$		0.9978	0.4391	0.4535	0.6298

TABLE VIII
RUNNING TIME ON MOVIELENS-1M AND LAST.FM DATASET.

Method	MovieLens-1M		Last.fm	
	train	test	train	test
AutoRec	$1.2s$	$1.1s$	$1.1s$	$2.1s$
NeuMF	$5.2s$	$9.4s$	$4.3s$	$0.9s$
ONCF	$10.4s$	$1.7s$	$4.2s$	$0.4s$
NGCF	$12.3s$	$52.2s$	$4.1s$	$15.0s$
IRGAN	$23.9s$	$10.6s$	$13.9s$	$23.6s$
NCR	$32.3s$	$85769.6s$	$8.0s$	$9826.1s$
NeuRec	$100.6s$	$12.2s$	$255.4s$	$8.7s$
LRML	$463.9s$	$15.4ss$	$1260.1s$	$9.4s$
NCPL (Ours)	$29.2s$	$19.2s$	$4.2s$	$10.7s$

4) Efficiency comparisons: To compare the efficiency of our proposed NCPL with baseline approaches, we test several neural network based models on the same Linux machine with a TITAN RTX GPU, 2 Intel Xeon CPUs (E5-2620 v4 @2.10GHz), and 128GB of RAM. Here we exclude the classical methods as they do not use GPU for speeding up. We record the running time of training and testing for one epoch in Table VIII, where we found that our NCPL needs a medium time compared with other models. Besides, NCR is time-consuming to predict the top-K item ranking list as it needs to scan the candidate item set K times.

V. CONCLUSION

In this paper, we develop a novel collaborative ranking model with deep neural networks called NCPL to effectively make the top-K recommendations. On the one hand, we

adopt deep neural networks that combine shallow embeddings with deep embeddings to learn an arbitrary continuous real utility function for personalized ranking. On the other hand, a pairwise ranking loss is optimized to deal with the naturally existing pairwise comparisons of certain users for different interacted items properly. Finally, our model could predict a rank-preserved score for each user-item pair, which is beneficial to top-K recommendations. Extensive experiments on four real-world datasets demonstrate the effectiveness of our proposed method.

REFERENCES

[1] S. Huang, J. Zhang, D. Schonfeld, L. Wang, and X. Hua, "Two-stage friend recommendation based on network alignment and series expansion of probabilistic topic model," *IEEE Transactions on Multimedia*, vol. 19, no. 6, pp. 1314–1326, 2017.

[2] C.-Y. Liu, C. Zhou, J. Wu, Y. Hu, and L. Guo, "Social recommendation with an essential preference space," in *AAAI*, 2018, pp. 346–353.

[3] G. Zhao, X. Lei, X. Qian, and T. Mei, "Exploring users' internal influence from reviews for social recommendation," *IEEE Transactions on Multimedia*, vol. 21, no. 3, pp. 771–781, 2019.

[4] Q. Zhu, X. Zhou, Z. Song, J. Tan, and L. Guo, "Dan: Deep attention neural network for news recommendation," in *AAAI*, 2019, pp. 5973–5980.

[5] Z. Xu, L. Chen, Y. Dai, and G. Chen, "A dynamic topic model and matrix factorization-based travel recommendation method exploiting ubiquitous data," *IEEE Transactions on Multimedia*, vol. 19, no. 8, pp. 1933–1945, 2017.

[6] L. Sun, X. Wang, Z. Wang, H. Zhao, and W. Zhu, "Social-aware video recommendation for online social groups," *IEEE Transactions on Multimedia*, vol. 19, no. 3, pp. 609–618, 2017.

[7] Z. Zhao, Q. Yang, H. Lu, T. Weninger, D. Cai, X. He, and Y. Zhuang, "Social-aware movie recommendation via multimodal network learning," *IEEE Transactions on Multimedia*, vol. 20, no. 2, pp. 430–440, 2018.

[8] Y. Zhou, J. Wu, T. H. Chan, S. Ho, D. Chiu, and D. Wu, "Interpreting video recommendation mechanisms by mining view count traces," *IEEE Transactions on Multimedia*, vol. 20, no. 8, pp. 2153–2165, 2018.

[9] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *WWW*, 2001, pp. 285–295.

[10] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *AAI*, vol. 2009, pp. 421 425:1–421 425:19, 2009.

[11] T. Hofmann, "Latent semantic models for collaborative filtering," *TOIS*, vol. 22, no. 1, pp. 89–115, 2004.

[12] J. Bobadilla, F. Serradilla, A. Hernando *et al.*, "Collaborative filtering adapted to recommender systems of e-learning," *Knowledge-Based Systems*, vol. 22, no. 4, pp. 261–265, 2009.

[13] J.-M. Yang and K. F. Li, "Recommendation based on rational inferences in collaborative filtering," *Knowledge-Based Systems*, vol. 22, no. 1, pp. 105–114, 2009.

[14] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender Systems Handbook*. Springer, 2011, pp. 1–35.

[15] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *UAI*, 2009, pp. 452–461.

[16] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *ICDM*, 2008, pp. 263–272.

[17] K. Wang, X. Duan, J. Ma, C. Sha, X. Wang, and A. Zhou, "Local weighted matrix factorization for implicit feedback datasets," in *DASFAA*, 2016, pp. 381–395.

[18] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *KDD*, 2008, pp. 426–434.

[19] B. H. Le, K. Mori, and R. Thawonmas, "An extension for bounded-svd—a matrix factorization method with bound constraints for recommender systems," *JIP*, vol. 24, no. 2, pp. 314–319, 2016.

[20] S. Rendle, "Factorization machines," in *ICDM*, 2010, pp. 995–1000.

[21] T. V. Nguyen, A. Karatzoglou, and L. Baltrunas, "Gaussian process factorization machines for context-aware recommendations," in *SIGIR*, 2014, pp. 63–72.

[22] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *WWW*, 2015, pp. 111–112.

[23] D. Liang, R. G. Krishnan, M. D.Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *WWW*, 2018, pp. 689–698.

[24] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *WWW*, 2017, pp. 173–182.

[25] S. Zhang, L. Yao, A. Sun, S. Wang, G. Long, and M. Dong, "Neurec: On nonlinear transformation for personalized ranking," in *IJCAI*, 2018, pp. 3669–3675.

[26] W. Niu, J. Caverlee, and H. Lu, "Neural personalized ranking for image recommendation," in *WSDM*, 2018, pp. 423–431.

[27] X. He, X. Du, X. Wang, F. Tian, J. Tang, and T. Chua, "Outer product-based neural collaborative filtering," in *IJCAI*, 2018, pp. 2227–2233.

[28] X. Wang, X. He, M. Wang, F. Feng, and T. Chua, "Neural graph collaborative filtering," in *SIGIR*, B. Piwowarski, M. Chevalier, É. Gaussier, Y. Maarek, J. Nie, and F. Scholer, Eds., 2019, pp. 165–174.

[29] R. He, W. Kang, and J. J. McAuley, "Translation-based recommendation," in *RecSys*, 2017, pp. 161–169.

[30] S. Hacker and L. Von Ahn, "Matchin: eliciting user preferences with an online game," in *ACM CHI*, 2009, pp. 1207–1216.

[31] S. Balakrishnan and S. Chopra, "Collaborative ranking," in *WSDM*, 2012, pp. 143–152.

[32] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic, "Climf: collaborative less-is-more filtering," in *IJCAI*, 2013, pp. 3077–3081.

[33] J. Yi, R. Jin, S. Jain, and A. Jain, "Inferring users' preferences from crowdsourced pairwise comparisons: A matrix completion approach," in *HCOMP*, 2013.

[34] D. Park, J. Neeman, J. Zhang, S. Sanghavi, and I. Dhillon, "Preference completion: Large-scale collaborative ranking from pairwise comparisons," in *ICML*, 2015, pp. 1907–1916.

[35] L. Wu, C.-J. Hsieh, and J. Sharpnack, "Large-scale collaborative ranking in near-linear time," in *KDD*, 2017, pp. 515–524.

[36] B. Song, X. Yang, Y. Cao, and C. Xu, "Neural collaborative ranking," in *CIKM*, A. Cuzzocrea, J. Allan, N. W. Paton, D. Srivastava, R. Agrawal, A. Z. Broder, M. J. Zaki, K. S. Candan, A. Labrinidis, A. Schuster, and H. Wang, Eds., 2018, pp. 1353–1362.

[37] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *KDD*, 2007, pp. 5–8.

[38] Z. Qiao, P. Zhang, W. Niu, C. Zhou, P. Wang, and L. Guo, "Online non-parametric max-margin matrix factorization for collaborative prediction," in *ICDM*, 2014, pp. 520–529.

[39] J. Wang, F. Tian, W. Liu, and X. Wang, "Ranking preserving nonnegative matrix factorization," in *IJCAI*, 2018, pp. 2776–2782.

[40] Y. Liu, L. Zhao, G. Liu, X. Lu, P. Gao, X.-L. Li, and Z. Jin, "Dynamic bayesian logistic matrix factorization for recommendation with implicit feedback." in *IJCAI*, 2018, pp. 3463–3469.

[41] T. Xiao, S. Liang, W. Shen, and Z. Meng, "Bayesian deep collaborative matrix factorization," in *AAAI*, 2019, pp. 5474–5481.

[42] Z. Sun, Q. Guo, J. Yang, H. Fang, G. Guo, J. Zhang, and R. Burke, "Research commentary on recommendations with side information: A survey and research directions," *Electronic Commerce Research and Applications*, vol. 37, 2019.

[43] Y. Yang, Y. Xu, E. Wang, J. Han, and Z. Yu, "Improving existing collaborative filtering recommendations via serendipity-based algorithm," *IEEE Transactions on Multimedia*, vol. 20, no. 7, pp. 1888–1900, 2018.

[44] C. Hsieh, L. Yang, Y. Cui, T. Lin, S. J. Belongie, and D. Estrin, "Collaborative metric learning," in *WWW*, 2017, pp. 193–201.

[45] M. Weimer, A. Karatzoglou, Q. V. Le, and A. J. Smola, "Cofi rank-maximum margin matrix factorization for collaborative ranking," in *NIPS*, 2008, pp. 1593–1600.

[46] M. Volkovs and R. S. Zemel, "Collaborative ranking with 17 parameters," in *NIPS*, 2012, pp. 2294–2302.

[47] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer, "Local collaborative ranking," in *WWW*, 2014, pp. 85–96.

[48] J. Hu and P. Li, "Decoupled collaborative ranking," in *WWW*, 2017, pp. 1321–1329.

[49] Y. Tay, L. A. Tuan, and S. C. Hui, "Latent relational metric learning via memory-based attention for collaborative ranking," in *WWW*, P. Champin, F. L. Gandon, M. Lalmas, and P. G. Ipeirotis, Eds., 2018, pp. 729–739.

[50] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *AISTATS*, 2011, pp. 315–323.

[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015, pp. 1,15.

[52] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *TiiS*, vol. 5, no. 4, p. 19, 2016.

[53] Netflix, "Netflix prize data set," 2009.

[54] I. Cantador, P. Brusilovsky, and T. Kuflik, "2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011)," in *RecSys*, 2011.
[55] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *TOIS*, vol. 20, no. 4, pp. 422–446, 2002.
[56] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve." *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.
[57] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang, "Irgan: A minimax game for unifying generative and discriminative information retrieval models," in *SIGIR*, 2017, pp. 515–524.
[58] J. Lee, S. Kim, G. Lebanon, and Y. Singer, "Local low-rank matrix approximation," in *ICML*, 2013, pp. 82–90.
[59] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *Symposium on Operating Systems Design and Implementation*, 2016, pp. 265–283.
[60] J. Lee, M. Sun, and G. Lebanon, "PREA: personalized recommendation algorithms toolkit," *J. Mach. Learn. Res.*, vol. 13, pp. 2699–2703, 2012.

Zhaopeng Li received the bachelor's degree in computer science from University of Electronic Science and Technology of China (UESTC) in 2018. He is currently pursuing the M.S. degree with University of Chinese Academy of Sciences. His research interests include recommender system and computer vision.

Qianqian Xu received the B.S. degree in computer science from China University of Mining and Technology in 2007 and the Ph.D. degree in computer science from University of Chinese Academy of Sciences in 2013. She is currently an Associate Professor with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. Her research interests include statistical machine learning, with applications in multimedia and computer vision. She has authored or coauthored 30+ academic papers in prestigious international journals and conferences (including T-PAMI, T-IP, NeurIPS, ICML, CVPR, AAAI, etc), among which she has published 6 full papers with the first author's identity in ACM Multimedia. She served as member of professional committee of CAAI, and member of online program committee of VALSE, etc.

Yangbangyan Jiang received the bachelor's degree in instrumentation and control from Beihang University (BUAA) in 2017. She is currently pursuing the Ph.D. degree with University of Chinese Academy of Sciences. Her research interests include machine learning and computer vision.

Ke Ma is a postdoctoral research fellow with the School of Computer Science and Technology, University of Chinese Academy of Sciences (UCAS), Beijing, China. He received the B.S. degree in mathematics from Tianjin University in 2009, M.E. degree in software engineering from Beihang University (BUAA) in 2013, and the Ph.D. degree in computer science form the Key Laboratory of Information Security (SKLOIS), Institute of Information Engineering (IIE), Chinese Academy of Sciences (CAS), in 2019. His research interests include statistical learning, algorithmic game theory and stochastic optimization for large-scale data analysis.

Xiaochun Cao , Professor of the Institute of Information Engineering, Chinese Academy of Sciences. He received the B.E. and M.E. degrees both in computer science from Beihang University (BUAA), China, and the Ph.D. degree in computer science from the University of Central Florida, USA, with his dissertation nominated for the university level Outstanding Dissertation Award. After graduation, he spent about three years at ObjectVideo Inc. as a Research Scientist. From 2008 to 2012, he was a professor at Tianjin University. He has authored and coauthored over 100 journal and conference papers. In 2004 and 2010, he was the recipients of the Piero Zamperoni best student paper award at the International Conference on Pattern Recognition. He is a fellow of IET and a Senior Member of IEEE. He is an associate editor of IEEE Transactions on Image Processing, IEEE Transactions on Circuits and Systems for Video Technology and IEEE Transactions on Multimedia.

Qingming Huang is a professor in the University of Chinese Academy of Sciences and an adjunct research professor in the Institute of Computing Technology, Chinese Academy of Sciences. He graduated with a Bachelor degree in Computer Science in 1988 and Ph.D. degree in Computer Engineering in 1994, both from Harbin Institute of Technology, China. His research areas include multimedia computing, image processing, computer vision and pattern recognition. He has authored or coauthored more than 400 academic papers in prestigious international journals and top-level international conferences. He is the associate editor of IEEE Trans. on CSVT and Acta Automatica Sinica, and the reviewer of various international journals including IEEE Trans. on PAMI, IEEE Trans. on Image Processing, IEEE Trans. on Multimedia, etc. He is a Fellow of IEEE and has served as general chair, program chair, track chair and TPC member for various conferences, including ACM Multimedia, CVPR, ICCV, ICME, ICMR, PCM, BigMM, PSIVT, etc.