Decoupled Collaborative Ranking

Jun Hu
Rutgers University
jh900@cs.rutgers.edu

Ping Li
Rutgers University and Baidu Research
pingli@stat.rutgers.edu

ABSTRACT

We propose a new pointwise collaborative ranking approach for recommender systems, which focuses on improving ranking performance at the top of recommended list. Our approach is different from common pointwise methods in that we consider user ratings as ordinal rather than viewing them as real values or categorical labels. In addition, positively rated items (higher rating scores) are emphasized more in our method in order to improve the performance at the top of recommended list.

In our method, user ratings are modeled based on an ordinal classification framework, which is made up of a sequence of binary classification problems in which one discriminates between ratings no less than a specific ordinal category c and ratings below that category ($\{\geq c\}$ vs. $\{< c\}$). The results are used subsequently to generate a ranking score that puts higher weights on the output of those binary classification problems concerning high values of c so as to improve the ranking performance at the top of list. As our method crucially builds on a decomposition into binary classification problems, we call our proposed method as Decoupled Collaborative Ranking (DCR). As an extension, we impose pairwise learning on DCR, which yields further improvement with regard to the ranking performance of the proposed method. We demonstrate through extensive experiments on benchmark datasets that our method outperforms many considered state-of-the-art collaborative ranking algorithms in terms of the NDCG metric.

Keywords

recommender systems, collaborative ranking, matrix factorization

1. INTRODUCTION

The main purpose of recommender systems is to make suitable recommendations of items that are potentially interesting to users. We consider a general recommendation setting: a set of ratings are recorded by m users over n items, which are represented by an $m \times n$ user-item (U-I) sparse rating matrix R, where the observed ratings take discrete scores and most of them are absent due to incomplete observations. The task of item recommendation in this scenario boils down to selecting the items which are potentially interesting to users based on predictions of the unobserved entries in R.

Methods designed for the purpose of item recommendation can be classified into three categories: pointwise, pairwise, and listwise. The pointwise approaches are quite similar to regression or classification methods, which access user ratings in the form of single points and optimize squared errors or classification errors. Most of the collaborative filtering (CF) approaches are pointwise approaches. Pairwise or listwise approaches usually optimize ranking-based evaluations and access observed ratings in the form of ordered pairs or lists. Considering the computational efficiency of algorithms, pointwise approaches are always preferred since the computational complexity of pointwise methods scales linearly with data size, while it scales quadratically in pairwise approaches. Listwise approaches are even more computationally expensive since they optimize an objective function with respect to a large set of permutations where one permutation defines one potential order of all the rated items. Adding one rating value may generate many permutations as input for optimizing the listwise objective function.

On the other hand, pairwise and listwise collaborative ranking methods generally outperform pointwise methods in terms of ranking performance [13, 24, 3]. In pointwise approaches, user ratings are usually defined as numerical value or categorical labels. In many studies concerning item recommendation [13, 11, 16, 7], it is argued that viewing ratings as numerical values or class labels may not accurately reflect user feedback as provided by qualitative ratings. For example, the difference of user preference between a 5-star rating score and a 4-star rating score could be different from that between 4-star and 3-star. Nevertheless, if we view ratings as numerical, the distance between 5-star and 4-star is considered identical to that between 4-star and 3-star. If ratings are considered as categorical labels, then all the rating scores are treated equally as nominal class labels. In view of this factor, pairwise and listwise methods were proposed to model the ordinality of user ratings, which demonstrated improvement of ranking performance compared with common pointwise approaches in the task of item recommendation. Practically, modeling user ratings as ordinal in the style of pointwise is not straightforward since pointwise methods ac-

WWW 2017, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4913-0/17/04.
http://dx.doi.org/10.1145/3038912.3052685

cess data in the form of single points, while the ordinality of user ratings is usually revealed as pairs or ordered lists.

In this paper, we propose a pointwise collaborative ranking method for item recommendation. Different from common pointwise approaches, we view user ratings as ordinal categorical labels. In addition, we care more about the performance at the top of recommended items, i.e., the items users are more likely to have a closer look at. Through empirical study, we observe that higher rating scores have a greater impact on the top-N ranking performance, compared with lower rating scores. Hence, in our method, we emphasize more on the higher rating values.

Our approach is based on an idea of ordinal classification [14]. More specifically, given a user-item preference matrix R where the ratings range from 1-star to S-star, we firstly decompose R into S binary matrices. An example of matrix decomposition is shown in Figure 1 when $S = 5$. After the matrix decomposition, we then collaboratively learn user and item latent factors through binary classification on each of the decoupled matrices. The final ranking score of an item is generated as a weighted sum of the predictions from each of the decoupled binary matrices. The purpose of this approach is to place an emphasis on higher rating scores. As our method crucially builds on a decomposition into binary classification problems, we call our proposed method as Decoupled Collaborative Ranking (DCR). As an extension, we combine pairwise strategies with DCR, which further improves the ranking performance of the proposed method. Through extensive experiments on benchmark datasets, we demonstrate that our method is competitive, usually markedly better than existing collaborative ranking approaches in terms of the NDCG metric [8]. Besides, DCR, as a pointwise approach, shows better performance in terms of time efficiency compared to pairwise or listwise approaches.

2. RELATED WORK

Item recommendation algorithms are usually inspired by learning to rank (LTR), where methods can be classified into pointwise, pairwise and listwise approaches.

Most of the collaborative filtering (CF) methods lie in the set of pointwise methods. CF algorithms can be categorized into two classes: neighborhood-based and model-based. One class of the neighborhood-based methods [22] aggregate similar users or items and predict the unobserved ratings based on the collected set of users or items [1, 21]. The other class of model-based approaches apply machine learning and data mining methods in the context of predictive models. For example, they assume that the sparse rating matrix R has low rank and hence matrix factorization techniques can be applied [12, 9, 17, 20, 13, 24, 18]. In particular, algorithms based on matrix factorization have attracted great attention because of their scalability and high prediction accuracy, demonstrated in the Netflix Prize competition [10]. In general, pointwise methods are efficient since their computational complexity scales linearly in the size of input data.

Pointwise approaches generally define user ratings as numerical values or categorical labels, and optimize regression-based or classification-based objective functions. In recent years, there have been many studies arguing that modeling ratings as numerical or categorical labels is improper for the task of item recommendation [11, 23, 3]. They propose

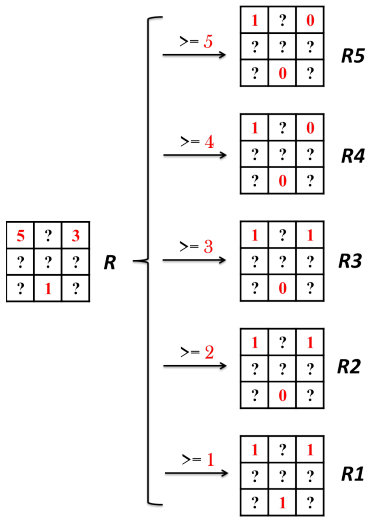

Figure 1: An example of matrix decomposition when $S = 5$. "?"s represent the unobserved entries. Each binary matrix in the right column is obtained by classifying the original rating matrix (left) based on the idea of ordinal classification. For example, the top right matrix, i.e., $R5$, is obtained by grouping observed entries: $r \geq 5$ as "1" and $r \leq 4$ as "0".

strategies to combine pairwise or listwise learning to rank methods with matrix factorization, where user ratings are defined as ordinal. For example, a listwise approach named Cofirank [24] optimizes a surrogate convex upper bound of NDCG error and matrix factorization is used as the basic rating predictor. Besides, Rendle et al. [19] and Liu et al. [15] model pairwise comparisons of observed ratings using Bradley-Terry model (a typical pairwise model) with low-rank structure. In [13], it is assumed that rating matrix R is locally low-rank and optimization is conducted on several pairwise surrogate ranking losses. In practice, these approaches which impose the ordinality of user ratings improve the ranking performance compared with general pointwise approaches. Therefore, in order to improve the ranking performance, we propose an approach to modeling the ordinality of user ratings in our pointwise method.

3. DECOUPLED COLLABORATIVE RANKING (DCR)

Our pointwise approach is based on two key ideas: (i) we consider rating scores as ordinal labels; (ii) focus more on higher rating scores in order to improve the ranking performance at the top of recommended list.

At the beginning of this section, we recall notations from previous sections: the rating matrix is denoted by $R \in \mathbb{R}^{m \times n}$ whose entry (rating) values are selected from a set of discrete ordered values $\{1, 2, ..., S\}$, with r_{ui} representing the observed rating of user u gives to item i.

3.1 Global Matrix Factorization

The basic idea of matrix factorization (MF) is that we assume R is low-rank and thus it can be approximated by $\hat{R} = UV^T$, where $U \in \mathbb{R}^{m \times f}$ and $V \in \mathbb{R}^{n \times f}$, f is the rank of approximation and $f \ll \min(m, n)$. The latent factors in U

and V can be obtained by optimizing a non-convex objective function. When it comes to ranking/recommendation, the items corresponding to unobserved entries of R are sorted in descending order of the numerical values obtained from matrix factorization. In this approach, the observed ratings in R are considered as real values.

3.2 Ordinal Classification

In our method, rating scores are considered as ordinal categorical labels. We merely concern the order among different rating scores. This ordinal view generally better reflects user feedback provided by qualitative rating scores.

We capture the ordinal nature of user ratings based on an ordinal classification method [14]. Given an S-level (i.e., $r \in \{1, 2, 3, ..., S\}$) rating matrix R, we decompose it into S binary matrices. The observed entry values in the t-th decoupled binary matrix ($t \in \{1, 2, ..., S\}$) are obtained by partitioning the ratings in R in the following manner: if $r \geq t$, then the entry is assigned a positive label "1" and if $r \leq t-1$, then a label "0" is assigned. The corresponding missing/unobserved entries in the decoupled binary matrices are still missing/unobserved. Finally, we obtain S binary matrices $R1$, $R2$,...,$R(S)$. A case example is shown in Figure 1 when $S = 5$. This idea originated from classical statistics, e.g., the so-called cumulative logit model [2].

3.2.1 Binary Classification

After matrix decomposition, we then model the ranking problem in each of the decoupled binary matrices as a binary classification problem. Assuming that each binary matrix has low rank, we apply matrix factorization and model the probability that r^t_{ui} is predicted as label "1" as follows:

$$P(r^t_{ui} = 1) = P(r_{ui} \geq t) = \frac{U^t_u {V^t_i}^T + 1}{2} \quad s.t. \quad \|U^t_u\| \leq 1, \quad \|V^t_i\| \leq 1 \tag{1}$$

r^t_{ui} is the label in the (u, i)-th entry of the t-th binary matrix. $P(r^t_{ui} = 1) = P(r_{ui} \geq t)$ since if $r_{ui} \geq t$, we will assign label "1" to the corresponding entry in t-th binary matrix. $U^t_u \in \mathbb{R}^f$ and $V^t_i \in \mathbb{R}^f$ are the latent factors of u-th user and i-th item in the t-th decoupled binary matrix. By imposing the constraint: $\|U^t_u\| \leq 1$ and $\|V^t_i\| \leq 1$, the predicted probability will locate in the range from 0 to 1.

Let Ω^t ($t \in \{1, 2, ..., S\}$) denote all the observed binary labels in the t-th decoupled binary matrix and let (u, i, r^t) denote one observed entry. The latent factors U^t_u and V^t_i for all the users and items can be learned by maximizing the log-likelihood on the training data:

$$\mathcal{L}^t = \sum_{(u,i,r^t) \in \Omega^t} \log P(r^t_{ui} = r^t | U^t_u, V^t_i)$$

Learning proceeds by stochastic gradient ascent on $\mathcal{L}^t$. Given a training example (u, i, r^t), the derivatives of the parameters are calculated as follows:

$$\frac{\partial \mathcal{L}^t}{\partial U^t_u} = \frac{1}{P(r^t_{ui} = r^t | U^t_u, V^t_i)} \frac{\partial P(r^t_{ui} = r^t | U^t_u, V^t_i)}{\partial U^t_u}$$

$$\frac{\partial \mathcal{L}^t}{\partial V^t_i} = \frac{1}{P(r^t_{ui} = r^t | U^t_u, V^t_i)} \frac{\partial P(r^t_{ui} = r^t | U^t_u, V^t_i)}{\partial V^t_i}$$

if r^t=1, $\frac{\partial P(r^t_{ui}=r^t|U^t_u,V^t_i)}{\partial U^t_u} = \frac{V^t_i}{2}$ and $\frac{\partial P(r^t_{ui}=r^t|U^t_u,V^t_i)}{\partial V^t_i} = \frac{U^t_u}{2}$;
if r^t=0, $\frac{\partial P(r^t_{ui}=r^t|U^t_u,V^t_i)}{\partial U^t_u} = -\frac{V^t_i}{2}$ and $\frac{\partial P(r^t_{ui}=r^t|U^t_u,V^t_i)}{\partial V^t_i} = -\frac{U^t_u}{2}$.

The constraints in Eq. (1) can be satisfied through gradient projection: $U^t_u \leftarrow \frac{U^t_u}{\|U^t_u\|}, V^t_i \leftarrow \frac{V^t_i}{\|V^t_i\|}$.

After learning the latent factors U^t_u and V^t_i, we can compute the probability of $P(r^t_{ui} = 1)$ (i.e., $P(r_{ui} \geq t)$) for all items through Eq. (1). We then sort all the items in descending order of their predicted probabilities (i.e., $P(r_{ui} \geq t)$) and recommend the top-N ranked items to users.

3.3 From Binary Classification to Ranking

In order to utilize all the information from each of the decoupled binary matrices, we need a mechanism to convert the binary classification results into a ranking score. We apply the following approach [14]:

$$SC_{ui} = \sum_{t=1}^{S} f(t) P(r_{ui} = t) \tag{2}$$

where $f(t)$ is a *relevance function* of t. Given a user u, we first compute the ranking score SC_{ui}, $\forall i$, and then sort all the items in descending order of SC_{ui}. Finally, we recommend the items at the top of ranked list to user u.

In Eq. (2), the probability distribution over expected item ratings can be obtained through a simple transformation from binary classifications as follows:

$$P(r_{ui} = t) = P(r_{ui} \geq t) - P(r_{ui} \geq t + 1) \tag{3}$$

Combining Eq. (2) and Eq. (3), we reformulate the scoring function for ranking as:

$$\begin{aligned} SC_{ui} &= \sum_{t=1}^{S} f(t)(P(r_{ui} \geq t) - P(r_{ui} \geq t + 1)) \\ &= f(1)P(r_{ui} \geq 1) + (f(2) - f(1))P(r_{ui} \geq 2) + ... \\ &\quad + (f(S) - f(S-1))P(r_{ui} \geq S) - f(S)P(r_{ui} \geq S + 1) \end{aligned}$$

Ratings are selected from $\{1, 2, ..., S\}$, and thus $P(r_{ui} \geq S + 1) = 0$. The above formulation can be written as:

$$SC_{ui} = f(1)P(r_{ui} \geq 1) + \sum_{t=2}^{S} (f(t) - f(t-1))P(r_{ui} \geq t) \tag{4}$$

We call the above method as **Decoupled Collaborative Ranking (DCR)**. The basic steps of DCR are summarized as follows:

1. collaboratively learn all the user and item latent factors $\{U^t_u, V^t_i\}$, $\forall u, i, t$ from each of the decoupled binary matrices through binary classification. The learning process in each of the binary matrices can be done in parallel;

2. calculate the ranking scores SC_{ui} for all users and items through Eq. (1) and Eq. (4);

3. for a user u, sort all the items in descending order of SC_{ui} and recommend items at the top of sorted list.

3.3.1 How to set $f(t)$ for ranking?

In this paper, we particularly want to improve the ranking performance at the top of recommended list. For this purpose, we propose to *emphasize more on the probabilities of higher rating scores*. It is motivated from empirical study that higher rating scores demonstrate a greater impact on top-N ranking performance.

We set $f(t)$ as a monotone increasing function of relevance level t. Consider two items A and B with probability distributions as follows (given that $r \in \{1, 2, 3, 4\}$):

Items	$P(r=1)$	$P(r=2)$	$P(r=3)$	$P(r=4)$
A	0.1	0.1	0.2	0.6
B	0.1	0.1	0.4	0.4

In the above example, we prefer item A to B, since item A is more likely to have a higher predicted rating score than B, and hence A should be put in front of B in the ranked list. For this purpose, we can choose any monotone increasing function as $f(t)$. In this paper, we investigate three typical monotone increasing functions: $f(t) = \log(t)$, $f(t) = t$, $f(t) = \exp(t)$, and comprehensively compare the performance of them (see Figure 3). We recommend to use $f(t) = t$ in terms of the ranking performance at the top of recommended list. To the best of our knowledge, there is no typical objective function for top-N recommendation formulated in the form of pointwise, and hence we are unable to learn the relevance function.

3.4 Empirical Study on Binary Classifications

We implement ranking algorithms on each of the binary matrices $R1$, $R2$,..., $R5$ ($r \in \{1, 2, ..., 5\}$) and test the performance of top-N recommendation in terms of NDCG@10 score on two datasets: Movielens100K and Netflix1M. Given one binary matrix, e.g., $R(t)$, we first calculate the probability of $P(r \geq t)$ for all the items whose rating scores are unobserved through binary classification on this binary matrix, and then sort these items in descending order of value $P(r \geq t)$. Finally, for each user we recommend the top-10 items in the ordered list. We also conduct global matrix factorization (see Section 3.1) on the original rating matrix R as the comparison. The performance is reported in Figure 2. There are two important observations:

(1) *Higher rating scores are more informative for top-N recommendation.* Let us compare rating scores "4" and "5" as an example. We know that the items in $R5$ are sorted in terms of the probability $P(r \geq 5)$ and hence observed rating score "4" is considered as negative label (i.e., "0") in $R5$, while the items in $R4$ are sorted in descending order of $P(r \geq 4)$ and hence rating score "4" is considered as positive label (i.e., "1"). In Figure 2, it demonstrates that the ranking performance of $R5$ is better than $R4$, which infers that mixing rating score "4" to "5" makes negative contribution to the ranking performance in terms of NDCG measure. This observation tells that "5"s are more important than "4"s.

(2) *It is not sufficient to rank items based on the information from a single binary matrix.* It is shown in Figure 2 that the ranking performance achieved through binary classification on a single binary matrix is worse than the performance achieved through matrix factorization on the original rating matrix. This result is understandable since each binary matrix only captures part information of the rating scores.

In a word, the first observation tells that we should emphasize more on higher rating scores for top-N recommendation and the second observation motivates us to propose a scoring function which should combine the results from the decoupled binary matrices. The scoring function in Eq. (4) satisfies both of these two requirements.

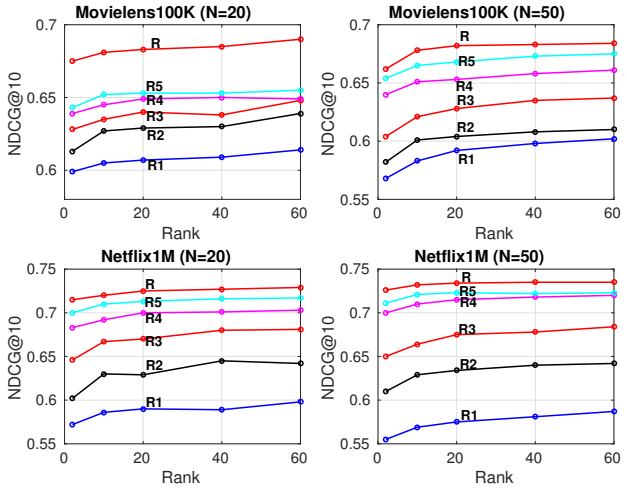

Figure 2: Comparison of ranking performance on individual rating matrices, including original rating matrix R and each of the decoupled binary matrices. "Rank" refers to the dimensionality for low-rank approximation.

4. EXTENSION: FURTHER IMPROVING RANKING VIA PAIRWISE LEARNING

In previous sections, we proposed the pointwise approach for item recommendation. As an extension, we leverage the learning-to-rank approach to further improving the ranking performance of DCR. Since this part concerns pairwise learning, we do not include it in the method of DCR. We propose it separately in case that users may need algorithms for more accurate recommendations. One issue with DCR is that the latent factors of all the users and items are learned through maximizing a log-likelihood in binary classification on each of the decoupled binary matrices, and it is not clear that these are the optimal features to use for the final ranking purpose. Ideally, one would expect to use features best suited for the final ranking task at hand. Hence, we propose to refine the features learned from DCR via optimizing a pairwise objective function.

The minimization of pairwise loss can lead to the maximization of ranking [4]. However, pairwise loss is usually not continuous and minimizing it is computationally intractable. Therefore, the popular way of optimization is to minimize a surrogate loss that forms a convex upper bound of the intractable loss. In this paper, we minimize the following loss:

$$\mathcal{E}(g) = \sum_{(u,i,j)\in\mathcal{O}} \mathcal{L}(Y_{uij} \cdot g(u,i,j)) + \lambda \sum_t (\|U^t\|_F^2 + \|V^t\|_F^2)$$
$$s.t. \quad \|U_u^t\|^2 \leq 1, \quad \|V_i^t\|^2 \leq 1 \quad \forall u, i, t \tag{5}$$

where $\mathcal{O}$ is the observed pairs of ratings and $(u, i, j) \in \mathcal{O}$ denotes one observed pair. Y_{uij} indicates the comparison of observed pairs such that if "$Y_{uij} = 1$" then user u prefers item i to item j and "$Y_{uij} = -1$" indicates that user u prefers item j to item i. $g(u, i, j) = SC_{ui} - SC_{uj}$, where SC_{ui} and SC_{uj} are the ranking scores and we choose $f(t) = t$ as the relevance function when computing ranking score. $\mathcal{L}$ is a non-increasing function and in our approach, we adopt

$\mathcal{L}(x) = \log(1+\exp(-x))$, which is often used in collaborative ranking (e.g., see [13, 16]).

We apply gradient descent method in the learning process. The first derivatives of $\mathcal{E}$ with respect to the latent factors in each binary matrix are calculated using chain rule:

$$\frac{\partial \mathcal{E}}{\partial U_u^t} = \sum_{(u,i,j)\in\mathcal{O}} \frac{\partial \mathcal{L}}{\partial g(u,i,j)} \frac{\partial g(u,i,j)}{\partial U_u^t} + 2\lambda U_u^t \tag{6}$$

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial V_i^t} = \sum_u \Bigg(& \sum_{j:r_{ui}>r_{uj}} \frac{\partial \mathcal{L}}{\partial g(u,i,j)} \frac{\partial g(u,i,j)}{\partial V_i^t} \\ & + \sum_{j:r_{ui}<r_{uj}} \frac{\partial \mathcal{L}}{\partial g(u,i,j)} \frac{\partial g(u,i,j)}{\partial V_i^t} \Bigg) + 2\lambda V_i^t\end{aligned} \tag{7}$$

It should be mentioned that there are two cases when calculating V_i^t: the derivative of V_i^t when i is preferred to j is different from that when j is preferred to i, which is shown in Eq. (7). The partial derivatives are:

$$\frac{\partial \mathcal{L}}{\partial g} = -\frac{Y_{uij}}{1+\exp(Y_{uij}g(u,i,j))}; \qquad \frac{\partial g}{\partial U_u^t} = \frac{V_i^t - V_j^t}{2};$$

$$\frac{\partial g}{\partial V_\alpha^t} = \begin{cases} \frac{1}{2}U_u^t & \text{if } \alpha = i \\ -\frac{1}{2}U_u^t & \text{if } \alpha = j \end{cases}$$

The full algorithm is shown in Algorithm 1.

Algorithm 1: Pairwise DCR

Input : $\{U_u^t, V_i^t\}$ learned from DCR, observed pairs of ratings $\mathcal{O}$; learning rate η; regularization parameter λ

Output: $\{U_u^t, V_i^t\}$, $\forall u, i, t$

```
while not converged do
    for all users u do
        find (u, i, j) ∈ O, all rating pairs by user u;
        calculate ΔU_u^t using Eq.(6);
        calculate ΔV_i^t using Eq.(7);
        U_u^t ← U_u^t − ηΔU_u^t;
        V_i^t ← V_i^t − ηΔV_i^t;
        U_u^t ← U_u^t / ||U_u^t||;   V_i^t ← V_i^t / ||V_i^t||
    end
end
```

5. EXPERIMENT

5.1 Experimental Settings

5.1.1 Datasets and Settings

Our algorithms are tested on four benchmark datasets: Movielens100K, Movielens1M, Movielens10M[1] and Netflix Prize dataset. The Movielens100K, Movielens1M, Movielens10M datasets are collected through the Movielens website during different periods. Netflix Prize dataset consists of three parts: training set, probe set and quiz set. The Netflix dataset in this paper refers to the first part. Netflix1M dataset is randomly sampled from Netflix training set. More statistics of these datasets are collected in Table 1.

[1]http://grouplens.org/datasets/movielens/

Table 1: Popular datasets used in the experiment

Datasets	Users	Items	Scale	Ratings
Movielens100K	943	1,682	1 - 5	100,000
Movielens1M	6,040	3,706	1 - 5	1,000,209
Movielens10M	71,567	10,681	0.5 - 5.0	10,000,054
Netflix1M	48,018	1,777	1 - 5	1,020,752
Netflix	480,189	17,770	1 - 5	100,480,507

We follow a popular setup [24, 13, 7] to partition each dataset into training and test sets. For each user in the dataset, we randomly select N items as training data and all the remaining items are used as test data. Therefore, users who have not rated $N+10$ will be dropped to guarantee that there would be at least 10 items in the test set for each user. In the experiments, we adopt the same settings in [24, 13, 7] by choosing N: 10, 20, 50.

5.1.2 Performance Metrics

In our experiments, we evaluate our proposed algorithms by Normalized Discounted Cumulative Gain (**NDCG**) [8], which is probably the most popular ranking metric for capturing the importance of retrieving good items at the top of ranked lists. It is formally given by:

$$NDCG@K(u) = \frac{DCG@K(u,\pi_u)}{DCG@K(u,\pi_u^*)}$$

where

$$DCG@K(u,\pi_u) = \sum_{k=1}^{K} \frac{2^{r_{u\pi_u(k)}} - 1}{\log_2(k+1)}$$

π_u is a permutation of items for user u, and π_u^* is the permutation that generates the maximum of $DCG@K$. $\pi_u(k)$ is the index of the k-th ranked item generated by our ranking model. In accordance with the setting of datasets, the largest value of K is 10.

5.2 Empirical study on DCR

In this section, we conduct empirical study on DCR. A crucial problem in DCR is how to choose the relevance function $f(t)$. It is explained in Section 3.3.1 that we should choose a monotone increasing function for the purpose of top-N recommendation. We investigate three typical monotone increasing functions, including: "$f(t) = \log(t)$", "$f(t) = t$", and "$f(t) = \exp(t)$".

We report the ranking performance of these three relevance functions as a function of *rank* (*rank* refers to the dimension of user and item latent factors) in terms of NDCG@10 scores in Figure 3.

5.2.1 Comparisons of relevance functions

From the results in Figure 3, we can clearly observe that "$f(t) = t$" performs the best, and "$f(t) = \log(t)$" performs slightly worse than "$f(t) = t$", while "$f(t) = \exp(t)$" performs the worst in most of the test cases, which tells that we should not choose a steeply increasing function which emphasizes "excessively" on the higher rating values. This observation can be explained in that if we extremely emphasize on the highest rating score, then we may lose too much information from the lower rating scores. The most extreme case is that we set the weight for the highest rating score as 1

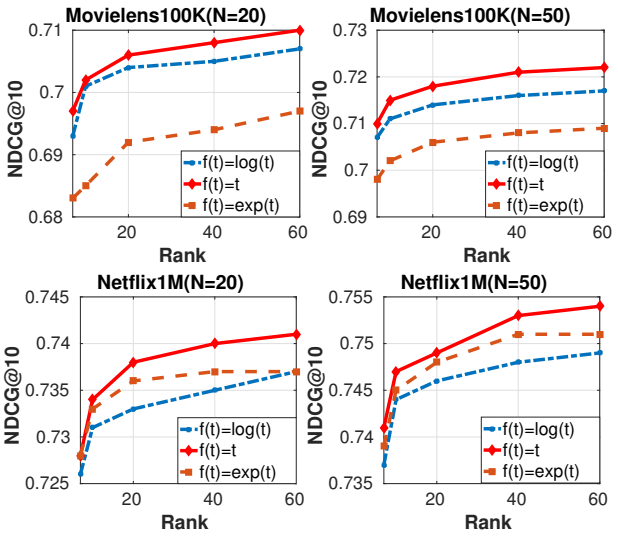

Figure 3: Comparisons on the ranking performance of different relevance functions as a function of $rank$.

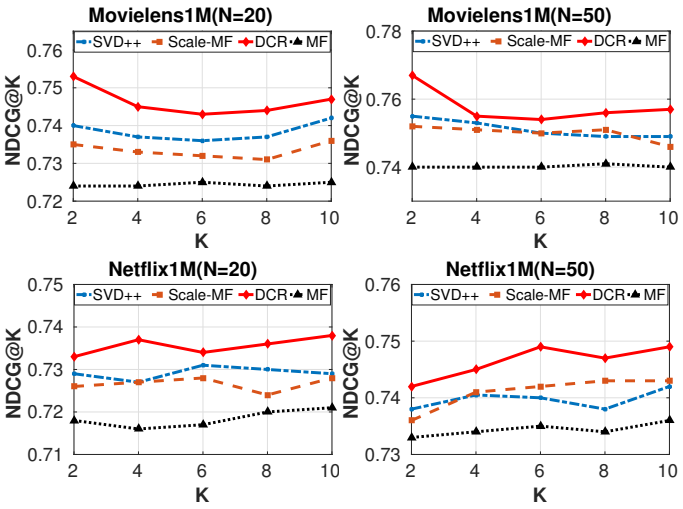

Figure 4: Comparisons with pointwise methods in terms of NDCG@K scores. K varies from 2 to 10. $rank = 20$ in DCR and $rank = 100$ in other methods.

while the weights for all the other scores are set as 0. Therefore, we only need to calculate the probability of $P(r = S)$, which can be obtained through binary classification on a single binary matrix $R(S)$. However, from the results in Figure 2, it is suggested to avoid ranking items based on the information from a single binary matrix. In terms of the empirical ranking performance, we choose "$f(t) = t$" as the relevance function of DCR.

It should be mentioned that it makes no sense to emphasize equally on all the ordered scores, i.e, choose $f(t) = c$ (c is a constant). If we do so, then Eq. (4) reduces to: $SC_{ui} = f(1)P(r_{ui} \geq 1)$, which merely considers $R1$. From the observation in Figure 2, this setting should be avoided.

5.2.2 *Effect of Parameter* $rank$

From the observations in Figure 2 and Figure 3, we can see that increasing the $rank$ can always improve the performance. From this point of view, we would prefer a higher $rank$. However, if we constantly increase the dimension of latent factors, we also increase the computational cost. In particular, in the Pairwise DCR method, we should not choose a too large $rank$ since pairwise learning is much more time consuming than pointwise approaches.

5.3 Compare with Pointwise Approaches

We compare DCR with several pointwise methods: (1) **MF**: the global matrix factorization as a baseline method; (2) **SVD++**: SVD++ is [9] considered as one of the most accurate rating predictors. (3) **Scale-MF**: we first scale the values of observed ratings by $y_{ui} = 2^{r_{ui}} - 1$ and conduct the global matrix factorization on the scaled rating scores. This scaling of rating can better reflect NDCG [6].

We demonstrate the performance of all the compared methods in terms of NDCG@K, where "K" is a variable, ranging from 2 to 10. We conduct experiments on two datasets using two different settings of partitions on training and test data. In order to make pair comparisons in the setting of same number of latent factors, we set $rank = 20$ for DCR and $rank = 100$ for other methods, since our method has S times more parameters than a single matrix factorization where $S = 5$ in these test datasets. The results are reported in Figure 4.

It demonstrates that DCR outperforms other pointwise methods. The difference between DCR and other methods is that we model the ordinality of user ratings in DCR and all the other methods treat user ratings as numerical values. Therefore, defining user ratings as ordinal more appropriately reflects the degree of user references. Besides, by comparing the performance of scale-MF with MF, we observe that reasonably scaling rating values can further improve the top-N ranking performance.

5.4 Compare with Push Collaborative Ranking

In this section, we compare DCR with a state-of-art collaborative ranking algorithm-Push Collaborative Ranking [5], which also aims at improving the ranking performance at the top of recommended list. We compare the ranking performance of DCR with three push algorithms: collaborative p-norm push, infinite push, and reverse-height push, on Movielens100K and Movielen1M datasets in terms of NDCG@5 and NDCG@10 scores. In each dataset, we choose $N = 20$ ratings as training samples and the other ratings are used as test data. We set $rank = 20$ in DCR and $rank = 100$ in push algorithms. For each method, we conduct 5 times of individual experiments and the average score is reported. The result is shown in the following table, which demonstrates the superiority of our approach in terms of the ranking performance.

Datasets	Movielens100K		Movielens1M	
Method	NDCG@5	NDCG@10	NDCG@5	NDCG@10
Inf-Push	0.6652	0.6733	0.7157	0.7210
RH-Push	0.6720	0.6823	0.7182	0.7225
P-Push	0.6530	0.6620	0.7104	0.7156
DCR	**0.6931**	**0.7073**	**0.7441**	**0.7432**

Table 2: Comparisons with pairwise and listwise approaches in terms of NDCG@10 (the higher the better). We report the average and standard deviation over 5 times of independent experiments. Values in bold-face indicate the best result among all the algorithms. Ranking performance of LCR on Netflix dataset is not reported since LCR method is very slow and running it on Netflix data takes too much time.

Datasets	Methods	N=10	N=20	N=50
Movielens100K	CofiRank	0.6625 ± 0.0023	0.6933 ± 0.0018	0.7021 ± 0.0031
	BT	0.6342 ± 0.0038	0.6487 ± 0.0052	0.7061 ± 0.0021
	LCR	0.6623 ± 0.0028	0.6680 ± 0.0028	0.6752 ± 0.0024
	DCR	**0.6901 ± 0.0012**	**0.7082 ± 0.0035**	**0.7241 ± 0.0021**
Movielens1M	CofiRank	0.7041 ± 0.0023	0.7233 ± 0.0013	0.7256 ± 0.0042
	BT	0.6752 ± 0.0021	0.7104 ± 0.0017	0.7528 ± 0.0030
	LCR	0.6978 ± 0.0031	0.7012 ± 0.0025	0.7252 ± 0.0018
	DCR	**0.7261 ± 0.0025**	**0.7431 ± 0.0027**	**0.7622 ± 0.0013**
Movielens10M	CofiRank	0.6902 ± 0.0012	0.7050 ± 0.0032	0.6971 ± 0.0015
	BT	0.7106 ± 0.0024	0.7160 ± 0.0032	0.7352 ± 0.0024
	LCR	0.6921 ± 0.0024	0.6877 ± 0.0027	0.6854 ± 0.0035
	DCR	**0.7132 ± 0.0017**	**0.7251 ± 0.0021**	**0.7421 ± 0.0018**
Netflix1M	CofiRank	0.7090 ± 0.0023	0.7188 ± 0.0034	0.7111 ± 0.0015
	BT	0.7183 ± 0.0024	0.7174 ± 0.0018	0.7451 ± 0.0031
	LCR	0.7014 ± 0.0026	0.7040 ± 0.0023	0.6847 ± 0.0029
	DCR	**0.7351 ± 0.0032**	**0.7381 ± 0.0024**	**0.7522 ± 0.0032**
Netflix	CofiRank	0.6615 ± 0.0051	0.6927 ± 0.0034	0.7058 ± 0.0054
	BT	0.7121 ± 0.0021	0.7320 ± 0.0041	0.7319 ± 0.0024
	LCR	-	-	-
	DCR	**0.7801 ± 0.0021**	**0.7914 ± 0.0021**	**0.8001 ± 0.0007**

5.5 Compare with Pairwise and Listwise Approaches

In this section, we compare DCR with pairwise and listwise approaches, where user ratings are modeled as ordinal. It includes the comparisons with:

- Pairwise models: (i)**Bradley-Terry model(BT)**. BT model is the most widely used method for modeling pairwise user preferences. It optimizes a pairwise objective function as:

$$-\sum_{(u,i,j)\in\Omega} \log \frac{\exp(U_u V_i^T)}{\exp(U_u V_i^T) + \exp(U_u V_j^T)} + \lambda_\Theta \|\Theta\|^2$$

  where $\Omega = \{(u,i,j) : r_{ui} > r_{uj}\}$ is the observed set of pairs of preferences and $\lambda_\Theta \|\Theta\|^2$ is the regularization term for all model parameters (i.e., U and V). Actually, the BT method is almost the same with **Bayesian Personalized Ranking (BPR)** model [19]. (ii)**Local Collaborative Ranking (LCR)** [22] is another collaborative ranking method which also optimizes a pairwise ranking loss. In LCR, R is approximated by many locally low-rank matrices.

- Listwise approach: **CofiRank** [24] also known as maximum margin matrix factorization is a famous collaborative ranking algorithm and it is always considered as a strong baseline method for collaborative ranking. It directly optimizes a surrogate convex upper bound of the NDCG error.

Settings In the experiment, we set the number of ratings for training $N = 10, 20, 50$, which is often used in collaborative ranking literature [22, 16, 24]. Besides, for each method, we conduct 5 times of independent experiments and report the average and standard deviation of NDCG@10. All the source code for the baseline methods can be found on the websites together with their original papers. We compare DCR with other methods in the setting of same number of latent factors, that is, if we set $rank = k$ for DCR, then $rank = S \cdot k$ for other methods, where S is the number of different ordered rating scores in a specific data set. In the experiments, we choose $rank = 40$. The regularization parameter of DCR is chosen from $\{0.01, 0.001\}$, and the initial learning rate is chosen from $\{0.05, 0.04, 0.03, 0.02, 0.01\}$ and the best ranking performance is reported.

5.5.1 Performance Comparisons

An extensive performance comparison of all the methods is shown in Table 2. We can see from the results that our proposed DCR method performs the best on all the datasets. In most of the test cases, the DCR method can improve the ranking performance by 2% to 6% against the best-performing baseline method. In particular, in Netflix dataset when $N = 10$, our method achieves over 10% performance improvement. Based on the t-test results, it is noticeable that on all datasets, the improvements of our ranking algorithm against all the other baseline approaches are statistically significant at the p-value<0.01. Besides, our proposed approach could consistently perform well in different settings of N.

5.5.2 Comparisons of Efficiency

We also report the running time (seconds) for the comparison methods on Netflix1M dataset in Table 3. The running time of Cofirank, LCR is obtained based on the software implemented by their authors. From the observation in Table 3, we see that our proposed method can run more than one hundred times faster than several algorithms, such as

Table 3: Running time (seconds)

Methods	CofiRank	BT	LCR	DCR
N=20	399.0	130.6	602.2	1.05
N=50	898.1	269.4	1232.1	2.31

LCR and CofiRank. The results are understandable in that DCR is a pointwise approach, which is considered more computationally efficient than pairwise and listwise approaches. Based on the ranking performance and time efficiency, DCR can be considered as a notable approach for practical usage.

5.6 Pairwise DCR and Compare with Improved-BT

We report the performance of Pairwise DCR and compare it with Improved Bradley-Terry model (**Improved-BT**) [7]. Pairwise DCR is an approach that first obtains the latent features through the pointwise DCR, and then refines these obtained features through optimizing a pairwise model, and thus Pairwise DCR can be considered as a combination of pointwise and pairwise methods. We compare it with Improved-BT, which can also be considered as a hybrid approach combining both of pointwise and pairwise collaborative ranking methods. We report the performance of DCR as a comparison. The ranking performance in terms of NDCG@10 on Movielens1M and Netflix1M datasets is shown in the following table when we choose N=10 and 20.

Datasets	Movielens1M		Netflix1M	
Method	N=10	N=20	N=10	N=20
Improved-BT	0.7368	0.7511	0.7355	0.7400
DCR	0.7261	0.7431	0.7351	0.7381
Pairwise DCR	**0.7471**	**0.7596**	**0.7495**	**0.7552**

Comparing the results of DCR with Pairwise DCR, we can conclude that refining the latent features of DCR by pairwise learning improves the ranking performance. On the other hand, Improved-BT, a model which combines pointwise and pairwise learning, slightly outperforms DCR. Based on this observation, it may infer that combining different collaborative ranking approaches (e.g., combine pointwise with pairwise methods) could be a new direction for improving the performance of collaborative ranking. We can also observe that our proposed Pairwise DCR method outperforms Improved-BT. In Figure 2, we have concluded that higher rating scores are more important than lower rating scores for top-N recommendation. Therefore, in Pairwise DCR higher rating values are emphasized more than lower rating values, while there is no such knowledge included in the Improved-BT model.

6. DISCUSSIONS ON DCR-LOGISTIC

Readers may ask why we use a predictive model as that in Eq. (1), rather than a seemingly more popular logistic mapping function as follows:

$$P(r_{ui} \geq t) = \frac{1}{1 + \exp(-U_u^t V_i^{tT})} \quad (8)$$

Indeed, in this paper we also implemented it using Eq. (8) (we call it **DCR-Logistic**). We report the ranking performance in terms of NDCG@10 in Figure 6 on Movielens1M and Netflix1M datasets when N is set as 50. We observe that DCR outperforms DCR-Logistic. To some extent, this result can be explained in that in Eq. (1), the parameters are bounded in a unit ball (i.e., $\|U_u^t\| \leq 1, \quad \|V_i^t\| \leq 1$), while the logistic link function has no constraints on the parameters and hence it might be prone to overfit. Even though we can set more constraints to the parameters in DCR-Logistic to avoid overfitting, this strategy will make this model more complicated. In particular, if we replace Eq. (1) by Eq. (8) in Pairwise DCR and add more constraints to the parameters, then the optimization of Pairwise DCR model will be quite computationally expensive.

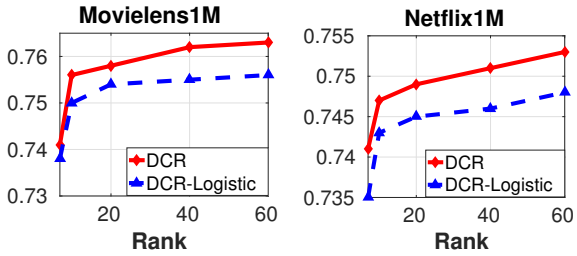

Figure 5: Compare DCR with DCR-Logistic.

7. CONCLUSION

In this paper, we focus on improving the ranking performance at the top of recommended list. We propose a pointwise collaborative ranking approach, named decoupled collaborative ranking (DCR). Different from common pointwise methods which consider user ratings as numerical values or categorical labels, in our method, we view rating scores as ordinal. For this purpose, we decompose the user-item rating matrix R into a sequence of binary matrices based on an ordinal classification method. After obtaining the predictions from each of the binary matrices, we finally form a ranking score through the weighted summation of the results from binary classifications. The weights are set for the purpose that we can emphasize more on higher rating scores, since we observe from empirical study that higher rating scores are more important than lower rating scores for top-N recommendation. In the end, as an extension of the DCR model, we improve the ranking performance of our method through optimizing a pairwise objective function. It is shown in the experiments that our proposed method can significantly outperforms many state-of-the-art recommendation algorithms in terms of the NDCG metric. Besides, our proposed method, as a pointwise approach, shows better performance in terms of time efficiency compared to pairwise or listwise approaches.

Acknowledgements

The work is partially supported by NSF-Bigdata-1419210 and NSF-III-1360971.

8. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[2] A. Agresti. *Categorical Data Analysis*. John Wiley & Sons, Inc., Hoboken, NJ, second edition, 2002.

[3] S. Balakrishnan and S. Chopra. Collaborative ranking. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 143–152. ACM, 2012.

[4] W. Chen, T.-Y. Liu, Y. Lan, Z.-M. Ma, and H. Li. Ranking measures and loss functions in learning to rank. In *Advances in Neural Information Processing Systems*, pages 315–323, 2009.

[5] K. Christakopoulou and A. Banerjee. Collaborative ranking with a push at the top. In *Proceedings of the 24th International Conference on World Wide Web*, pages 205–215. ACM, 2015.

[6] D. Cossock and T. Zhang. Statistical analysis of bayes optimal subset ranking. *Information Theory, IEEE Transactions on*, 54(11):5140–5154, 2008.

[7] J. Hu and P. Li. Improved and scalable bradley-terry model for collaborative ranking. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 949–954. IEEE, 2016.

[8] K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *SIGIR*, pages 41–48, Athens, Greece, 2000.

[9] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[10] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[11] Y. Koren and J. Sill. Ordrec: an ordinal model for predicting personalized item rating distributions. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 117–124. ACM, 2011.

[12] O. Koyejo, S. Acharyya, and J. Ghosh. Retargeted matrix factorization for collaborative filtering. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 49–56. ACM, 2013.

[13] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer. Local collaborative ranking. In *Proceedings of the 23rd international conference on World wide web*, pages 85–96. International World Wide Web Conferences Steering Committee, 2014.

[14] P. Li, C. J. Burges, and Q. Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in neural information processing systems*, pages 897–904, 2007.

[15] N. N. Liu, M. Zhao, and Q. Yang. Probabilistic latent preference analysis for collaborative filtering. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 759–766. ACM, 2009.

[16] D. Park, J. Neeman, J. Zhang, S. Sanghavi, and I. S. Dhillon. Preference completion: Large-scale collaborative ranking from pairwise comparisons. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1907–1916, 2015.

[17] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

[18] S. Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57, 2012.

[19] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

[20] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM, 2008.

[21] Y. Shi, M. Larson, and A. Hanjalic. Exploiting user similarity based on rated-item pools for improved user-based collaborative filtering. In *Proceedings of the third ACM conference on Recommender systems*, pages 125–132. ACM, 2009.

[22] Y. Shi, M. Larson, and A. Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, 47(1):3, 2014.

[23] H. Steck. Evaluation of recommendations: rating-prediction and ranking. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 213–220. ACM, 2013.

[24] M. Weimer, A. Karatzoglou, Q. V. Le, and A. J. Smola. COFI RANK - maximum margin matrix factorization for collaborative ranking. In *Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, 2007*, pages 1593–1600, 2007.