2016 IEEE 16th International Conference on Data Mining

Improved and Scalable Bradley-Terry Model for Collaborative Ranking

Jun Hu
Department of Computer Science
Rutgers University
Piscataway, NJ 08854, USA
jh900@cs.rutgers.edu

Ping Li
Department of Statistics and Biostatistics
Department of Computer Science
Rutgers University
pingli@stat.rutgers.edu

***Abstract*—In[1] collaborative ranking, the Bradley-Terry (BT) model is widely used for modeling pairwise user preferences. However, when this model is combined with matrix factorization on sparsely observed ratings, a challenging *identifiability* issue arises since the optimization will involve non-convex constraints. Besides, in some situations, fitting the Bradley-Terry model yields a *numerical challenge* as it may include an objective function that is unbounded from below. In this paper, we will discuss and develop a simple strategy to resolve these issues. Specifically, we propose an Improved-BT model by adding a penalty term, and we develop two parallel algorithms to make Improved-BT model scalable. Through extensive experiments on benchmark datasets, we show that our proposed method outperforms many considered state-of-the-art collaborative ranking approaches in terms of both ranking performance and time efficiency.**

I. Introduction

In this paper, we consider a basic setting of recommender systems: a set of ratings are recorded by m users over n items, and represented by an $m \times n$ user-item (U-I) matrix $R \in \mathbb{R}^{m \times n}$. It is a common situation that the ratings take discrete values, e.g., from 1 to 5, and most of them are absent due to incomplete observations, making R a highly sparse matrix. The task of recommendation in such setting boils down to selecting the items which are potentially interesting to users based on the predictions of the unobserved entries.

Collaborative filtering (CF) is a popular method for recommender systems. There are many CF approaches proposed within the last decade, especially during the period of the Netflix Prize competition [19]. In the literature of CF, matrix factorization (MF) might be the most popular method due to its scalability and prediction accuracy. In matrix factorization, the sparse rating matrix R is assumed to be low-rank and hence can be decomposed into the product of two low-dimensional matrices. Another class of interesting methods for CF [18] propose to aggregate similar users or items and predict the unobserved ratings based on the collected similar users or items (e.g., [1], [17]). Most of the CF methods commonly aim at improving the rating prediction accuracy, which is often measured by root-mean-square error (RMSE).

In modern recommender systems, however, it is considered more crucial to investigate the ranking performance, especially at the top of the ranked lists of items. Recently, it has been widely proposed to combine learning-to-rank (LTR) [10], [6], [5], [16] strategies with matrix factorization [3], forming a new class of collaborative ranking methods.

In collaborative ranking, pairwise models are popular and the Bradley-Terry (BT) model is one of the most popular approach to modeling pairwise preferences. In many other fields (e.g., learning-to-rank) where the BT model is widely applied, in order to ensure the **identifiability** of model parameters, the so-called sum-to-zero constraint is typically adopted in order to control the range of the model parameters [9], [2]. However, when the Bradley-Terry model is combined with matrix factorization, the corresponding constraint becomes non-convex, rendering the resulting optimization problem challenging. Besides, if we apply the Bradley-Terry model without any modification, we may encounter an implicit **numerical challenge**: suppose that the pairwise user preferences are represented as graph and in some cases, adding or subtracting a same constant to some specific nodes will not change the preference graph, however the value of the objective function of the Bradley-Terry model can go to infinity.

In order to resolve the identifiability issue and the numerical challenge, we propose to add a penalty term to the objective function of the Bradley-Terry model, a modification that we call **Improved-BT** model. Our work is inspired by the LTR literature in particular the work on combining pointwise regression with pairwise model [16]. We also develop parallel solvers to make Improved-BT model scalable. We show, through extensive experiments on benchmark datasets, that the Improved-BT model can achieve considerable improvements on ranking performance and time efficiency compared to state-of-the-art rankers.

II. Related Work on Collaborative ranking

For the purpose of achieving good ranking performance, learning-to-rank (LTR) methods [10], [6], [5], [16] have been widely used in recommender systems, forming a new category of collaborative ranking methods [3]. For example, Rendle et al. [15] and Liu et al. [11] model pairwise comparisons of observed ratings using Bradley-Terry model with low-rank structure. Besides, in [12], the authors formulated an objective function based on the Bradley-Terry model and developed a

[1]Supported by NSF-Bigdata-1419210 and NSF-III-1360971. Code is available at https://github.com/bssbbsmd/Collaborative-Ranking .

2374-8486/16 $31.00 © 2016 IEEE
DOI 10.1109/ICDM.2016.161

large-scale non-convex implementation that trains a factored form of the matrix via alternating minimization. There are also many other interesting models for collaborative ranking: CofiRank [20] is a notable approach which optimizes a surrogate convex upper bound of NDCG error and uses matrix factorization as the basic rating predictor. Another interesting approach assumes that the rating matrix is locally low-rank [8] and optimizes pairwise surrogate ranking losses.

III. Methodology and Strategy

In this section, we first introduce the Bradley-Terry (BT) model and explain how BT model can be applied in collaborative ranking, and then point out the issues when the Bradley-Terry model is combined with matrix factorization. We also present an effective strategy to deal with the proposed issues.

A. Bradley-Terry Model

In 1952, Bradley and Terry [4] proposed a logit model for paired evaluations. Let $P(i > j)$ denote the probability that item i is preferred over item j. The Bradley-Terry (BT) model is given by: $\log \frac{P(i>j)}{P(j>i)} = \beta_i - \beta_j$. Alternatively, we can write

$$P(i > j) = \frac{\exp(\beta_i)}{\exp(\beta_i) + \exp(\beta_j)} = \frac{\exp(\beta_i - \beta_j)}{1 + \exp(\beta_i - \beta_j)} \quad (1)$$

where β_i and β_j are relevance scores for item i and item j.

B. Bradley-Terry Model Meets Matrix Factorization

In recommender systems, the relevance scores β_i and β_j in Eq. (1) are often referred to as rating values. Assuming that the rating matrix $R \in \mathbb{R}^{m\times n}$ is of low rank, β_i and β_j can be obtained through matrix factorization (MF) in that R can be approximated by the product of two matrices $U \in \mathbb{R}^{m\times f}$ and $V \in \mathbb{R}^{n\times f}$: $\hat{R} = UV^T$, where f is the dimensionality/rank of the approximation. Each row of U and V describe the latent features for a specific user u and an item i respectively. Thus, the estimated rating a user u gives to an item i could be obtained by $\hat{r}_{ui} = \langle U_u, V_i \rangle$, where U_u is the uth row of U, V_i is the ith row of V, and $\langle \cdot, \cdot \rangle$ is the dot product.

Let us denote the observed set of ratings as O and the observed set of pairs of user preferences as $\Omega = \{(u, i, j) : r_{ui} > r_{uj}, r_{ui} \in O, r_{uj} \in O\}$ where r_{ui} is an observed rating. By replacing the reference scores β_i and β_j in Eq. (1) with estimated rating values $\hat{r}_{ui}$ and $\hat{r}_{uj}$, the pairwise ranking aggregation on the observed pairs can be directly formulated as minimizing the negative log-likelihood function, i.e.,

$$\begin{aligned} \text{BT-OPT} &:= -\log \prod_{(u,i,j)\in\Omega} P(\hat{r}_{ui} > \hat{r}_{uj}) \\ &= -\sum_{(u,i,j)\in\Omega} \log P(\hat{r}_{ui} > \hat{r}_{uj}) \\ &= -\sum_{(u,i,j)\in\Omega} \log \frac{\exp(\hat{r}_{ui})}{\exp(\hat{r}_{ui}) + \exp(\hat{r}_{uj})} \\ &= -\sum_{(u,i,j)\in\Omega} \log \frac{\exp(U_u V_i^T)}{\exp(U_u V_i^T) + \exp(U_u V_j^T)} \end{aligned} \quad (2)$$

The objective function in Eq. (2) has been widely used in collaborative ranking for modeling pairwise preferences [12], [3]. However, if we apply matrix factorization to the Bradley-Terry model without any modification, then we will encounter an **identifiability** issue as well as a **numerical challenge**.

1) Identifiability issue (and non-convex constraints): In Eq. (2), if we add a same constant to $\hat{r}_{ui}$ and $\hat{r}_{uj}$, then the likelihood will remain the same. At a technical level, one says that the parameters are not identifiable. Similar identifiability issues exist in other common problems, such as multiclass logistic regression [9].

In order to deal with this issue, the usual approach is to impose a sum-to-zero constraint to the model [9], [2]. In our case, assuming that there are n items, this yields constraints $\sum_{i=1}^{n} \hat{r}_{ui} = 0$ for all u. In terms of the optimization variables U and V, we accordingly obtain the constraints $\sum_{i=1}^{n} U_u V_i^T = 0$ for all u. These constraints are non-convex and thus difficult too handle from an optimization viewpoint. In particular, popular optimization methods used in this context such as (stochastic) gradient descent or alternating optimization of U and V are no longer applicable. As a result, dealing with non-identifiability becomes more challenging when the Bradley-Terry model is combined with matrix factorization.

2) Numerical challenge (no minimizer of objective function): As illustrated by Figure 1, for better visualization and interpretation, pairwise comparisons can be represented by a directed graph (item as vertex and pairwise relationship as edge). If item i is preferred to item j, then we draw a directed edge from node i to node j and vice versa. If item i and item j have the same level of preference, then we draw both directions. We show that if we directly apply the Bradley-Terry model without any modification, we will encounter a numerical issue in that we cannot obtain a minimizer of the negative log-likelihood in Eq. (2). For example, given a preference graph as shown in Figure 1, for any solution β_A, β_B, β_C, β_D which satisfies the comparison relationship in the figure, if we add a positive constant c to node A and B and subtract the same constant from C and D, then the preference graph does not change. However, the likelihood corresponding to A$\longrightarrow$D (i.e., $\frac{\exp(\beta_A+c)}{\exp(\beta_A+c)+\exp(\beta_D-c)}$) and B$\longrightarrow$C (i.e., $\frac{\exp(\beta_B+c)}{\exp(\beta_B+c)+\exp(\beta_C-c)}$) will always increase if we increase c, while the likelihood corresponding to A$\longleftrightarrow$B and C$\longleftrightarrow$D does not change. Therefore, the negative likelihood is unbounded from below, and consequently has no minimizer.

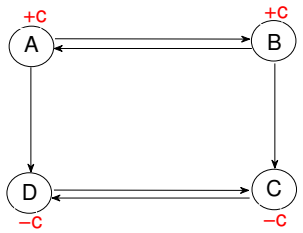

Fig. 1. The preference graph of an example to show the numerical challenge.

C. Regularization

To avoid overfitting, regularization is popular in MF-based collaborative filtering methods. The following is a common way of regularization:

$$\text{BT-OPT(1)} := -\sum_{(u,i,j)\in\Omega} \log \frac{\exp(U_u V_i^T)}{\exp(U_u V_i^T) + \exp(U_u V_j^T)} + \sum_{u=1}^{m} \lambda_{U,u} \|U_u\|_F^2 + \sum_{i=1}^{n} \lambda_{V,i} \|V_i\|_F^2$$

where $\|\cdot\|_F^2$ represents the squared Frobenius norm. While regularization is a often good idea to prevent overfitting in matrix factorization, it cannot directly resolve the issues discussed earlier. In fact, both of the identifiability issue and numerical challenge result from the shift of the values $U_u V_i^T$ and $U_u V_j^T$, while regularization constrains U_u and V_i separately.

D. *Improved Bradley-Terry Model*

We propose a simple and yet effective way to deal with the aforementioned: **identifiability issue** and **numerical challenge**. In fact, these two issues result from the shift of the estimated rating values $\hat{r}_{ui}$ and $\hat{r}_{uj}$ (i.e., $U_u V_i^T$ and $U_u V_j^T$): (I) identifiability issue occurs because adding a same constant to both of the estimated rating values will not affect the objective function; (II) the numerical problem occurs because adding and subtracting a same constant from specific nodes will not change the preference graph (e.g., an example is shown in Figure 1), however the likelihood can go to infinity. Hence, we propose to restrain the shift of $U_u V_i^T$ and $U_u V_j^T$. For this purpose, we reformulate the objective function by adding a penalty term as follows:

$$\text{BT-OPT(2)} := \sum_{(u,i,j)\in\Omega} \Big\{ \underbrace{-\log \frac{\exp(U_u V_i^T)}{\exp(U_u V_i^T) + \exp(U_u V_j^T)}}_{\text{pairwise loss}} + \gamma \underbrace{\big((r_{ui} - U_u V_i^T)^2 + (r_{uj} - U_u V_j^T)^2\big)}_{\text{penalty}} \Big\} + \sum_{u=1}^{m} \lambda_{U,u} \|U_u\|_F^2 + \sum_{i=1}^{n} \lambda_{V,i} \|V_i\|_F^2 \quad (3)$$

where $\gamma \geq 0$ is a balance factor between the pairwise loss and additive penalty term. By setting a penalty term into the Bradley-Terry model will make the estimated rating values converge to the real observed rating values and hence the situation that those estimated rating values uncontrollably shift cannot happen. We call this new method as improved Bradley-Terry model (**Improved-BT**).

The additive penalty term in Eq. (3) is actually different from the traditional regression loss in the way of accessing input data, as it accesses observed ratings in the form of pairwise while traditional regression loss usually accesses data in the form of single point. While this paper mainly discusses the pairwise BT model, we should mention that if we delete the pairwise term in Eq. (3), we can still obtain reasonable ranking results by solely optimizing the penalty term.

IV. Learning

Methods based on stochastic gradient descent (SGD) have become increasingly important, especially for large-scale industrial applications. In this paper, we develop SGD algorithms to solve the Improved-BT model. Besides, we adopt a notable share-memory parallel algorithm-HOGWILD! to make the learning process of our proposed method scalable to multicore machines. More details on the provably guarantees or analysis about HOGWILD! refer to the paper [13].

A. Parallel Stochastic Gradient Descent

We first apply the general SGD algorithm for solving the Improved-BT model. In practice, we set different regularization parameters for each user and item as:

$$\lambda_{U,u} = \frac{\lambda}{\#\text{observed pairs which contain user } u}$$

$$\lambda_{V,i} = \frac{\lambda}{\#\text{observed pairs which contain item } i}$$

As a result, we only have one hyperparameter γ for the regularization parameters. This is a popular strategy to reduce the number of regularization parameters. Let $\hat{r}_{ui} = U_u V_i^T$, $\hat{r}_{uj} = U_u V_j^T$, for any training example $(u,i,j) \in \Omega$, partial derivatives of the model parameters are calculated as:

$$\frac{\partial \text{BT-OPT(2)}}{\partial U_u} = -\frac{1}{1+\exp(\hat{r}_{ui} - \hat{r}_{uj})}(V_i - V_j) + 2\lambda_{U,u} U_u + \gamma\big(-2V_i(r_{ui} - \hat{r}_{ui}) - 2V_j(r_{uj} - \hat{r}_{uj})\big)$$

$$\frac{\partial \text{BT-OPT(2)}}{\partial V_i} = -\frac{1}{1+\exp(\hat{r}_{ui} - \hat{r}_{uj})} U_u + 2\lambda_{V,i} V_i + \gamma\big(-2U_u(r_{ui} - \hat{r}_{ui})\big)$$

$$\frac{\partial \text{BT-OPT(2)}}{\partial V_j} = \frac{1}{1+\exp(\hat{r}_{ui} - \hat{r}_{uj})} U_u + 2\lambda_{V,j} V_j + \gamma\big(-2U_u(r_{uj} - \hat{r}_{uj})\big)$$

Algorithm 1: Parallel SGD for Improved-BT

Input : observed rating matrix R; the set of observed rating pairs Ω; balance factor γ; regularization parameter λ; learning rate η
Output: U and V

```
while not converged do
    for each thread t ∈ {1, 2, ..., T} do
        repeat
            choose (u, i, j) ∈ Ω uniformly at random;
            update U_u ← U_u − η ∂BT-OPT(2)/∂U_u;
            update V_i ← V_i − η ∂BT-OPT(2)/∂V_i;
            update V_j ← V_j − η ∂BT-OPT(2)/∂V_j;
        until sampling S times is done;
    end
    η ← η/2 ;                    // update learning rate
end
```

Let T be the number of threads and S be the sample size for each thread. In the paper, we take the value $S = \frac{|\Omega|}{T}$. The full algorithm of parallel SGD is described in Algorithm 1.

Problem with SGD: In practice, we may need to investigate when γ is set to be large. In such situation, the Improved-BT will solely recover a regression model. However, if we set a large value for γ, then the entry values of the partial derivatives $\frac{\partial \text{BT-OPT(2)}}{\partial U_u}$, $\frac{\partial \text{BT-OPT(2)}}{\partial V_i}$, $\frac{\partial \text{BT-OPT(2)}}{\partial V_j}$ become too large (unless we use an extremely small learning rate). As a result, after a few iterations of parameter update, the algorithm may encounter numerical errors, e.g., entries in UV^T may exceed the limitation of a real number that a machine can handle.

B. Parallel Sampling-based Stochastic Gradient Descent

Algorithm 2: Parallel SSGD for Improved-BT

Input : observed rating matrix R; the set of observed rating pairs Ω; balance factor γ; regularization parameter λ; learning rate η
Output: U and V

while not converged **do**
  for each thread $t \in \{1, 2, ..., T\}$ **do**
    repeat
      choose $(u, i, j) \in \Omega$ uniformly at random;
      sample $z \in [0, 1]$ uniformly at random;
      if $z < \frac{1}{1+\gamma}$ **then**
        update $U_u \leftarrow U_u - \eta \frac{\partial \text{PairLoss}}{\partial U_u}$;
        update $V_i \leftarrow V_i - \eta \frac{\partial \text{PairLoss}}{\partial V_i}$;
        update $V_j \leftarrow V_j - \eta \frac{\partial \text{PairLoss}}{\partial V_j}$;
      else
        update $U_u \leftarrow U_u - \eta \frac{\partial \text{Penalty}}{\partial U_u}$;
        update $V_i \leftarrow V_i - \eta \frac{\partial \text{Penalty}}{\partial V_i}$;
        update $V_j \leftarrow V_j - \eta \frac{\partial \text{Penalty}}{\partial V_j}$;
      end
    until sampling S times is done;
  end
  $\eta \leftarrow \frac{\eta}{2}$; `// update learning rate`
end

Here, we propose a sampling-based SGD method to tackle the problem with SGD, by dividing BT-OPT(2) into two parts: a regularized pairwise loss, represented by:

$$\text{PairLoss} := -\sum_{(u,i,j)\in\Omega} \log \frac{\exp(U_u V_i^T)}{\exp(U_u V_i^T) + \exp(U_u V_j^T)} + \sum_{u=1}^{m} \lambda_{U,u} \|U_u\|_F^2 + \sum_{i=1}^{n} \lambda_{V,i} \|V_i\|_F^2 \quad (4)$$

and the penalty term similar to a regularized regression loss:

$$\text{Penalty} := \sum_{(u,i,j)\in\Omega} (r_{ui} - U_u V_i^T)^2 + (r_{uj} - U_u V_j^T)^2 + \sum_{u=1}^{m} \lambda_{U,u} \|U_u\|_F^2 + \sum_{i=1}^{n} \lambda_{V,i} \|V_i\|_F^2 \quad (5)$$

We can calculate the partial derivatives of U_u, V_i and V_j for these two objective functions separately. In this way, the balance factor γ is excluded in the calculation of each of the partial derivatives and hence the update of model parameters will not be affected by large γ. We first sample a real number z uniformly at random between 0 and 1 and compare z with $\frac{1}{1+\gamma}$. If $z < \frac{1}{1+\gamma}$, then we optimize the "PairLoss"; else we optimize the penalty term. A description of the sampling-based SGD method (SSGD) is shown in Algorithm 2.

This nice trick is inspired by [16]. The difference lies again in the way the data are accessed. We always access data points in pairs even when only the penalty term is used.

V. Experiments

A. Datasets

We test algorithms on three popular datasets: *MovieLens1M*, *MovieLens10M* (see http://grouplens.org/datasets/movielens/), and *Netflix*. Some statistics of the datasets are shown below:

TABLE I
STATISTICS OF TEST DATASETS.

Datasets	# Users	#Items	#Ratings	Density
Movielens1M	6,040	3,706	1,000,209	4.47 %
MovieLens10M	71,567	10,681	10,000,054	5.29%
Netflix	480,189	17,770	100,480,507	1.18%

In the paper, we partition each dataset into training and test parts following a popular setup in [20], [8], [12]. For each user, we randomly select N ratings as training samples and all the remaining observed ratings are used for testing. Since we evaluate algorithms using NDCG@10, there should be at least 10 observed ratings in the test set for each user. Hence, users who have less than N+10 observed ratings will be dropped.

B. Regularization and Learning Rate

In the experiments, after tuning the regularization parameter λ on several datasets, we observe that when λ locates in a specific range, e.g., 1 to 1000, the ranking performance is not sensitive to the change of λ. Hence, we fix $\lambda = 100$. In the paper, learning rate is chosen from $\{0.1, 0.05, 0.01, 0.005, 0.001\}$. For the reproducibility of our proposed method, we recommend a setting: $\lambda = 100$ and $\eta = 0.01$.

C. Ranking Metric

In our experiments, we evaluate our proposed algorithms by Normalized Discounted Cumulative Gain (NDCG) [7]: $NDCG@K(u) = \frac{DCG@K(u,\pi_u)}{DCG@K(u,\pi_u^*)}$ 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 kth ranked item generated by our ranking model. According to the settings of datasets, in the experiments we set K to 10.

D. Impact of Balance Factor γ

Some results measured by NDCG@10 are reported in Figure 2. The first interesting observation is that the best choice of γ is affected by the training sample size N. When N gets larger, the best choice of γ tends to be smaller.

We know that in Eq. (3), the objective function recovers pairwise BT model when $\gamma = 0$ and recovers regression model when γ goes to infinity. Accordingly in Figure 2, the

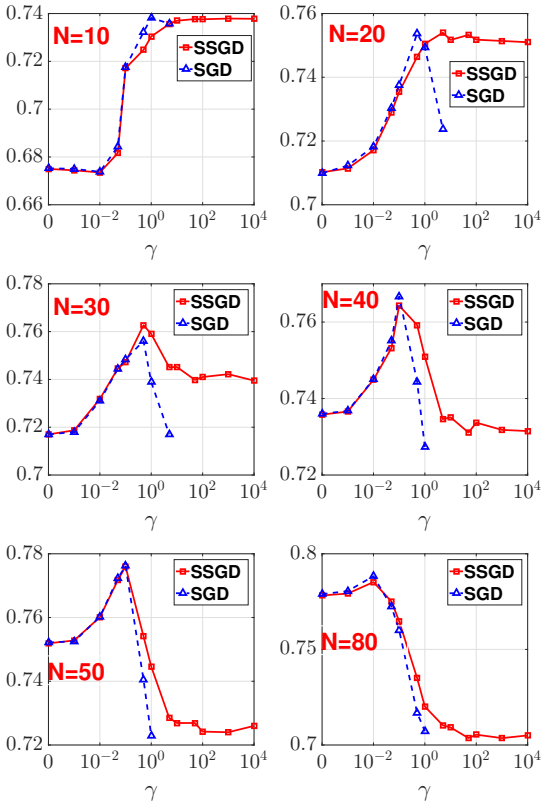

Fig. 2. Experimental results measured by NDCG@10 on Movielens1M as a function of varying balance factor γ. The number of training samples N is chosen from {10, 20, 30, 40, 50, 80}. The pre-defined parameters include: regularization parameter $\lambda = 100$, learning rate $\eta = 0.01$, and $rank = 100$.

leftmost points of all the performance curves represent the results of pairwise BT model ($\gamma = 0$), while the rightmost points of the performance curves roughly show the results of regression model. This figure suggests that the regression model outperforms pairwise BT model when N is small, while pairwise BT performs better when N is large. This observation might be explained by the fact that the number of training samples as pairwise increases quadratically in N, while the number of training samples as single points increases linearly.

Besides of the aforementioned observations, it is clear that our proposed Improved-BT model outperforms both of BT model and regression model, since we can always choose the best γ according to different N. In particular, when N is chosen in a specific range (e.g., $N = 30, 40, 50$ in our example), Improved-BT model shows significant performance improvement over those two methods when best γ is selected.

E. SGD versus SSGD

In Figure 2, we observe that general SGD and SSGD perform almost the same when γ is small (e.g., when $\gamma \leq 0.1$). However, as γ increases, the advantage of SSGD over SGD appears: when $\gamma \geq 10$, there is no experimental report for SGD since SGD encounters numerical difficulty in the optimization process (see Section IV-A). However, SSGD never runs into such trouble when γ becomes large since γ is excluded in the calculation of partial gradients of model parameters.

F. Parallelization and Scalability

We demonstrate the scalability of our proposed methods in Table II and Figure 3. All the experiments were conducted on a single workstation: DELL Precision T7600 with two Intel@Xeon(R) E5-2687W 3.10GHZ 16-core CPUs. Figure 3 plots the ranking performance as a function of running time. We see that both of SGD and SSGD converge to similar results in different number of threads. It is also shown that when provided more threads, the algorithms always converge faster.

TABLE II
SCALABILITY OF PROPOSED METHODS.

Methods	Threads	1	2	4	8	16
SGD	Time (sec)	5.976	3.949	2.602	1.603	0.923
	Speedup	1x	1.5x	2.3x	3.7x	6.5x
SSGD	Time (sec)	4.451	3.289	2.025	1.322	0.742
	Speedup	1x	1.4x	2.2x	3.4x	6.0x

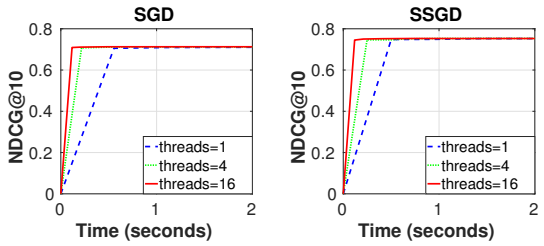

Fig. 3. The performance of parallel SGD and SSGD on Movielens1M dataset. Parameters are selected as: $N = 20$, $\eta = 0.05$, $\lambda = 100$, and $rank = 100$.

G. Compare with Other Methods

Methods for comparison: (a) *Factorization Machine (FM)* [14] is a notable rating prediction method. (b) *CofiRank* [20] also known as maximum margin matrix factorization is always considered as a strong baseline method for collaborative ranking. (c) *Local Collaborative Ranking (LCR)* [18] is a state-of-the-art collaborative ranking method where R is approximated by many locally low-rank matrices. (d) *AltSVM* [12] demonstrates very competitive ranking performance in the original paper, which reduces matrix factorization to alternating SVM problems.

Settings: In the experiment, we set $N = 10, 20, 50$ and we conduct 5 times of independent experiments for each method in each setting. In order to make fair comparisons, we use the source code given by the authors and adopt the settings from their original papers or software. For our proposed Improved-BT, we fix $\lambda = 100$, rank $f = 100$, and learning rate η is chosen from {0.1, 0.05, 0.01, 0.005, 0.001}. Since the best choice of γ is decided by the sparsity level of training data, we recommend a setting: when $N = 10$, $\gamma = 10$; when $N = 20$, $\gamma = 0.1$ or 1; when $N = 50$, $\gamma = 0.01$.

TABLE III
COMPARISONS OF COLLABORATIVE RANKING METHODS. WE REPORT THE NDCG@10 IN THE SETTINGS OF $N = 10, 20, 50$ AND VALUES IN BOLD-FACE INDICATE THE BEST PERFORMANCE. "-" REPRESENTS UNREPORTED VALUES SINCE LCR RUNS FAIRLY SLOWLY ON THE LARGE-SCALE NETFLIX DATA.

Datasets	Settings	CofiRank	FM	LCR	AltSVM	Improved-BT
Movielens1M	N=10	0.7091 ± 0.0023	0.7166 ± 0.0024	0.6978 ± 0.0031	0.6694 ± 0.0014	**0.7386 ± 0.0012**
	N=20	0.7226 ± 0.0013	0.7194 ± 0.0033	0.7012 ± 0.0025	0.7154 ± 0.0032	**0.7511 ± 0.0007**
	N=50	0.7287 ± 0.0042	0.7268 ± 0.0030	0.7152 ± 0.0018	0.7610 ± 0.0025	**0.7756 ± 0.0031**
Movielens10M	N=10	0.6902 ± 0.0012	0.7016 ± 0.0024	0.6921 ± 0.0024	0.6429 ± 0.0032	**0.7106 ± 0.0035**
	N=20	0.7050 ± 0.0032	0.6990 ± 0.0032	0.6877 ± 0.0027	0.7058 ± 0.0015	**0.7264 ± 0.0032**
	N=50	0.6971 ± 0.0015	0.6961 ± 0.0021	0.6854 ± 0.0035	0.7402 ± 0.0034	**0.7502 ± 0.0021**
Netflix	N=10	0.6615 ± 0.0051	0.7148 ± 0.0025	-	0.6461 ± 0.0013	**0.7334 ± 0.0017**
	N=20	0.6927 ± 0.0034	0.7220 ± 0.0035	-	0.7303 ± 0.0024	**0.7589 ± 0.0029**
	N=50	0.7058 ± 0.0054	0.7379 ± 0.0034	-	0.7520 ± 0.0047	**0.7632 ± 0.0019**

Performance comparisons: Table III presents the comprehensive comparisons of all the methods. In the table, we can see that Improved-BT performs the best on all three datasets. Since some datasets (e.g., Netflix) are fairly large, the improvement of most cases in the Table III is fairly considerable. Besides, the results show that our proposed approach always obtains the best ranking performance in different N values.

Running time: We also report the running time for the comparison methods on Movielens1M dataset in Table IV. Our method runs almost 100 times faster than several algorithms, such as LCR and CofiRank, even in the 1-thread setting.

TABLE IV
RUNNING TIME (SEC) OF COMPARED METHODS. WE REPORT THE RESULT OF 1-THREAD PROGRAM ON THE MOVIELENS1M DATASET.

Methods	CofiRank	FM	LCR	AltSVM	Improved-BT
N=10	230.4	107.2	499.3	3.5	2.1
N=20	399.0	214.3	1002.2	6.8	4.3
N=50	898.1	412.4	2432.1	24.6	13.7

VI. CONCLUSION

Collaborative ranking is crucial for recommender systems. A modern approach for collaborative ranking is to combine matrix factorization with Bradley-Terry model. However, there comes an identifiability issue (and non-convex constraints) when matrix factorization is combined with the Bradley-Terry model. Besides, an implicit numerical error (no minimizer of objective function) may occur if we directly apply the Bradley-Terry model for pairwise data without any modification. In this paper, we proposed an Improved-BT model to address the aforementioned issues. We solved Improved-BT through parallel stochastic gradient descent. It was shown in the experiments that our proposed method outperforms many considered state-of-the-art collaborative ranking methods on both of ranking performance and time efficiency.

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, 2012.

[4] R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[5] 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.

[6] D. Cossock and T. Zhang. Subset ranking using regression. In *Learning theory*, pages 605–619. Springer, 2006.

[7] K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 41–48, 2000.

[8] 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, 2014.

[9] P. Li. Abc-boost: Adaptive base class boost for multi-class classification. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 625–632, 2009.

[10] 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.

[11] 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, 2009.

[12] D. Park, J. Neeman, J. Zhang, S. Sanghavi, and I. S. Dhillon. Preference completion: Large-scale collaborative ranking from pairwise comparisons. *arXiv preprint arXiv:1507.04457*, 2015.

[13] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

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

[15] 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.

[16] D. Sculley. Combined regression and ranking. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 979–988, 2010.

[17] 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, 2009.

[18] 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.

[19] A. Töscher, M. Jahrer, and R. M. Bell. The bigchaos solution to the netflix grand prize. *Netflix prize documentation*, 2009.

[20] M. Weimer, A. Karatzoglou, Q. V. Le, and A. J. Smola. Cofi rank - maximum margin matrix factorization for collaborative ranking. In *Advances in Neural Information Processing Systems 20*, pages 1593–1600. 2007.