COLLABORATIVE RANKING-BASED RECOMMENDER SYSTEMS

by

JUN HU

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Ping Li

And approved by

New Brunswick, New Jersey

October, 2018

ABSTRACT OF THE DISSERTATION

Collaborative Ranking-based Recommender Systems

By JUN HU

Dissertation Director:

Ping Li

Recommender systems are by far one of the most successful applications of big data and machine learning. The goal of recommender systems is to find what is likely to be of interest, thus enabling personalization and tailored services. Among all the recommendation algorithms, collaborative filtering is the most common technique. In this thesis, we present our research results in the field of ranking-based collaborative filtering. We view the task of recommendation as providing a personalized ranked list of items for each user and thus formulate it as a ranking problem. In order to generate accurate recommended lists, we look into the technique of combining learning-to-rank with conventional collaborative filtering methods for solving the recommendation task and comprehensively discuss the challenges and advantages of this approach. Particularly, we propose an improved pairwise ranking model and a multi-objective ranking framework to address the issues which occur during the combination of learning-to-rank and collaborative filtering. In addition, we propose a new ordinal approach to modeling the ordinal nature of user preference scores, which demonstrates distinct superiority compared to the numerical and (nominal) categorical views of user ratings.

Acknowledgements

Firstly, I would like to express my highest gratitude to my advisor Prof. Ping Li for the continuous support of research, for his patience, motivation, immense knowledge. He always gave me plenty of freedom and the best resources to investigate topics that I am interested in, and also provided invaluable support and advice to help me make research progresses. Without him, it is impossible to accomplish this thesis.

Besides my research advisor, I would like to thank the rest of my thesis defense committee: Prof. Zheng Zhang, Prof. Yongfeng Zhang, and Prof. Han Xiao for their insightful comments and encouragement. In addition, I would also like to thank Prof. Vladimir Pavlovic, Prof. Desheng Zhang and Prof. Shan Muthukrishnan to serve as my qualified exam committee members and Prof. Tomasz Imieliski, who gave me precious advice during the first two years of my Ph.D. study.

I thank my current and previous labmates: Jie Shen, Jie Gui, Yuan Cao, Jing Wang, Martin Slawski, Anshumali Shrivastava, Ji Zhang, for the stimulating discussions which inspire me and widen my research from various perspectives, and for the fun we had during the past years. In addition, I would thank my talented friends and collaborators: Chaolun Xia, Meng Li, Yan Wang, Yan Zhu, Han Zhang, Ruilin Liu etc, for sharing creative ideas and pushing deadlines together. Without their help, it would become extremely hard to constitute the success of this research. My sincere thanks also go to my mentors at Yahoo! Research: Dr. Meizhu Liu, Dr. Changwei Hu, and Dr. Yifan Hu, who provided me with an opportunity to join their team as an intern.

Last but not the least, I would like to thank my parents for supporting me spiritually throughout my life. I also thank my girlfriend for persistently supporting my research.

Dedication

To my parents

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Recommender systems are by far one of the most successful applications of big data and machine learning. They are crucial to the success of many Internet companies. For example, Amazon[1] brings more than 30% of revenues from recommender systems, and 75% of what people watch in Netflix[2] is from some sort of recommendation. Recommender systems emerged as a relatively independent research area in the mid-1990s with the rapid development of internet services [1, 2, 3, 4]. The increasing importance of recommender systems as the mainstream technique for e-commerce has attracted a large community of researchers and developers from different fields to work together promoting the progress of recommendation technology.

1.1 Overview of Recommender Systems

Recommender systems are information filtering systems providing suggestions for items to be of use to target users [5, 6, 3, 7]. The suggestions relate to various decision-making processes, such as what items to buy, what music to listen to, or what online news to read. Common recommender systems can be found in many web applications. For example, Netflix, a video-on-demand company, matches tens of millions customers with a huge inventory of movies based on the analysis of customers' past tastes; Amazon is the pioneer in using recommender systems. The recommendation in Amazon sits on a huge volume of collective information of its customers and suggests interesting items to the potential users; Facebook[3] and Linkedin[4], the social and professional networking sites, build very large social networks/graphs based on the

[1]https://www.amazon.com/

[2]https://www.netflix.com/

[3]https://www.facebook.com/

[4]https://www.linkedin.com/

relationships between users. Interesting services (e.g., friends, jobs) are often recommended when users log in their websites. Other popular applications of recommender systems are also seen in business [8, 9, 10], web search [11], education [12, 13, 14], e-government [15, 16, 17], tourism [18, 19, 20] etc. In addition, there are many types of recommender systems developed in order to make efficient and accurate recommendations in different scenarios, e.g., collaborative filtering [21], content-based (CB) [22, 23] and knowledge-based (KB) [24] recommender systems, network-based recommender systems [25, 26], fuzzy recommender systems [27], context awareness-based recommender systems [28, 29] and group recommender systems [30].

There are various reasons why service providers may want to exploit the recommendation technology. Firstly, it is easy to increase the sales of products by applying a reasonable recommender system. For example, after watching a movie at Netflix, a user is probably interested in watching another interesting one and recommender systems can provide multiple choices for satisfying users' interests. Secondly, recommender systems are able to sell more diverse items. Without applying typical recommendation approaches, only popular items may be demonstrated to users while niche items will never be viewed by users. However, niche products are able to grow to become a large share of total sales [31, 32, 33]. Thus, it is vital to develop recommendation approaches to promote the sales of niche items. Moreover, recommendation systems can increase user satisfaction. If service providers accurately recommend interesting, relevant, high-quality products, then it will increase users' subjective evaluation of the systems and in turn increase the usage of provided services. In addition, when old customers are recognized and taken seriously through personalized services, their fidelity to the products can be hugely increased. In order to achieve the aforementioned functions, recommender systems are designed to increase the relevance, novelty [34], serendipity [35] and diversity.

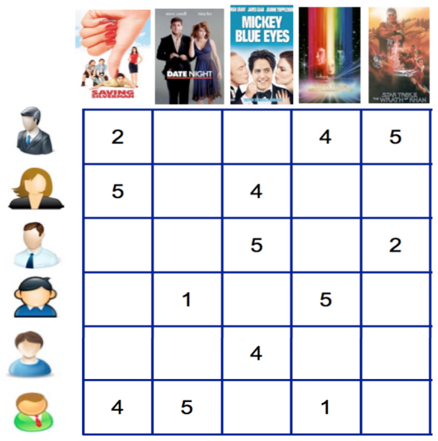
Among all the requirements of recommender systems, the core problem is to find user personalization and thus enable customized recommendations. In order to analyze user behaviors, many e-commerce websites provide different ways to collect users' likes or dislikes through, e.g., asking for a rating score for a purchase, providing text boxes to collect user reviews, etc. Another effective approach to understanding user behavior is to track users' activities in the websites following the users' click events or the act of a user buying or browsing an item, which can be considered as an endorsement for that item. All the above users' feedback provides rich knowledge for understanding users' tastes. After collecting rich content from users, items and user-item interactions, recommender systems are able to infer users' internal interest and then make customized services to satisfy users' needs.

The basic principle of recommendation is that there are significant co-relationships between similar users or similar items and users generally show consistent behavior from the past to the future. For example, users with similar personalities may demonstrate the similar interest in the same set of items; items with similar features (e.g., same brand, similar function, and appearance, etc) may receive same rating scores. This intuition directly motivates the set of neighborhood-based (so-called memory-based [36]) recommendation approaches, which have been widely applied in commercial recommender systems. In addition, users generally show consistent preferences over items. For example, users who watched many science fiction movies in the past tend to like science fiction films more than other types of movies (e.g., literary movies or documentary films) in future. This consistency of users' internal preferences is an important assumption in many recommendation algorithms, even though users' interests may change a little over time.

The aforementioned users' internal preferences can be learned in a data-driven manner from a rating matrix, which is obtained from users' rating history. The resulting recommendation model is used to make predictions for target users. Generally speaking, the larger the number of rated items that are available for a user, the easier it is to make robust predictions about the future behavior of the user. In this thesis, we mainly discuss recommendation approaches when only the rating matrix is given. This problem is always considered as the basic problem in the research of recommender systems and collaborative filtering is one of the prominent algorithms to solve this problem.

1.2 Collaborative Filtering

There are two typical kinds of data used as inputs for recommender systems. One is the user-item interaction information, such as rating matrix or click events; the other one refers to attributes associated with users and items, such as user profiles, item categories or keywords



2			4	5
5		4		
		5		2
	1		5	
		4		
4	5		1	

Figure 1.1: The input data of collaborative filtering. Each row represents a user and each column represents an item. One entry with a value represents the preference score one user gives to an item. The scores in the empty entries are those we need to estimate.

describing items' properties. The set of algorithms utilizing user-item interaction data are referred to as collaborative filtering algorithms. In Figure 1.1, it demonstrates a rating matrix as the data processed by collaborative filtering algorithms. Another set of algorithms dealing with attribute-based data are called content-based recommendation methods.

There are two types of collaborative filtering algorithms:

- *Memory-based methods*: it is also referred as neighborhood-based collaborative filtering algorithms. In a user-centric manner described in [37], memory-based algorithms try to find users that are similar to an active user (i.e., the target user we want to make predictions for) and utilize the information collected from similar users to predict the preferences for that active user. In an item-centric manner, items which have similar properties to those a user likes in the past will be recommended to the user.

  Practically, the memory-based methods are pretty easy to implement and the results have high interpretability, and thus the memory-based recommender systems have been widely used in the industry. On the other hand, memory-based algorithms generally perform unsatisfactorily when the rating matrix is too sparse as the user or item similarity calculated on a small set of observed rating scores is not always reliable.

- *Model-based methods*: In model-based methods, machine learning models are used to solve the prediction problem. The common approach is to learn user latent factors and item latent factors, and the predictions on unobserved entries in the rating matrix are made through the combination of user and item latent factors. In this thesis, we mainly discuss the matrix factorization approach [38, 39, 40, 41, 42, 43, 44], which is considered as the most widely used collaborative filtering method. Matrix factorization approach learns a user latent vector for each user and an item latent vector for each item, and the final prediction is made through the dot product of one pair of user and item latent vectors. There is another set of popular collaborative filtering methods which learn user and item embedding through deep learning models [45, 46, 47, 48, 49, 48, 50, 51, 52].

Traditional collaborative filtering algorithms show very high accuracy in rating prediction task. They are popularized through the Netflix Prize Challenge [53], which granted 1 million dollars for the winner.

While collaborative filtering has been investigated and improved extensively over the past years, there is still room for substantial improvement. In this thesis, we focus on improvement by modeling the ordinal nature of user ratings. In current main-stream collaborative filtering algorithms, discrete rating scores are often viewed as either numerical values or (nominal) categorical labels. Practically, viewing user rating scores as numerical values or categorical labels can not precisely reflect the exact degree of user preferences. Different users may have distinct degrees of sensitivity over a sequence of ordered rating scores, resulting in that the distances between any pair of adjacent ratings (e.g., 5-star and 4-star) are different for individual users. Nevertheless, when user ratings are interpreted as numerical values, it equals to the assumption that the user preferences modeled by several consecutive integers (e.g., “1”, “2” to “5”) are equidistant. While from the viewpoint of (nominal) categorical labels, the distances between pairs of ordered scores are ignored. In Chapter 3, we propose an additive ordinal regression model which is able to model the ordinal nature of user rating scores.

1.3 Collaborative Ranking

The accuracy of recommender systems is generally evaluated using two kinds of metrics. One set of metrics are used to evaluate the accuracy of rating prediction, e.g., root mean squared error (RMSE), and the other set of metrics are used to evaluate the ranking performance of recommendation, such as normalized discounted cumulative gain (NDCG).

Most of the collaborative filtering models treat the recommendation problem as a prediction problem in which the squared error between estimated rating scores and observed rating scores is minimized. However, the ultimate goal of recommender systems is to generate a personalized ranked item list for each user. It is intuitive to think about that the relative order of items inferred from algorithms is more important than accurately predicting the rating scores. For example, suppose that the true rating score from a user to an item i is 3 and to an item j is 4 and thus recommender systems should rank the item j in front of the item i in terms of user preference. We develop one algorithm I which predicts the rating score for the item i as 3.6 and rating score for the item j as 3.4. We also develop another algorithm II which predicts the rating score for the item i as 3.4 and the rating score for the item j as 3.6. Then, we should prefer algorithm II since the predicted score for the item j is higher than the score of the item i and thus item j will be ranked in front of item i. However, algorithm I and algorithm II generate the same prediction error in terms of the common metric for evaluating rating prediction accuracy (i.e., RMSE). [54]. Therefore, it is more reasonable to model the recommendation task as a ranking problem.

In the real applications of recommender systems, we should emphasize more on the top of a ranked list, since very few items can be presented to users at a time due to the limited size of user interfaces. For example, when a user buys products through a smartphone, then only top 3 or 5 recommended items can be viewed by the user because of the small size of the phone screen. Furthermore, users generally pay no attention to the bottom of a ranked list and thus the (rating prediction or ranking) accuracy at the bottom of the ranked list should be less emphasized or even ignored. If the lower-ranked items are predicted accurately, while higher-ranked items have severe prediction errors, then the recommender system actually fails to deliver satisfactory results.

In order to achieve better ranking performance, it is proposed to incorporate learning-to-rank techniques into collaborative filtering methods, forming a new set of *collaborative ranking* algorithms. Since the ranking performance is evaluated using different ranking-centric evaluations and thus the general collaborative ranking approaches are designed to directly optimize different kinds of ranking metrics in order to provide more accurate ranked lists.

In accordance with the type of optimized ranking objectives, collaborative ranking methods are generally divided into:

- *Pointwise collaborative ranking methods*. Most of rating prediction approaches are considered as pointwise methods. In general, the pointwise method can also be approximated by a regression problem or a classification problem, where rating scores are accessed in the form of single points throughout the learning process.

- *Pairwise collaborative ranking methods*. In pairwise ranking method, the correct order of any observed rating pairs should be kept during optimization. We only care about the relationship that one item is preferred over the other one, while the fact that whether the rating scores are accurately predicted does not matter.

- *Listwise collaborative ranking methods*. The ranked list of items can be viewed as permutations of items, and the best permutation of items should be the same with that generated from the observed rating scores of items. Practically, it is unfeasible to evaluate all the permutations generated from a large set of items, and thus the common approach of listwise methods is to optimize listwise ranking metrics, such as NDCG and reciprocal rank [55]. However, general listwise ranking errors are not convex and thus it is proposed to optimize the smooth version (or surrogate functions) of those errors.

Among the aforementioned three kinds of collaborative ranking approaches, pointwise methods are more computationally efficient than pairwise and listwise methods because the size of input data for pointwise methods is $O(n)$, where n is the number of observed rating scores, while the size of input data for pairwise methods is $O(n^2)$ and worst case of listwise methods is $O(n!)$ (i.e., total number of permutations). Therefore, pointwise methods are more suitable for large-scale industrial applications. On the other hand, pairwise and listwise methods usually outperform pointwise methods in terms of ranking accuracy, because pairwise and listwise methods directly optimize ranking metrics, which is more reasonable for the ranking purpose.

In this thesis, we mainly establish research discussing the following two problems:

- *Does there exist an algorithm which generates accurate ranked lists like pairwise and listwise methods, but also operates as efficient as pointwise methods?* To answer this question, we propose a ranking method which transfers cumulative probability distributions of ordered rating scores to ranking scores. Our proposed method is a pointwise method which models the ordinal nature of user ratings. In addition, we emphasize more on higher rating scores in order to improve the ranking performance at the top of the ranked list. This part of content is introduced in Chapter 2.

- *Are there any technical issues in current collaborative ranking methods and how to solve them?* In essence, most of the current pairwise collaborative ranking methods are derived from the combination of the Bradley-Terry Model and matrix factorization. However, if we simply combine these two techniques without any further improvements, the pairwise methods will encounter the identifiability issue and the objective function of the Bradley-Terry model will have no minimizer. Therefore, we propose an improved Bradley-Terry Model in Chapter 4 to solve the aforementioned two issues. Meanwhile, we further observe that the user latent factors obtained from pairwise collaborative ranking methods are noneffective for the personalized item recommendation task. We will look into the problem why those user latent factors don't contribute to the item recommendation task. To that end, we propose a multi-objective method to learn both of effective user and item latent factors. This part of content is introduced in Chapter 5.

Chapter 2

Decoupled Collaborative Ranking

In this chapter, 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.

2.1 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 sparse rating matrix $\mathbf{R}$, 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 $\mathbf{R}$.

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 [42, 43, 56]. In pointwise approaches, user ratings are usually defined as numerical value or categorical labels. In many studies concerning item recommendation [42, 57, 58, 59], 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 access data in the form of single points, while the ordinality of user ratings is usually revealed as pairs or ordered lists.

In this chapter, 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 [60]. More specifically, given a user-item preference matrix $\mathbf{R}$ where the ratings range from 1-star to C-star, we firstly decompose $\mathbf{R}$ into C binary matrices. An example of matrix decomposition is shown in Fig. 2.1 when $C = 5$. 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 [54]. Besides, DCR, as a pointwise approach, shows better performance in terms of time efficiency compared to pairwise or listwise approaches.

2.2 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 [36] aggregate similar users or items and predict the

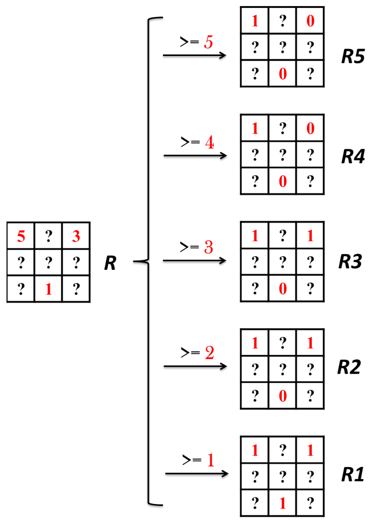


Figure 2.1: An example of matrix decomposition when $C = 5$. "?"s represent the unobserved entries. Each binary matrix in the right column is obtained by classifying the original rating matrix (left) based on the idea of ordinal classification. For example, the top right matrix, i.e., $\mathbf{R5}$, is obtained by grouping observed entries: $r \geq 5$ as "1" and $r \leq 4$ as "0".

unobserved ratings based on the collected set of users or items [61, 62]. 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 $\mathbf{R}$ has low rank and hence matrix factorization techniques can be applied [38, 39, 40, 41, 42, 43, 44]. 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 [63]. 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 [57, 64, 56]. They propose strategies to combine pairwise or listwise LTR methods with matrix factorization, where user ratings are defined as ordinal. For example, a listwise approach named Cofirank [43] optimizes a surrogate convex upper bound of NDCG error and matrix factorization is used as the basic rating predictor. Besides, Rendle et al. [65] and Liu et al. [66] model pairwise comparisons of observed ratings using Bradley-Terry model (a typical pairwise model) with low-rank structure. In [42], it is assumed that rating matrix $\mathbf{R}$ 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.

2.3 Decoupled Collaborative Ranking

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 $\mathbf{R} \in \mathbb{R}^{m \times n}$ whose entry (rating) values are selected from a set of discrete ordered values $\{1, 2, ..., C\}$, with r_{ui} representing the observed rating of user u gives to item i.

2.3.1 Global Matrix Factorization

The basic idea of matrix factorization (MF) is that we assume $\mathbf{R}$ is low-rank and thus it can be approximated by $\hat{\mathbf{R}} = \mathbf{U}\,\mathbf{V}^{\mathsf{T}}$, where $\mathbf{U} \in \mathbb{R}^{m \times f}$ and $\mathbf{V} \in \mathbb{R}^{n \times f}$, f is the rank of approximation and $f \ll \min(m, n)$. The latent factors in $\mathbf{U}$ and $\mathbf{V}$ can be obtained by optimizing a non-convex objective function. When it comes to ranking/recommendation, the items corresponding to unobserved entries of $\mathbf{R}$ are sorted in descending order of the numerical values obtained from matrix factorization. In this approach, the observed ratings in $\mathbf{R}$ are considered as real values.

2.3.2 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 [60]. Given an C-level (i.e., $r \in \{1, 2, 3, ..., C\}$) rating matrix $\mathbf{R}$, we decompose it into C binary matrices. The observed entry values in the t-th decoupled binary matrix ($t \in \{1, 2, ..., C\}$) are obtained by partitioning the ratings in $\mathbf{R}$ 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 C binary matrices $\mathbf{R1}$, $\mathbf{R2}$,...,$\mathbf{R(C)}$. A case example is shown in Fig. 2.1 when $C = 5$. This idea originated from classical statistics, e.g., the so-called cumulative logit model [67].

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:

$$\begin{aligned} P(r^t_{ui} = 1) = P(r_{ui} \geq t) = \frac{\mathbf{U}^t_u \mathbf{V}^{t\intercal}_i + 1}{2} \\ s.t. \quad \| \mathbf{U}^t_u \| \leq 1, \quad \| \mathbf{V}^t_i \| \leq 1 \end{aligned} \tag{2.1}$$

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. $\mathbf{U}^t_u \in \mathbb{R}^f$ and $\mathbf{V}^t_i \in \mathbb{R}^f$ are the latent factors of u-th user and i-th item in the t-th decoupled binary matrix. By imposing the constraint: $\| \mathbf{U}^t_u \| \leq 1$ and $\| \mathbf{V}^t_i \| \leq 1$, the predicted probability will locate in the range from 0 to 1.

Let Ω^t ($t \in \{1, 2, ..., C\}$) 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 $\mathbf{U}^t_u$ and $\mathbf{V}^t_i$ for all the users and items can be learned by maximizing the log-likelihood on the training data:

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

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:

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

if r^t=1, $\frac{\partial P(r^t_{ui}=r^t|\mathbf{U}^t_u,\mathbf{V}^t_i)}{\partial \mathbf{U}^t_u} = \frac{\mathbf{V}^t_i}{2}$ and $\frac{\partial P(r^t_{ui}=r^t|\mathbf{U}^t_u,\mathbf{V}^t_i)}{\partial \mathbf{V}^t_i} = \frac{\mathbf{U}^t_u}{2}$; if r^t=0, $\frac{\partial P(r^t_{ui}=r^t|\mathbf{U}^t_u,\mathbf{V}^t_i)}{\partial \mathbf{U}^t_u}$=$-\frac{\mathbf{V}^t_i}{2}$ and $\frac{\partial P(r^t_{ui}=r^t|\mathbf{U}^t_u,\mathbf{V}^t_i)}{\partial \mathbf{V}^t_i}$=$-\frac{\mathbf{U}^t_u}{2}$. The constraints in Eq. (2.1) can be satisfied through gradient projection: $\mathbf{U}^t_u \leftarrow \frac{\mathbf{U}^t_u}{\| \mathbf{U}^t_u \|}, \mathbf{V}^t_i \leftarrow \frac{\mathbf{V}^t_i}{\| \mathbf{V}^t_i \|}$.

After learning the latent factors $\mathbf{U}^t_u$ and $\mathbf{V}^t_i$, we can compute the probability of $P(r^t_{ui} = 1)$ (i.e., $P(r_{ui} \geq t)$) for all items through Eq. (2.1). 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.

2.3.3 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 [60]:

$$SC_{ui} = \sum_{t=1}^{C} f(t)P(r_{ui} = t) \tag{2.2}$$

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

$$P(r_{ui} = t) = P(r_{ui} \geq t) - P(r_{ui} \geq t+1) \tag{2.3}$$

Combining Eq. (2.2) and Eq. (2.3), we reformulate the scoring function for ranking as:

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

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

$$SC_{ui} = f(1)P(r_{ui} \geq 1) + \sum_{t=2}^{C}(f(t) - f(t-1))P(r_{ui} \geq t) \tag{2.4}$$

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 $\{\mathbf{U}_u^t, \mathbf{V}_i^t\}$, $\forall u, i, t$ 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. (2.1) and Eq. (2.4);
3. for a user u, sort all the items in descending order of SC_{ui} and recommend items at the top of sorted list.

Question: how to set $f(t)$ for ranking?

In this chapter, 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 chapter, 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 Fig. 2.3). 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.

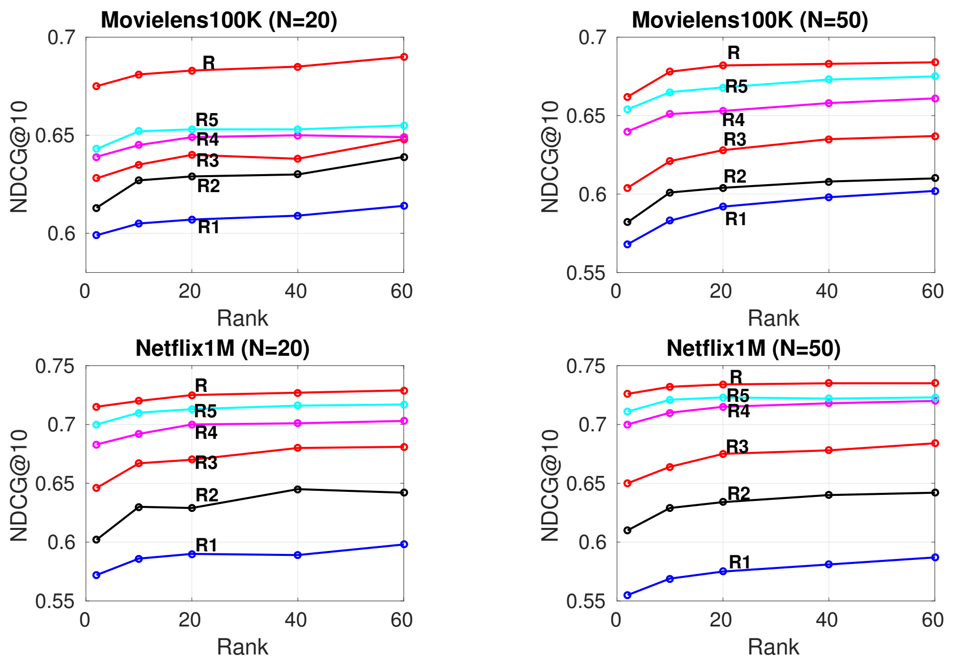


Figure 2.2: Comparison of ranking performance on individual rating matrices, including original rating matrix $\mathbf{R}$ and each of the decoupled binary matrices. "Rank" refers to the dimensionality for low-rank approximation.

2.3.4 Empirical Study on Binary Classifications

We implement ranking algorithms on each of the binary matrices $\mathbf{R1}$, $\mathbf{R2}$,..., $\mathbf{R5}$ ($r \in \{1, 2, ..., 5\}$) 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 2.3.1) on the original rating matrix $\mathbf{R}$ as the comparison. The performance is reported in Fig. 2.2. 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 $\mathbf{R5}$ 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 $\mathbf{R5}$, while the items in $\mathbf{R4}$ are sorted in descending order of $P(r \geq 4)$ and hence rating score "4" is considered as positive label (i.e., "1"). In Fig. 2.2, it demonstrates that the

ranking performance of $\mathbf{R5}$ is better than $\mathbf{R4}$, 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 Fig. 2.2 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. (2.4) satisfies both of these two requirements.

Careful readers may ask why we do not remove $\mathbf{R}\,1$ since all the labels in $\mathbf{R1}$ is "1" and hence this matrix is non-informative. Actually, $\mathbf{R1}$ reflects certain kind of hidden profile of user actions. For example, $r_{ui}^{1} = 1$ in $\mathbf{R}\,1$ can represent that uth user has watched the ith movie. It is reasonable to consider that a user demonstrates more interests in a movie which he has watched than the one he just skipped. The SVD++ approach [39] also includes this kind of hidden information into their rating prediction model. Therefore, we still keep $\mathbf{R1}$ in our approach.

2.4 Extension: Further Improve 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 [68]. 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 chapter, we minimize the following loss:

$$\begin{aligned} \mathcal{E}(g) = \sum_{(u,i,j)\in\mathcal{O}} \mathcal{L}(Y_{uij}\cdot g(u,i,j)) + \lambda \sum_{t} (\parallel \mathbf{U}^t \parallel_F^2 + \parallel \mathbf{V}^t \parallel_F^2) \\ s.t. \quad \parallel \mathbf{U}_u^t \parallel^2 \le 1, \quad \parallel \mathbf{V}_i^t \parallel^2 \le 1 \quad \forall u,i,t \end{aligned} \tag{2.5}$$

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 [42, 58]).

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:

$$\frac{\partial\mathcal{E}}{\partial\mathbf{U}_u^t} = \sum_{(u,i,j)\in\mathcal{O}} \frac{\partial\mathcal{L}}{\partial g(u,i,j)} \frac{\partial g(u,i,j)}{\partial\,\mathbf{U}_u^t} + 2\lambda\,\mathbf{U}_u^t \tag{2.6}$$

$$\begin{aligned} \frac{\partial\mathcal{E}}{\partial\mathbf{V}_i^t} = \sum_u \Bigg(\sum_{j:r_{ui}>r_{uj}} \frac{\partial\mathcal{L}}{\partial g(u,i,j)} \frac{\partial g(u,i,j)}{\partial\,\mathbf{V}_i^t} \\ + \sum_{j:r_{ui}<r_{uj}} \frac{\partial\mathcal{L}}{\partial g(u,i,j)} \frac{\partial g(u,i,j)}{\partial\,\mathbf{V}_i^t} \Bigg) + 2\lambda\,\mathbf{V}_i^t \end{aligned} \tag{2.7}$$

It should be mentioned that there are two cases when calculating $\mathbf{V}_i^t$: the derivative of $\mathbf{V}_i^t$ when i is preferred to j is different from that when j is preferred to i, which is shown in Eq.

(2.7). The partial derivatives are:

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial g} &= -\frac{Y_{uij}}{1+\exp(Y_{uij}g(u,i,j))} \\
\frac{\partial g}{\partial \mathbf{U}_u^t} &= \frac{\mathbf{V}_i^t - \mathbf{V}_j^t}{2} \\
\frac{\partial g}{\partial \mathbf{V}_\alpha^t} &= \begin{cases} \frac{1}{2}\,\mathbf{U}_u^t & \text{if } \alpha = i \\ -\frac{1}{2}\,\mathbf{U}_u^t & \text{if } \alpha = j \end{cases}
\end{aligned}
$$

The full algorithm is shown in Algorithm 1.

Algorithm 1: Pairwise DCR

```
Input  : $\{\mathbf{U}_u^t, \mathbf{V}_i^t\}$ learned from DCR, observed pairs of ratings $\mathcal{O}$; learning rate $\eta$;
         regularization parameter $\lambda$
Output: $\{\mathbf{U}_u^t, \mathbf{V}_i^t\}, \forall u, i, t$
1  while not converged do
2      for all users u do
3          find $(u,i,j) \in \mathcal{O}$, all rating pairs by user $\mathbf{U}$;
4          calculate $\Delta\,\mathbf{U}_u^t$ using Eq.(2.6);
5          calculate $\Delta\,\mathbf{V}_i^t$ using Eq.(2.7);
6          $\mathbf{U}_u^t \leftarrow U_u^t - \eta\Delta\,\mathbf{U}_u^t$;
7          $\mathbf{V}_i^t \leftarrow V_i^t - \eta\Delta\,\mathbf{V}_i^t$;
8          $\mathbf{U}_u^t \leftarrow \frac{\mathbf{U}_u^t}{\|\mathbf{U}_u^t\|}$;   $\mathbf{V}_i^t \leftarrow \frac{\mathbf{V}_i^t}{\|\mathbf{V}_i^t\|}$
9      end
10 end
```

2.5 Experiment

2.5.1 Experimental Settings

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 chapter refers to the first part. Netflix1M dataset is randomly sampled from Netflix training set. More statistics of these datasets are collected in Table 2.1.

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

Table 2.1: 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 [43, 42, 59] 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 [43, 42, 59] by choosing N: 10, 20, 50.

Performance metric In our experiments, we evaluate our proposed algorithms by Normalized Discounted Cumulative Gain (**NDCG**) [54], 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.

2.5.2 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 2.3.3 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

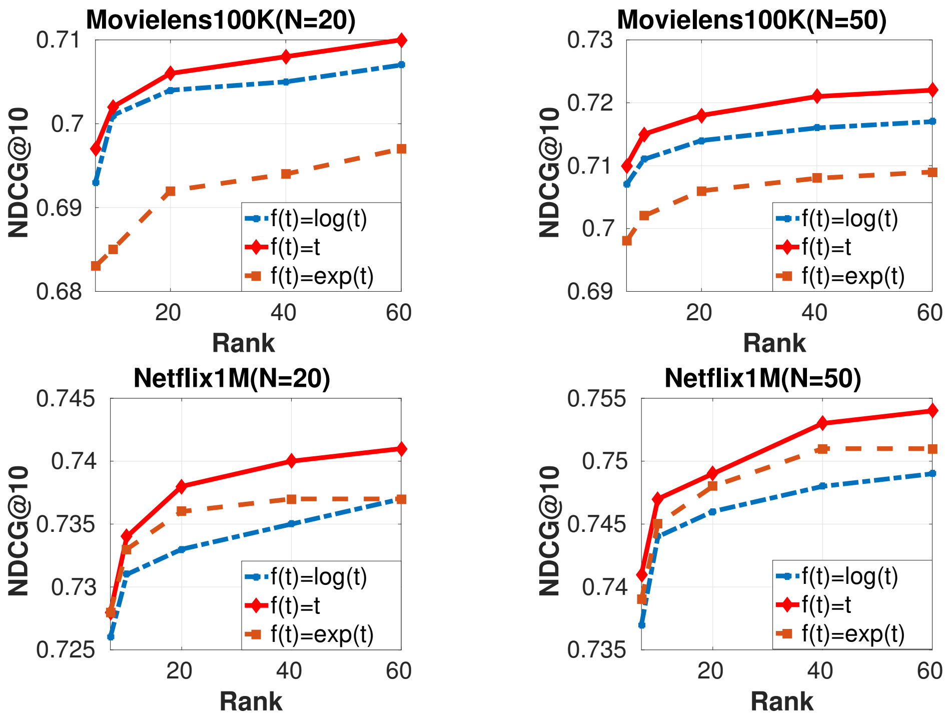


Figure 2.3: Comparisons on the ranking performance of different relevance functions as a function of $rank$.

Fig. 2.3.

Comparisons of relevance functions

From the results in Fig. 2.3, 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 while the weights for all the other scores are set as 0. Therefore, we only need to calculate the probability of $P(r = C)$, which can be obtained through binary classification on a single binary matrix $\mathbf{R}(\mathbf{C})$. However, from the results in Fig. 2.2, 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. (2.4) reduces to: $SC_{ui} = f(1)P(r_{ui} \geq 1)$, which merely considers $\mathbf{R}\,1$. From the observation in Fig. 2.2, this setting should be avoided.

Effect of Parameter $rank$

From the observations in Fig. 2.2 and Fig. 2.3, 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.

2.5.3 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 [39] 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 [69].

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 C times more parameters than a single matrix factorization where $C = 5$ in these test datasets. The results are reported in Fig. 2.4.

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.

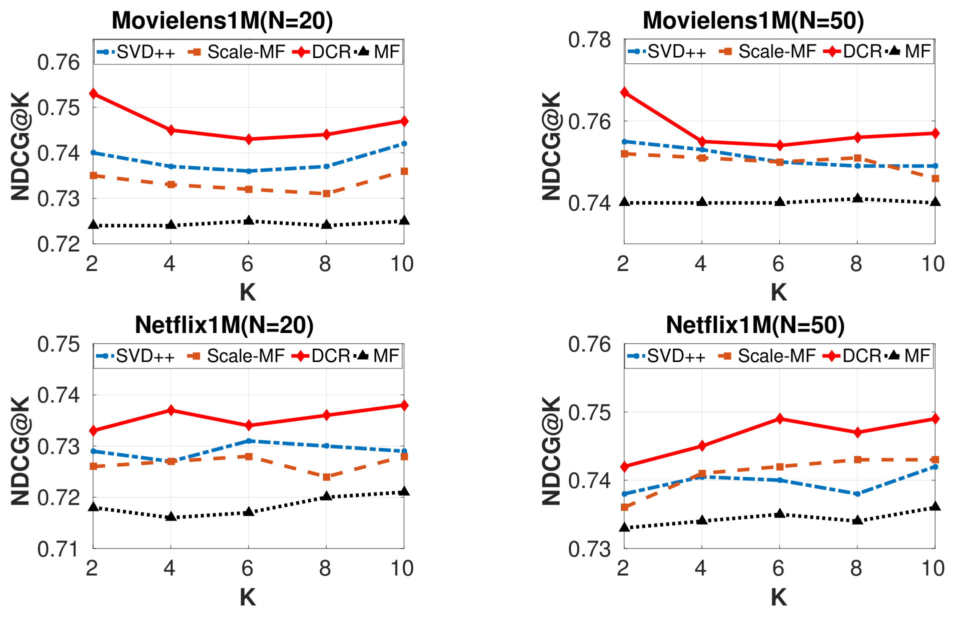


Figure 2.4: Comparisons with pointwise methods in terms of NDCG@K scores. K varies from 2 to 10. $rank = 20$ in DCR and $rank = 100$ in other methods.

Table 2.2: Compare with push-based top-K ranking algorithms.

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**

2.5.4 Compare with Push Collaborative Ranking

In this section, we compare DCR with a state-of-art collaborative ranking algorithm-Push Collaborative Ranking [70], 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.

2.5.5 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:
$$-\sum_{(u,i,j)\in\Omega} \log \frac{\exp(\mathbf{U}_u \mathbf{V}_i^{\intercal})}{\exp(\mathbf{U}_u \mathbf{V}_i^{\intercal}) + \exp(\mathbf{U}_u \mathbf{V}_j^{\intercal})} + \lambda_{\Theta}\|\Theta\|^2$$
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., $\mathbf{U}$ and $\mathbf{V}$). Actually, the BT method is almost the same with **Bayesian Personalized Ranking (BPR)** model [65]. (ii) **Local Collaborative Ranking (LCR)** [36] is another collaborative ranking method which also optimizes a pairwise ranking loss. In LCR, $\mathbf{R}$ is approximated by many locally low-rank matrices.

- Listwise approach: **CofiRank** [43] 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 [36, 58, 43]. 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 $rank = C \cdot k$ for other methods, where C 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

Table 2.3: Comparisons with pairwise and listwise approaches in terms of NDCG@10.

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**

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.

Performance Comparisons

An extensive performance comparison of all the methods is shown in Table 2.3. We report the average and standard deviation over 5 times of independent experiments. 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.

Comparisons of Efficiency

We also report the running time (seconds) for the comparison methods on Netflix1M dataset in Table 2.4. The running time of Cofirank, LCR is obtained based on the software implemented by their authors. From the observation in Table 2.4, we see that our proposed method can run more than one hundred times faster than several algorithms, such as 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.

Table 2.4: 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

2.5.6 Effectiveness of Pairwise DCR

We report the performance of Pairwise DCR and compare it with Improved Bradley-Terry model (Improved-BT) [59]. 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 conclude that refining the latent features of DCR by pairwise learning is able to improve the ranking performance. On the other

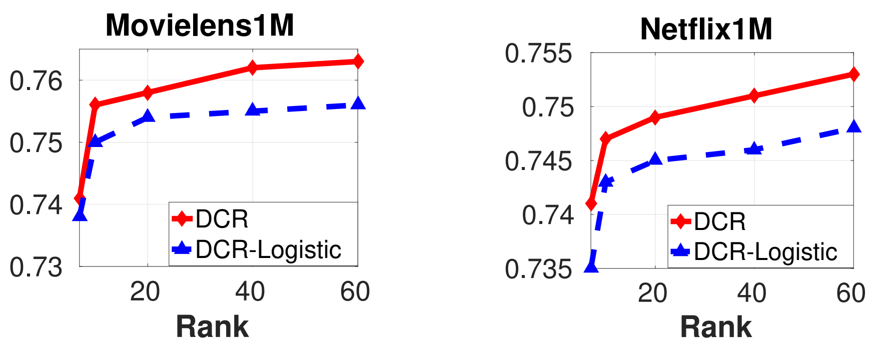


Figure 2.5: Compare DCR with DCR-Logistic.

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 Fig. 2.2, 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.

2.6 Discussions on DCR-Logistic

Readers may ask why we use a predictive model as that in Eq. (2.1), rather than a seemingly more popular logistic mapping function as follows:

$$P(r_{ui} \geq t) = \frac{1}{1 + \exp(-\mathbf{U}_u^t \mathbf{V}_i^{t\mathsf{T}})} \tag{2.8}$$

Indeed, in this chapter we also implemented it using Eq. (2.8) (we call it **DCR-Logistic**). We report the ranking performance in terms of NDCG@10 in Fig. **??** 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. (2.1), the parameters are bounded in a unit ball (i.e., $\| \mathbf{U}_u^t \| \leq 1, \quad \| \mathbf{V}_i^t \| \leq 1$), 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. (2.1) by Eq. (2.8) in Pairwise DCR and add more constraints to the parameters, then the optimization of Pairwise DCR model will be quite computationally expensive.

2.7 Conclusion

In this chapter, 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 $\mathbf{R}$ 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.

Chapter 3

Collaborative Filtering via Additive Ordinal Regression

Accurately predicting user preferences/ratings over items are crucial for many Internet applications, e.g., recommender systems, online advertising. In current main-stream algorithms regarding the rating prediction problem, discrete rating scores are often viewed as either numerical values or (nominal) categorical labels. Practically, viewing user rating scores as numerical values or categorical labels cannot precisely reflect the exact degree of user preferences. It is expected that for each user, the quantitative distance/scale between any pair of adjacent rating scores could be different.

In this chapter, we propose a new ordinal regression approach. We view ordered preference scores in an additive way, where we are able to model users' internal rating patterns. Specifically, we model and learn the quantitative distances/scales between any pair of adjacent rating scores. In this way, we can generate a mapping from users' assigned discrete rating scores to the exact magnitude/degree of user preferences for items. In the application of rating prediction, we combine our newly proposed ordinal regression method with matrix factorization, forming a new ordinal matrix factorization method. Through extensive experiments on benchmark datasets, we show that our method significantly outperforms existing ordinal methods, as well as other popular collaborative filtering methods in terms of the rating prediction accuracy.

3.1 Introduction

Accurately predicting user preferences/ratings over items can be beneficial to many applications, such as recommender systems, personalized search, online advertising, etc. Generally speaking, rating prediction is the task of predicting a user's rating score for a given item based on her/his past ratings or other information [71]. Given a sparse user-item rating matrix $\mathbf{R}$ where observed ratings take discrete values, the rating prediction task boils down to imputing

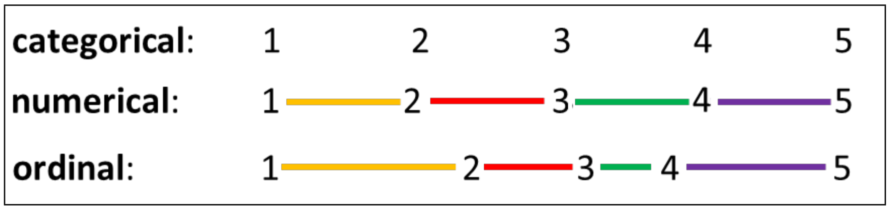


Figure 3.1: An example to show the difference among (nominal) categorical, numerical and ordinal values/labels. The colored (if visible) horizontal lines show the distances between pairs of adjacent rating scores.

the unknown entries in $\mathbf{R}$ based on a small portion of observed ratings. Collaborative filtering (CF) is one of the prominent approaches for rating prediction task [72].

In most collaborative filtering algorithms, user ratings are often viewed as numerical values or (nominal) categorical labels. Such a numerical or categorical view may not precisely capture the exact magnitude/degree of user preferences. Practically it is expected that for each user, the scale between any pair of adjacent rating scores should be different. For example, assuming that rating values range from 1-star to 5-star, one user may be clear about the difference between assigning a 5-star rating or a 4-star rating, while he/she may be confused about assigning a 4-star rating or a 3-star rating. In this case, the distance between "5-star" and "4-star" should be greater than that between "4-star" and "3-star" if we quantify the star-based rating values to the real magnitude in terms of user preferences. On the other hand, different users may have distinct degrees of sensitivity over a sequence of ordered rating scores, resulting in that the distances between any pair of adjacent ratings (e.g., 5-star and 4-star) are different for individual users. Nevertheless, when user ratings are interpreted as numerical values, it equals to the assumption that the user preferences modeled by several consecutive integers (e.g., "1", "2" to "5") are equidistant. While from the viewpoint of (nominal) categorical labels, the distances between pairs of ordered scores are ignored. An example of three different views of rating scores is shown in Fig. 3.1. In view of aforementioned reasons, it is considered to be more natural to view user ratings as ordinal.

In this work, we propose a new ordinal regression model. Most of current statistical ordinal regression methods, e.g., logistic ordinal regression [73], ordinal classification [60, 74], can be performed using a generalized linear model [74]. They view ordered scores as categorical labels and model the ordinality by transforming the cumulative probabilities of these ordered scores to their posterior probability distributions. Different from those ordinal regression models viewing preference scores as ordinal categorical labels, we introduce a new ordinal regression approach by viewing ordered and discrete rating scores as *ordinal real values*. The advantage of our ordinal method is that we can learn a (quantified) mapping from discrete rating scores to the exact magnitude/degree in terms of user's internal preferences. Specifically, we view ordered scores in an additive way, and by optimizing a utility function (e.g., squared loss between observed and estimated scores) which is defined in a measurable ordinal scale, we will learn the exact magnitude between any pair of adjacent rating scores. By correctly learning the personalized mappings for individual users, it is reasonable to expect that the rating prediction can be more accurate.

We apply our newly proposed ordinal regression model in collaborative filtering by combining matrix factorization, forming a new ordinal collaborative filtering method, called "decomposed matrix factorization (DMF)". More specifically, in the first step, our ordinal regression method introduces a decomposition of a sparse rating matrix into a sequence of decomposed binary matrices and matrix factorization is then applied on each of the decomposed matrices. The final predicted rating scores are formed as a weighted sum of the predictions from each of the decomposed matrices, where the weights model the internal distances/scales between different pairs of adjacent rating values. Through extensive experiments on benchmark datasets, we demonstrate that DMF is competitive, often substantially better than existing ordinal collaborative filtering methods, as well as other popular rating prediction algorithms.

3.2 Related Work

Rating prediction has been by far widely discussed in the literature. Collaborative filtering (CF) is the most popular approach for the rating prediction problem, which is based on the assumption that users who have expressed similar interests in the past will share common interests in the future [1, 75, 76].

There are several broad categories of collaborative filtering approaches developed for predicting ratings [36, 77]. One category of common approaches: neighborhood-based CF algorithms [61, 62, 78], also referred to as memory-based methods, predict the rating that one user gives to an item by aggregating the ratings from similar users to that item (i.e., user-based CF) or from the ratings given by that user to similar items (i.e., item-based CF). Another class of so-called model-based approaches use machine learning and data mining methods in the context of predictive models. Matrix factorization is the most popular one, which imposes a low-rank assumption on the given sparse U-I rating matrix $\mathbf{R}$ [39, 79, 42, 40, 41, 80, 81, 82, 83]. In general, based on the low-rank assumption, it factorizes $\mathbf{R}$ into $\mathbf{U}\,\mathbf{V}^{\intercal}$, where $\mathbf{U} \in \mathbb{R}^{m \times f}$ and $\mathbf{V} \in \mathbb{R}^{n \times f}$ (m is the number of rows, n is the number of columns in $\mathbf{R}$ and f is the rank for approximation). This category of matrix factorization approaches have attracted great attention because of their scalability and high prediction accuracy, as demonstrated in the Netflix Prize competition [63].

In most of the aforementioned CF models, user ratings are considered as numerical values or (nominal) categorical labels and it is arguable that these two views of rating values may not precisely reflect the degree in terms of user's internal preferences. Thus, a small number of works propose to view rating scores as ordinal categorical labels in the rating prediction problem. For example, in [57, 84] authors apply logistic ordinal regression [73] to model the ordinality of user feedbacks. They feed a robust rating predictor (i.e., SVD++[39]) into the logistic ordinal regression model and jointly learn model parameters of SVD++ and logistic ordinal model by maximizing a likelihood, which is transformed from cumulative probabilities of ordered rating scores. Another model called "Ordinal Matrix Factorization (OMF)" [85] also combines matrix factorization with logistic ordinal regression. In that work, the ordinality of ratings is modeled as cumulative Gaussian density and then the cumulative probabilities are coupled into a hierarchical Bayesian framework. Different from OrdRec, the parameters of OMF are learned in the Bayesian framework through Gibbs sampling, while in OrdRec parameters are learned through gradient method. As shown in [84, 85], ordinal methods generally outperform other rating prediction methods.

Our proposed ordinal regression model differs with logistic ordinal regression in that we are able to learn a mapping to quantify discrete rating scores to the exact degree in terms of individual users' internal preferences. Specifically, we transfer discrete rating scores to the magnitude which is defined in the ordinal scale and learn/train model in the ordinal space. It is expected that by accurately learning individual users' internal rating patterns can generate more accurate rating predictions.

3.3 Additive Ordinal Regression

In this section, we introduce a new ordinal regression method called “additive ordinal regression”, as we define rating scores in an additive fashion.

3.3.1 An Additive Way to View Ordered Values

We view discrete ordered scores in an additive way and define their magnitudes in a measurable ordinal scale. The “magnitude” in the ordinal scale can be interpreted differently in individual applications. For example, in recommender systems, it can be considered as the degree of user preferences. Without loss of generality, let’s consider discrete ordered scores $\{1, 2, ..., C\}$ and let $l(c)$ represent the magnitude of a given ordered score $c \in \{1, 2, ..., C\}$. Assuming that there is a base label which can be viewed as “0” and let the magnitude of this base label be 0, i.e., $l(0) = 0$. We obtain $l(c)$ in an additive way:

$$l(c) = l(c-1) + scale(c-1, c) = \sum_{i=1}^{c} scale(i-1, i) \tag{3.1}$$

where $scale(c-1, c)$ is the scale/distance between ordered scores “$c-1$” and “c”. If discrete ordered scores are viewed as numerical values, then it equals to the assumption that $scale(c-1, c) = 1, \forall c \in \{1, 2, ..., C\}$.

3.3.2 Mapping Numerical Scale to Ordinal Scale

In the above subsection, we compute the magnitude for all the discrete integer values $l(c)$ ($\forall c \in \{1, 2, ..., C\}$). However, we do not know the magnitudes for real values between any two consecutive integer scores, e.g., $l(1.5)$. Here we propose the mechanism to map from the whole numerical scale to the ordinal space. Without loss of generality, we consider continuous values between two discrete ordered values $c-1$ and c, and values in other intervals can be computed the same way.

Given that t locates in the interval $[c-1, c]$, we model the magnitude of t as:

$$\begin{aligned} l(t) &= l(c-1) + scale(c-1, c) \cdot ratio(t, c-1) \\ &= \sum_{i=1}^{c-1} scale(i-1, i) + scale(c-1, c) \cdot ratio(t, c-1) \end{aligned} \tag{3.2}$$

where $l(c-1)$ and $scale(c-1, c)$ are defined in Eq. (3.1). $ratio(t, c-1)$ is a value between [0,1], formulated as follows:

$$ratio(t, c-1) = \frac{dist(c-1, t)}{dist(c-1, t) + dist(t, c)} \tag{3.3}$$

where $dist(\cdot, \cdot)$ is a distance measure (or kernel) (e.g., Euclidean distance). This $ratio$ value tells which score (i.e., $c-1$ or c) t is closer to and how much t is closer to $c-1$ or c than the other one. For example, if $ratio = 0.5$, then t is in the middle of $c-1$ and c.

Given than $t \in [c-1, c]$, Eq. (3.2) can also be represented as:

$$l(t) = \sum_{i=1}^{C} scale(i-1, i) \cdot ratio(t, i-1) \tag{3.4}$$

if we set $ratio(t, i-1) = 1, \forall i, 1 \leq i \leq c-1$, and set $ratio(t, i-1) = 0, \forall i, c < i \leq C$. This kind of setting to the $ratio$ can be explained in that since we know $t \in [c-1, c]$ and the magnitude is defined in an additive way (i.e., adding from small scores), then we should add the magnitudes of all the intervals which are less than $c-1$, while ignore all the intervals which are greater than c.

Additionally, it should be mentioned that the method to calculate the $ratio$ is not unique. Practitioners can develop different functions to calculate the $ratio$ in different applications.

3.3.3 Additive Ordinal Regression

In real applications such as search and recommendation, we generally deal with problems in which we are given an observed (training) set of data and each data entry contains an input feature vector x and a response score y which takes values from a discrete set of ordered scores $\{1, 2, ..., C\}$. The task is to develop a predictive model to estimate a score $\hat{y}$ for a test input x. The estimated score $\hat{y}$ can be either discrete or continuous based on different applications. In this section, we propose a new ordinal regression model, which returns a continuous value as the response for a given input feature vector x, rather than probability distributions over ordered labels which is the general output of logistic ordinal regression.

In our model, the observed scores and estimated responses are defined in a measurable ordinal scale. We cannot directly formulate the predicted value using Eq. (3.2), since we are merely given input feature x and do not know which interval the estimated value $\hat{y}$ will locate in. Therefore, we formulate $\hat{y}$ as follows:

$$\hat{y} = \sum_{c=1}^{C} scale(c-1, c) \cdot ratio(f_{[c-1,c]}(x), c-1) \tag{3.5}$$

where $f_{[c-1,c]}(x)$ is an internal model used to generate the $ratio$ for interval $[c-1, c]$. Since $\hat{y}$ has chance to be located in all intervals, every interval (i.e., $[c-1, c]$, $\forall c$) should potentially contribute to the computation of $\hat{y}$. The degree of contribution for each interval is decided by the $ratio$ value.

The parameters of our proposed ordinal regression method, including $scale(c-1, c), \forall c \in \{1, 2, ..., C\}$ and all the parameters in the internal model $f_{[c-1,c]}(x)$ can be jointly learned by optimizing a squared loss between observed scores and estimated scores on the training samples. It should be emphasized that in our method the objective function is defined in the ordinal scale where the distances more accurately reflect individual users' internal preferences.

Practically, the scale/distance between adjacent ordered scores models a single user's discrimination ability between these two adjacent scores. For example, if we enlarge the interval between labels "4" and "5", and shrink the interval between "3" and "4" (see the example in Fig. 3.1), then it can be explained in that for someone, it is much easier to discriminate between labels "4" and "5" compared with scores "4" and "3". In real applications such as personalized search and collaborative filtering, different users may have different degrees of sensitivity over a sequence of ordered scores, and thus the internal scale between a specific pair of adjacent scores can also be different for individual users.

3.4 Decomposed Matrix Factorization

In this section, we apply our newly proposed ordinal regression method in collaborative filtering for the task of rating prediction. In this problem, a set of ratings or preference scores are recorded by m users over n items, which are represented by an $m \times n$ user-item matrix

$\mathbf{R} \in \mathbb{R}^{m \times n}$. The ratings take discrete ordered values, e.g., 1-star to 5-star, and most of them are absent due to incomplete observations, making $\mathbf{R}$ a highly sparse matrix. The task of rating prediction then boils down to predicting the unknown ratings based on a small portion of observed ratings in $\mathbf{R}$.

We combine matrix factorization with ordinal regression model, forming a new ordinal collaborative filtering method. In order to simplify the notations, we use w^c to replace $scale(c-1, c)$, representing the magnitude between ordered scores $c-1$ and c. We replace $\hat{y}$ with $\hat{r}$ here and use r^c to represent $ratio(f_{[c-1,c]}(x), c-1)$, where the internal model $f_{[c-1,c]}(x)$ is obtained from matrix factorization and feature vector x refers to user and item latent factors. Finally, Eq. (3.5) is reformulated as follows:

$$\hat{r} = \sum_{c=1}^{C} w^c \cdot r^c \tag{3.6}$$

Note that this formulation (3.6) is related to that in [60], with a distinction. For the purpose of ranking, [60] predicted rating as $\hat{r} = \sum_{c=1}^{C} cProb(r = c)$, and the task boils down to estimating the class probabilities, which was formulated both as a multi-class classification problem and as an ordinal classification problem. [60] commented that using either $\sum_{c=1}^{C} 2^c Prob(r = c)$ or $\sum_{c=1}^{C} cProb(r = c)$ does not make a noticeable difference in the ranking accuracy. In this study, however, since our goal is to accurately predict the ratings instead of rankings, we must learn the weights w^c carefully.

3.4.1 Decomposing User Ratings

Before discussing our proposed rating prediction model, we introduce a decomposition procedure: given a sparse rating matrix $\mathbf{R}$ where observed user ratings take values from a set of discrete scores $\{1, 2, ..., C\}$, we decompose the C-level rating matrix $\mathbf{R}$ into C binary matrices $\mathbf{R1}$, $\mathbf{R2}$,..., $\mathbf{R(C)}$ ($\mathbf{R(c)} \in \mathbb{R}^{m \times n}$, $\forall c$). Each binary matrix discriminates between observed ratings no less than a specific ordinal category c and ratings below that ordinal category ($\{\geq c\}$ vs. $\{\leq c-1\}$). For example, the entries in the c-th decomposed matrix $\mathbf{R(c)}$ are set in the following manner: if the observed rating value $r_{u,i} \geq c$ in $\mathbf{R}$, then we set "1" at the corresponding position of $\mathbf{R(c)}$ (i.e., the intersection of u-th row and i-th column); if $r_{u,i} \leq c-1$ in $\mathbf{R}$, then we set "0" at the corresponding position of $\mathbf{R(c)}$; if an entry is missing (unobserved) in $\mathbf{R}$, then its corresponding position in $\mathbf{R(c)}$ is still missing. Therefore, the observed values in $\mathbf{R(c)}$ discriminate rating scores $\{r \geq c\}$ and $\{r \leq c-1\}$. An example is shown in 3.2 when $C = 4$, $m = n = 2$, where a "?" represents a missing entry. On the left side of the vertical line is the original rating matrix $\mathbf{R}$ and the right side of the vertical line are decomposed binary matrices $\mathbf{R1}$, $\mathbf{R2}$, $\mathbf{R3}$, $\mathbf{R4}$.

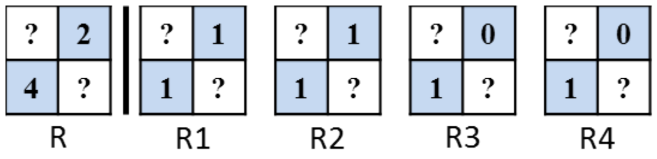


Figure 3.2: Matrix decomposition.

The purpose of matrix decomposition is that each binary matrix can provide information for each interval of the ordered scores. For example, the c-th decomposed binary matrix $\mathbf{R(c)}$ captures information for the interval $[c-1, c]$, since this binary matrix discriminates ratings $\{r \geq c\}$ with $\{r \leq c-1\}$. We then apply matrix factorization on each of the decomposed sparse binary matrices. The imputed values in the entries of binary matrix to some extent model how much the estimations of rating scores are closer to $c-1$ or c, which is interpreted as the $ratio$ value in Eq. (3.3) (or r^c in Eq. (3.6)). We will explain the procedures in details.

3.4.2 Rating Prediction on Decomposed Binary Matrices

We assume that each decomposed binary matrix has low rank and apply matrix factorization on each of them. Specifically, given any matrix $\mathbf{R(c)}$ ($c \in \{1, 2, ..., C\}$), we formulate a rating predictor on the sparse decomposed binary matrix as follows:

$$\begin{aligned} r^c &= \frac{u^c v^{c\intercal} + 1}{2} \\ s.t. \quad & \|u^c\|^2 \leq 1, \quad \|v^c\|^2 \leq 1 \end{aligned} \tag{3.7}$$

where $u^c \in \mathbb{R}^f$ is a row vector of user latent factors and $v^c \in \mathbb{R}^f$ is a row vector of item latent factors on c-th decomposed matrix and f is the dimensionality of low-rank approximation. $r^c \in [0, 1]$ is the predicted value on $\mathbf{R(c)}$. We set constraints $\|u^c\|^2 \leq 1$ and $\|v^c\|^2 \leq 1$ since we expect that the predicted value r^c ranges from 0 to 1. This formulation is inspired by [86].

The predicted value r^c calculated from the latent features in the c-th decomposed matrix represents the "$ratio$" for interval $[c-1, c]$ in Eq. (3.5). Recall that in the c-th decomposed matrix, "1" is set if the original observed rating score "$r \geq c$", and "0" is set when observed rating score "$r \leq c-1$". Thus, the estimated vale of r^c ($0 \leq r^c \leq 1$) indicates how much r is closer to c or $c-1$.

3.4.3 Combining Predictions from Decomposed Matrices

The task of rating prediction is to generate an estimated rating value for a user-item combination. After obtaining the predictions from each of the binary matrices through Eq. (3.7), we then use Eq. (3.6) to generate the final prediction.

In practice, it is expected that each user has distinct rating pattern or degree of sensitivity over the same set of ordered scores. Therefore, the value of w^c (i.e., scale between $c-1$ and c) in Eq. (3.6) should be different for individual users. Then, the prediction model becomes:

$$\hat{r}_{u,i} = \sum_{c=1}^{C} w_u^c \cdot r_{u,i}^c \tag{3.8}$$

where $\hat{r}_{u,i}$ is the final predicted rating value that the u-th user gives to the i-th item and $r_{u,i}^c$ is the prediction in the corresponding entry of the c-th decomposed binary matrix. w_u^c represents that each user has distinct ordinal scale over interval $[c-1, c]$.

By combining Eq. (3.7) and Eq. (3.8), we formulate the final ordinal rating predictor as follows:

$$\begin{aligned} \hat{r}_{u,i} = \sum_{c=1}^{C} \frac{w_u^c}{2}\left(\mathbf{U}_u^c \mathbf{V}_i^{c\mathsf{T}} + 1\right) &= \sum_{c=1}^{C} \frac{w_u^c}{2}\left(\sum_{j=1}^{f} \mathbf{U}_{u,j}^c \cdot \mathbf{V}_{i,j}^c + 1\right) \\ s.t. \quad \| \mathbf{U}_u^c \|^2 \leq 1, \quad \| \mathbf{V}_i^c \|^2 \leq 1 \quad \forall u, i, c \end{aligned} \tag{3.9}$$

where $\mathbf{U}_u^c$ (the u-th row of $\mathbf{U}^c$) and $\mathbf{V}_i^c$ (the i-th row of $\mathbf{V}^c$) denote the latent factors for the u-th user and the i-th item from matrix factorization on the c-th binary matrix.

Finally, model parameters, including the latent factors from each of decomposed matrices and the scale variables between any pair of adjacent rating scores, are jointly learned through minimizing a regularized squared loss between estimated scores and observed scores, which is formulated as follows:

$$\begin{aligned}&\min \sum_{(u,i)\in\Omega} \Big(r_{u,i} - \sum_{c=1}^{C} \frac{w_u^c}{2}\big(\mathbf{U}_u^c \mathbf{V}_i^{c\intercal} + 1\big)\Big)^2 + \sum_{u,c} \lambda_{U,u} \| \mathbf{U}_u^c \|^2 + \sum_{i,c} \lambda_{V,i} \| \mathbf{V}_i^c \|^2 \\ &s.t. \quad \| \mathbf{U}_u^c \|^2 \leq 1, \quad \| \mathbf{V}_i^c \|^2 \leq 1 \quad \forall u, i, c\end{aligned} \tag{3.10}$$

where Ω is the observed set of rating values, and $r_{u,i} \in \Omega$ represents one observed rating score. Shown in the above objective function, we introduce regularization terms (i.e., $\sum_{u,c} \lambda_{U,u} \| \mathbf{U}_u^c \|^2$ and $\sum_{i,c} \lambda_{V,i} \| \mathbf{V}_i^c \|^2$) in order to prevent from overfitting. We call this new ordinal collaborative filtering method "decomposed matrix factorization (DMF)", as our model is based on matrix factorization on a sequence of decomposed matrices.

As an extreme case of our ordinal method, if we set identical weights (i.e., $w^c = 1, \forall c$), then the ratings are still defined as numerical, since now all the ordered rating scores are equidistant (see Fig. 3.1 as an example). Then, our method seemingly just models "residuals" incrementally, where the "residuals" are the difference between small rating values and large rating values. If we use predefined w^c, then DMF is similar to a method called decoupled collaborative ranking proposed in [86] which also introduces the idea of decomposing $\mathbf{R}$.

3.4.4 Comparisons with OrdRec and OMF

There are two principal differences between DMF with OrdRec and OMF, which are implemented based on the logistic ordinal regression method. Firstly, DMF factorizes C decomposed binary matrices where each one captures the insight of user assigning one pair of adjacent rating scores, while OrdRec and OMF factorize only one matrix. This property indicates that DMF essentially catches more patterns when users assign rating scores than the other two methods. Secondly, additive ordinal regression defines the full mapping from users' rating scores to unified ordinal magnitudes, allowing us to easily compare the magnitudes of the same rating score from individual users. Suppose that "4-star" from user A assigned to item i maps to 4.2 (in the ordinal scale) while "4-star" from user B assigned to item i maps to 3.7 (in the ordinal scale) and then we can conclude that user A is probably more interested in item i than user B even though they gave the same rating score to that item. Our ordinal method provides a simpler and more straightforward way to quantify this kind of difference than logistic ordinal regression.

3.5 Parallel SGD for Learning DMF

Methods based on stochastic gradient descent (SGD) have become increasingly important, especially for large-scale industrial applications. It is also fairly easy for practitioners to reproduce the SGD methods. Hence, in this chapter, we develop an algorithm based on SGD to solve the optimization problem in Eq. (3.10). Besides, we adopt a notable share-memory parallelized algorithm, which is called HOGWILD! [87], to make the learning process of our proposed method scalable to multi-core machines.

3.5.1 Regularization Parameters

As shown in Eq. (3.10), we set different regularization parameters for different users and items. In practice, we apply a popular strategy to reduce the number of regularization parameters:

$$\begin{aligned}\lambda_{U,u} =& \frac{\lambda}{\text{\#observed ratings given by user } u} \\ \lambda_{V,i} =& \frac{\lambda}{\text{\#observed ratings gave to item } i}\end{aligned}$$

As a result, we only have one hyper-parameter λ for all the regularization parameters.

3.5.2 Parallelized Stochastic Gradient Descent

In this section, we develop an EM-like SGD algorithm to learn the model parameters. We divide the model parameters of DMF into two groups: $\{\mathbf{U}^c, \mathbf{V}^c\}_{c=1}^C$ and $\{w^c\}_{c=1}^C$. The first set of parameters represent the user and item latent factors and the other set of parameters are the weights set for each of the decomposed matrices, modeling the different scales/distances between any pair of adjacent scores. Our SGD algorithm alternatively updates these two set of parameters. Let $e_{u,i}$ denote the predicted error, $r_{u,i} - \hat{r}_{u,i}$. We summarize the EM-like SGD algorithm as follows:

1. Firstly, we initialize the model parameters: $\{\mathbf{U}^c, \mathbf{V}^c\}_{c=1}^C$ and $\{w^c\}_{c=1}^C$. In our method, we initialize $\{\mathbf{U}^c, \mathbf{V}^c\}_{c=1}^C$ from a normal distribution and $\{w^c\}_{c=1}^C$ are all set to 1.

2. We then fix $\{w^c\}_{c=1}^C$ and estimate $\{\mathbf{U}^c, \mathbf{V}^c\}_{c=1}^C$ by minimizing the loss function in Eq. (3.10). Given an observed rating $r_{u,i}$ (i.e., $(u, i) \in \Omega$), we update the model parameters

as:

$$\mathbf{U}_u^c \leftarrow \mathbf{U}_u^c + \eta_1 \cdot (e_{u,i} \cdot w_u^c \cdot \mathbf{V}_i^c - 2\lambda_{U,u} \cdot \mathbf{U}_u^c) \tag{3.11}$$

$$\mathbf{V}_i^c \leftarrow \mathbf{V}_i^c + \eta_1 \cdot (e_{u,i} \cdot w_u^c \cdot \mathbf{U}_u^c - 2\lambda_{V,i} \cdot \mathbf{V}_i^c) \tag{3.12}$$

The constraints in Eq. (3.10) can be satisfied by gradient projection during the update of model parameters: $\mathbf{U}_u^c \leftarrow \frac{\mathbf{U}_u^c}{\|\mathbf{U}_u^c\|}$ and $V_i^t \leftarrow \frac{\mathbf{V}_i^c}{\|\mathbf{V}_i^c\|}$. By looping through all the training samples several iterations, we will obtain a basic estimation of the parameters $\{\mathbf{U}^c, \mathbf{V}^c\}_{c=1}^C$.

3. Similar to step 2, we then fix $\{\mathbf{U}^c, \mathbf{V}^c\}_{c=1}^C$ and estimate $\{w^c\}_{c=1}^C$ by minimizing the same loss function in Eq. (3.10). The update formulation of $\{w^c\}_{c=1}^C$ is given as:

$$w_u^c \leftarrow w_u^c + \eta_2 \cdot e_{u,i} \cdot (\mathbf{U}_u^c \mathbf{V}_i^{c\mathsf{T}} + 1) \tag{3.13}$$

By looping through all the training samples in Ω several iterations, we obtain the updated parameters of $\{w^c\}_{c=1}^C$.

4. We repeat step 2 and step 3 for several rounds until meeting the stopping criteria and return $\{w^c, \mathbf{U}^c, \mathbf{V}^c\}_{c=1}^C$. The procedure stops when the prediction error on the validation set does not decrease or the program exceeds the number of rounds initially set.

We implement scalable DMF model based on the HOGWILD! framework. Let T be the number of threads and S be the sample size for each thread. In the chapter, we take the value $S = \frac{|\Omega|}{T}$. The pseudo-code of the algorithm is described in Algorithm 2.

3.6 Experiments

We empirically investigate the performance of our ordinal regression method in the context of rating prediction.

Datasets Our algorithms are tested on four public datasets. A full statistic of datasets is described in Table 3.1. In each dataset, we randomly partition the data into two parts: 4/5 as training and 1/5 as test. We perform 5-fold cross validation on the training data.

Performance Metrics: Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) are two popular quantified measurements used to evaluate the accuracy of rating prediction.

Algorithm 2: Parallelized SGD for DMF

```
Input  : observed set $\Omega$; max round of update MAXROUND; regularization parameters
         $\lambda$; learning rate $\eta_1$ for user and item latent factors, $\eta_2$ for scale parameters;
         number of threads $T$; $rank$ of approximation
Output: $\{w^c, \mathbf{U}^c, \mathbf{V}^c\}_{c=1}^{C}$
1  while stopping criteria is not met do
2      while stopping criteria is not met do
3          for each thread $t \in \{1, 2, ..., T\}$ do
4              repeat
5                  choose $r_{u,i} \in \Omega$ uniformly at random;
6                  for each $c \in \{1, 2, ..., C\}$ do
7                      update $\mathbf{U}_u^c$ by Eq. (3.11);
8                      update $\mathbf{V}_i^c$ by Eq. (3.12);
9                      $\mathbf{U}_u^c \leftarrow \frac{\mathbf{U}_u^c}{\|\mathbf{U}_u^c\|}$ and $V_i^t \leftarrow \frac{\mathbf{V}_i^c}{\|\mathbf{V}_i^c\|}$;
10                 end
11             until sampling $S$ times is done;
12         end
13     end
14     while stopping criteria is not met do
15         for each thread $t \in \{1, 2, ..., T\}$ do
16             repeat
17                 choose $r_{u,i} \in \Omega$ uniformly at random;
18                 for each $c \in \{1, 2, ..., C\}$ do
19                     update $w^c$ by Eq. (3.13);
20                 end
21             until sampling $S$ times is done;
22         end
23     end
24     $\eta_1 = \frac{\eta_1}{2}$, $\eta_2 = \frac{\eta_2}{2}$;
25 end
```

They are defined as: $\text{RMSE} = \sqrt{\frac{\sum_i (y_i - \hat{y}_i)^2}{N}}$ and $\text{MAE} = \frac{\sum_i |y_i - \hat{y}_i|}{N}$. In our problem setting, y_i is the real rating value in the test set, $\hat{y}_i$ is the predicted rating value and N is the size of the test data.

3.6.1 Effectiveness of DMF

In order to study the principles in our method, we compare the performance of DMF with two baseline methods which have been discussed in the previous sections: (i) **Matrix Factorization (MF)**. In MF, an estimated rating value is represented as $\hat{r}_{u,i} = \mathbf{U}_u \mathbf{V}_i^\intercal$ and these latent factors could be learned through minimizing an objective function in the form of squared loss with

Table 3.1: Statistics of datasets

Datasets	#Users	#Items	Rating Scale	#Ratings	Density
MovieLens100K	943	1,682	1 - 5 (5 levels)	100,000	6.30%
MovieLens1M	6,040	3,706	1 - 5 (5 levels)	1,000,209	4.47%
MovieLens20M	138,493	27,278	0.5 - 5.0(10 levels)	20,000,263	5.29%
Netflix100K	13,632	586	1 - 5 (5 levels)	112,244	1.20%
Netflix1M	48,018	1,777	1 - 5 (5 levels)	1,020,752	1.41%
Netflix	480,189	17,770	1 - 5 (5 levels)	100,480,507	1.18%

quadratic regularization terms. (ii) We preset $w^c = 1$, $\forall c$, in Eq. (3.9), then DMF just models the incremental "residuals" between small rating values and large rating values, where the rating values are defined as numerical values. We call this method **DMF-Res** and set it as a baseline method for comparison. The DMF-Res method is similar to applying decoupled collaborative ranking [86] for the rating prediction task.

Results are reported in different settings of $rank$ (i.e., the f in Eq. (3.9)) for low-rank approximation. In the experiments, we choose $rank$ from the set of $\{20, 50, 100\}$, and the prediction accuracy of different approaches is evaluated in terms of RMSE metric. Other parameters, e.g., learning rate, regularization parameter, are selected by cross-validation. The experimental results on some datasets are shown from Table 3.2 to Table 3.4.

Table 3.2: RMSE on MovieLens100K

Methods	$rank$=20	$rank$=50	$rank$=100
MF	0.925	0.923	0.922
DMF-Res	0.920	0.919	0.917
DMF	**0.904**	**0.902**	**0.901**

Table 3.3: RMSE on MovieLens1M

Methods	$rank$=20	$rank$=50	$rank$=100
MF	0.872	0.870	0.869
DMF-Res	0.867	0.865	0.864
DMF	**0.848**	**0.846**	**0.845**

MF vs. DMF-Res. The difference between MF and DMF-Res is that MF model is based on matrix factorization on the original rating matrix $\mathbf{R}$, while in DMF-Res model, each binary matrix is factorized. Hence, by setting same $rank$ to both methods, DMF-Res has C times

Table 3.4: RMSE on MovieLens20M

Methods	$rank$=20	$rank$=50	$rank$=100
MF	0.812	0.808	0.802
DMF-Res	0.790	0.787	0.786
DMF	**0.769**	**0.767**	**0.766**

more model parameters than MF. To make fair comparison between MF and DMF-Res in the setting of equal number of model parameters, we can compare MF of "$rank = 100$" with DMF-Res of "$rank = 20$" on MovieLens100K and MovieLens1M datasets since in both of these two datasets, there exist 5 different rating values. In such setting, Tables 3.2 and 3.3 show that DMF-Res slightly outperforms MF. In view of this, DMF-Res method in which we assume that all the ordered rating scores are equidistant (i.e., $w^c = 1, \forall c \in \{1, 2, ..., C\}$) is still considered as another way of matrix factorization approach to modeling user ratings as numerical.

DMF vs. DMF-Res. DMF and DMF-Res differ in that in DMF, the scales between any pair of adjacent rating scores (i.e., $\{w^c\}_{c=1}^{C}$) are learned, while in DMF-Res, all these variables are preset as identical values. Hence, DMF is an ordinal regression method while DMF-Res does not apply ordinal information. Through the comparisons of DMF with DMF-Res from Table 3.2 to Table 3.4, it clearly shows that the weight variables have significant impact on the rating prediction performance. In other words, modeling user ratings as ordinal is sound for rating prediction and our proposed ordinal regression method works well in this problem.

3.6.2 Comparison with Other Ordinal Algorithms

In this section, we compare our proposed DMF method with two existing ordinal matrix factorization methods: **OrdRec** [57] and **OMF** [85], which are both built on the logistic ordinal regression framework. In OrdRec, authors feed SVD++ rating predictor as an internal model. Model parameters of OrdRec are learned through gradient ascent on a likelihood. OMF is a hierarchical Bayesian model and ordinal information of ratings is coupled into the model using logistic ordinal regression by changing logistic function to the cumulative Gaussian density.

The performance comparison is shown in Table 3.5. All the methods are compared in the

Table 3.5: Comparing DMF with ordinal rating predictors

Datasets	Metrics	OrdRec	OMF	DMF
MovieLens1M	MAE	0.672	0.675	**0.666**
	RMSE	0.857	0.862	**0.846**
MovieLens20M	MAE	0.603	0.608	**0.586**
	RMSE	0.783	0.787	**0.766**
Netflix1M	MAE	0.735	0.738	**0.722**
	RMSE	0.932	0.938	**0.914**
Netflix-Training	MAE	0.662	0.670	**0.642**
	RMSE	0.851	0.858	**0.823**

same number of latent factors. More specifically, we set $rank = 20$ for DMF and $rank = C \cdot 20$ for OrdRec and OMF. Based on the t-test results, it is noticeable that on all large-scale datasets, the improvement of our prediction algorithm against all the other ordinal approaches (i.e., OrdRec and OMF) are statistically significant at p-value< 0.01. The results show the superiority of additive ordinal regression over general logistic ordinal regression on the rating prediction problem.

The results of performance comparison among these ordinal methods can be explained in that: logistic ordinal regression focuses on learning a correct probability distribution over a sequence of ordered rating scores, while our proposed ordinal regression method focuses more on learning the correct mapping from user assigned rating scores to the exact degree of user preference. That is, our ordinal regression model has greater power in quantifying the given rating score to a preference score. Besides, evaluation metric such as RMSE is calculated using squared distance, and it is possible that directly minimizing this loss (in our method) is more suitable than a classification loss (logistic ordinal regression). Other advantages of DMF over OrdRec and OMF are explained in Section 3.4.4.

3.6.3 More Discussions on DMF

The impact of rank for matrix approximation

The $rank$ is a tuning parameter for matrix factorization. From Table 3.2 to Table 3.4, we observe that increasing the value of $rank$ can always slightly improves the prediction performance. In order to investigate the impact of $rank$ more detailedly, we report the prediction performance of MF and DMF on the MovieLens20M dataset as a function of $rank$ in Fig. 3.3, which also shows that DMF is able to achieve a stable performance at a relative lower $rank$.

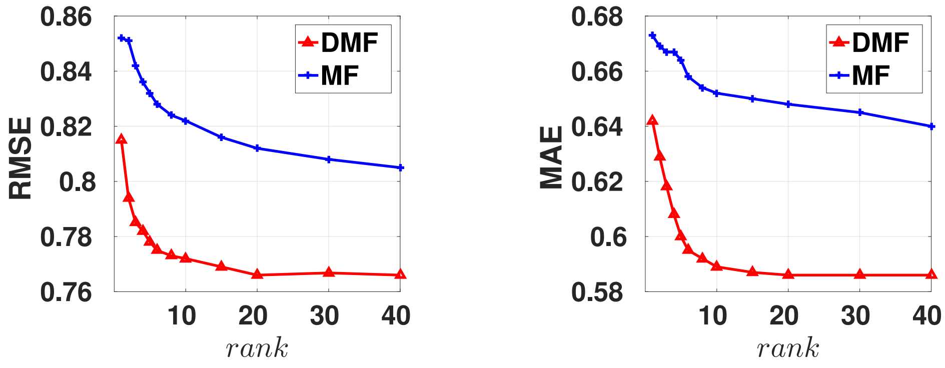


Figure 3.3: The impact of $rank$. Results are reported on the MovieLens20M dataset.

Discussion on the number of decomposed matrices

Given that observed rating scores take from a discrete set of values $\{1, 2, ..., C\}$, an interesting question concerning our ordinal regression method is that why rating values are decomposed into C different intervals, rather than less interval scales such as $\frac{C}{2}$ or more intervals such as $2C$. In order to answer this question, we conduct experiments by decomposing the original rating matrix into different number of matrices and investigate the prediction performance as a function of the number of decomposed matrices.

The performance on MovieLens1M dataset is shown in Fig. 3.4. In that dataset, ratings take discrete values from $\{1, 2, 3, 4, 5\}$. We decompose the original rating matrix into different number of binary matrices in each experiment. For example, if we only obtain 1 binary matrix, then this matrix only distinguishes ratings between two groups: $\{1, 2\} \leq \frac{5}{2} = 2.5$ and $\{3, 4, 5\} > \frac{5}{2} = 2.5$. Readers can also understand it as we treat C in Eq. (3.9) as a tuning parameter, and test this parameter here.

The results in Fig. 3.4 show that: (i) when the number of decomposed matrices ≤ 5, we obtain better performance as the increasing number of decomposition; (ii) While the number of decomposed matrices ≥ 8, we obtain poorer prediction performance as the increasing number of decomposed matrices. The first observation can be explained in that as the number of

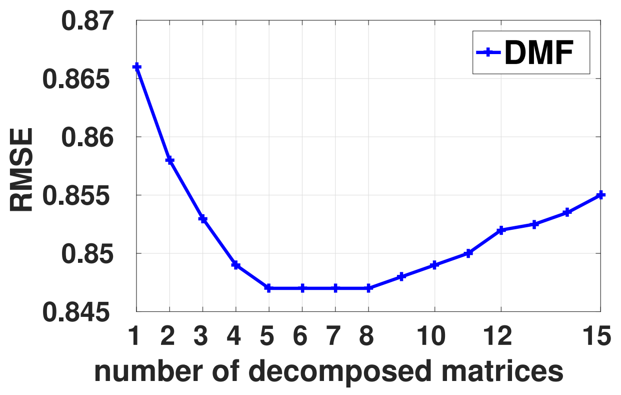


Figure 3.4: The performance when the original rating matrix is decomposed into different number of matrices.

decomposition increases (when this number ≤ 5), we captures more information about the distinct rating values. When we have 5 decomposed matrices (i.e., we model 5 intervals), the information of distinct ordered scores is fully used. When using more than 5 intervals, we actually capture redundant information. For example, if scores are decomposed into 10 intervals, i.e., < 1, [1,1.5), [1.5-2),..., [4.5-5.0), $\geq$5, some intervals (e.g., [1.5, 2), [2.5, 3)) are meaningless, since no score in the training set locates in this interval. The performance becomes poorer due to overfitting. Therefore we choose the number of decomposed matrices as the number of distinct ordered scores.

Effectiveness of parallelized SGD

In Fig. 3.5, we empirically evaluate the effectiveness of our proposed parallelized SGD algorithm on the two largest datasets in terms of RMSE metric. The prediction accuracy after each round of updating parameters is reported. As shown in Algorithm 2, in each round, we first fix $\{w^c\}_{c=1}^C$ and update $\{\mathbf{U}^c, \mathbf{V}^c\}_{c=1}^C$ and then fix $\{\mathbf{U}^c, \mathbf{V}^c\}_{c=1}^C$ and update $\{w^c\}_{c=1}^C$. Therefore, there are 2 experimental results shown within one round on the performance curve. In Fig. 3.5, it shows that after each update of model parameters the prediction performance can be gradually improved. We also observe that much of the improvement takes place in the first 2 rounds and after 3 rounds the performance becomes stable and thus in practice, it might be enough to update model parameters in only 2 or 3 rounds.

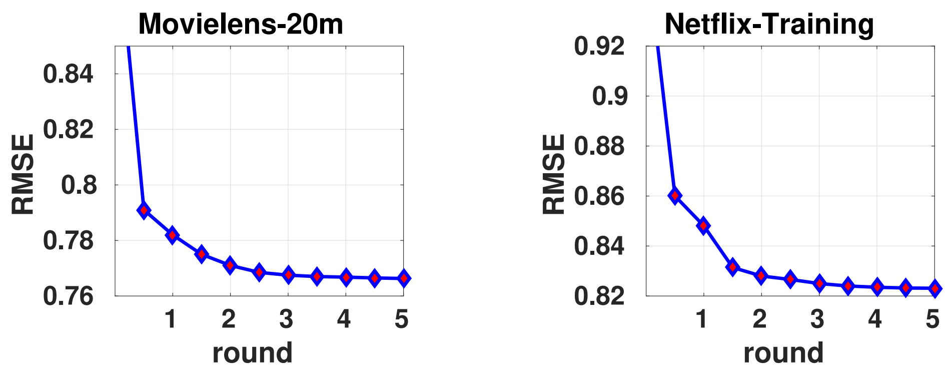


Figure 3.5: The effectiveness of Algorithm 2. In each round, we first report the results after updating $\{\mathbf{U}^c, \mathbf{V}^c\}_{c=1}^C$ and then report the results after updating $\{w^c\}_{c=1}^C$.

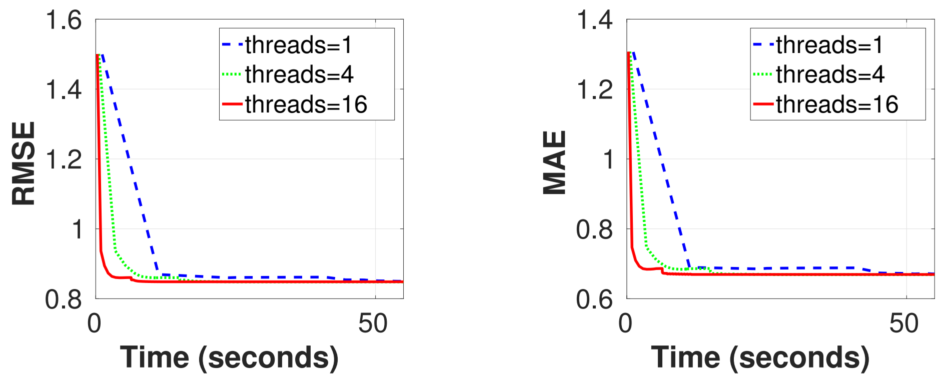


Figure 3.6: The accuracy of parallelized SGD measured by RMSE and MAE with respect to running time. The experiments are reported on MovieLens1M dataset with $rank = 20$.

We also demonstrate the parallelization performance of our proposed algorithm in Fig. 3.6. All the experiments were conducted on a single workstation: DELL Precision T7600 with two Intel@Xeon(R) E5-2687W 3.10GHZ 16-core CPUs. Fig. 3.6 plots the prediction performance as a function of running time. We see that the rating prediction accuracy in terms of both RMSE and MAE metrics converges to similar results in different number of threads.

3.6.4 Comparisons with More Rating Predictors

In this section, we compare DMF with many rating prediction methods, to demonstrate the superiority of our ordinal regression method when applied to the rating prediction problem. We first compare DMF to two baseline neighborhood-based methods: **UserAvg** which predicts

Table 3.6: Comparisons of rating predictors.

Datasets	Metrics	UserAvg	ItemAvg	MF	BPMF	BSVD	FM	DMF
MovieLens100K	MAE	0.835	0.817	0.723	0.722	0.723	0.721	**0.710**
	RMSE	1.041	1.025	0.923	0.920	0.917	0.912	**0.901**
MovieLens1M	MAE	0.830	0.782	0.704	0.696	0.673	0.668	**0.666**
	RMSE	1.036	0.978	0.869	0.865	0.862	0.855	**0.846**
MovieLens20M	MAE	0.751	0.732	0.638	0.632	0.609	0.599	**0.586**
	RMSE	0.963	0.941	0.802	0.798	0.788	0.783	**0.766**
Netflix100K	MAE	0.872	0.871	0.851	0.858	0.852	0.848	**0.826**
	RMSE	1.122	1.120	1.126	1.098	1.088	1.075	**1.035**
Netflix1M	MAE	0.810	0.809	0.742	0.739	0.735	0.734	**0.722**
	RMSE	1.024	1.023	0.947	0.943	0.935	0.933	**0.914**
Netflix-Training	MAE	0.812	0.809	0.687	0.677	0.667	0.656	**0.642**
	RMSE	1.016	1.015	0.880	0.865	0.855	0.843	**0.823**

rating values by averaging the ratings which are given by the same active user; **ItemAvg** which predicts rating values by averaging the ratings given to the same item. We then compare DMF to model-based methods: **MF** as a baseline method. **BPMF** [41], a Bayesian probabilistic matrix factorization method; **BSVD** [40] is an improved regularized matrix factorization approach; **Factorization Machine (FM)** [80] is popular in recommender systems, which can mimic a lot of matrix factorization methods (e.g., SVD++ [39]). In the experiment, we model the feature of FM as SVD++ (see section 4.1.3 in paper [80]) since it achieves the best performance. The main purpose of this chapter is to discuss our newly proposed ordinal regression approach and therefore we mainly compare DMF with other methods which also apply the same basic predictive model-matrix factorization. Practically, we can also combine other basic predictive models such as neural network in our ordinal regression method and we leave it as future work.

We report our settings used in the DMF method: the regularization parameter λ is set as 0.5; learning rates $\eta_1 = 10^{-2}$ and $\eta_2 = 10^{-3}$ are used in the first round, and updated by $\eta_1 = \eta_1/2$ and $\eta_2 = \eta_2/2$ in the following rounds. All the other matrix factorization-based methods are compared in the setting of equal number of model parameters. For example, if there are 5 distinct rating scores and we set $rank = k$ in DMF, then we will set $rank = 5k$ in other methods. This setting of comparison is fairer than the setting of comparison in same $rank$, considering both of the model complexity and performance. If we compare in the setting of same $rank$, one may still doubt that the improvement of performance results from adding more model parameters. Besides, DMF with $rank = k$ generally performs slightly worse than DMF with $rank = C \cdot k$, which demonstrates more superiority of our proposed method if it even outperforms other methods in the setting of $rank = k$. In our experiments, we set $rank = 20$ for DMF, and set $rank = C \cdot 20$ for other compared methods.

The empirical results are reported in Table 3.6. Observed from that table, we see that DMF outperforms all the methods for comparison in terms of both of RMSE and MAE metrics. Except for the first two neighborhood-based models, all the other compared methods apply the same basic predictive model (i.e., matrix factorization) by incorporating other understanding of the patterns when users assign rating scores. For example, they combine more elements such as mean, user's rating bias, user's rating history with basic user and item latent factors. However, all of the compared methods in this section did not introduce the ordinal interpretation of rating scores. From the performance comparisons shown in Table 3.6, we can conclude that modeling the ordinal nature of user preference in a correct manner can significantly improve the performance of rating prediction over methods which ignore the ordinal nature of user preference scores.

3.7 Conclusion

In this chapter, we discussed the ordinality of user ratings: the distance/scale between any pair of adjacent scores should be different. Based on this intuition, we proposed a new ordinal regression approach to modeling the ordinal nature of user ratings. Different from current statistic ordinal regression methods which consider ordered scores as ordinal categorical labels, in our method discrete rating scores can be mapped to the exact magnitudes in terms of individual users' internal preferences. By combing matrix factorization, we applied our proposed ordinal method in collaborative filtering for the task of rating prediction. Through extensive experiments on popular datasets, we demonstrated the superiority of our proposed method over other ordinal methods in terms of the accuracy of rating prediction. We also compared our method with many notable collaborative filtering methods and show the effectiveness to model ordinal user preference scores in order to improve the rating prediction accuracy.

Chapter 4

Improved Bradley-Terry Model for Collaborative Ranking

In order to improve the ranking performance of recommender systems in a basic setting where only a user-item rating matrix is observed, many popular approaches combine learning-to-rank strategies with matrix factorization, forming a new category of collaborative ranking methods. In 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, an *identifiability* issue arises. Besides, in some situations, fitting the Bradley-Terry model yields a *numerical challenge* as it may involve an objective function that is unbounded from below. To the best of our knowledge, these two issues have not yet been discussed in the collaborative ranking literature.

In this chapter, we will discuss and develop a simple strategy to resolve these issues. More specifically, we propose an Improved-BT model by adding a penalty term. The resulting objective function of the proposed method has an interesting interpretation: it balances a regularized pairwise model with a regularized regression model-regularized SVD, which is often considered as a good rating predictor. We develop two algorithms for Improved-BT: a regular stochastic gradient descent (SGD) solver and a sampling-based stochastic gradient descent (SSGD) solver. Moreover, we parallelize these two SGD solvers to make them scalable. Through extensive experiments on benchmark datasets, we show that Improved-BT exhibits excellent performance at different sparsity levels of user-item rating matrices. We also show that our proposed method outperforms many considered state-of-the-art collaborative ranking approaches in terms of both ranking performance and time efficiency.

4.1 Introduction

The main purpose of recommender systems (RS) is to make suitable recommendations of items that are potentially interesting to users. In this chapter, we also consider the basic setting of recommendation: a set of ratings are recorded by m users over n items, and represented by an $m \times n$ user-item (U-I) matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$. It is a common situation that the ratings take discrete values, e.g., 1 to 5, and most of them are absent due to incomplete observations, making $\mathbf{R}$ a highly sparse matrix. The task of recommendation in such setting boils down to selecting the items whose rating values are unobserved and that are predicted a high rating.

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 [88]. In the literature of CF, matrix factorization (MF) might be the most popular method (e.g., [44, 43, 39, 41, 40, 38]) due to its scalability and high prediction accuracy. In matrix factorization, the sparse rating matrix $\mathbf{R}$ 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 [36] propose to aggregate similar users or items and predict the unobserved ratings based on the collected similar users or items (e.g., [61, 62]). It should be mentioned that most of the CF methods aim at improving the rating prediction accuracy, which is usually measured by root-mean-square error (RMSE).

However, in modern recommender systems, it is considered more crucial to investigate the ranking performance, especially at the top of the ranked list of items. The normalized discounted cumulative gain (NDCG) has probably become the most popular measure for modern recommendations. In order to improve the ranking performance of recommender systems, it has been widely proposed to combine learning-to-rank strategies with matrix factorization [56], forming a new class of collaborative ranking methods.

In collaborative ranking, it is quite common to treat user ratings as ordinal. The Bradley-Terry model is one of the most common approach for modeling pairwise preferences (e.g., [36, 65, 66, 89, 56]). In many other fields (e.g., learning-to-rank) where the Bradley-Terry model is widely applied, in order to ensure the **identifiability** of the solutions, sum-to-zero or sum-to-a-constant constraints are typically used to control the range of the model parameters

[90]. However, when the Bradley-Terry model is combined with matrix factorization, the corresponding constraints become non-convex, rendering the resulting optimization problem challenging. Besides, if we directly apply the Bradley-Terry model without any modification, we may encounter an implicit **numerical challenge**: if we represent the pairwise user preferences as graph and in some cases, adding and subtracting a same constant from specific nodes will not change the preference graph, however the value of the objective function of the Bradley-Terry model can go to infinity. We will discuss these issues in detail in this chapter.

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. To the best of our knowledge, the presented chapter is the first to discuss and deal with the above two issues when the Bradley-Terry model is combined with matrix factorization for the purpose of collaborative ranking. At the same time, the Improved-BT model gives rise to another interesting interpretation: it balances a regularized pairwise model with a regularized SVD-like model (regularized SVD is considered as a good rating predictor), although they are slightly different in the form of accessing the input data. We develop two algorithms for optimizing the Improved-BT: a general stochastic gradient descent (**SGD**) solver and another sampling-based stochastic gradient descent (**SSGD**) solver. Moreover, we make both of these two solvers scalable by adopting a parallelized SGD framework. Finally, through extensive experiments on popular benchmark datasets, it is demonstrated that the proposed approach can achieve significant improvements with regard to ranking performance and time efficiency relative to state-of-the-art collaborative ranking approaches.

4.2 Related Work

4.2.1 Pairwise Learning to Rank

In the field of learning-to-rank (LTR), the main problem is to estimate a ranking function that maps given feature vectors to relevance scores, e.g., given a query as input, a ranking web page is returned by a search engine. In LTR, it is quite popular to model relevance scores as pairwise, e.g. SVMRank [91], LambdaRank [92], RankNet [93], RankBoost [94], BoltzRank [95]. In the general document search problem where learning-to-rank techniques are most applied, the content of documents represented as feature vectors plays a significant role in the final ranking performance.

4.2.2 Collaborative Ranking

Collaborative ranking is quite different from LTR. In the basic setting of collaborative ranking, there is no feature vector. The only observed data is a highly sparse user-item rating matrix. However, the ranking models in LTR can be inspiring for collaborative ranking, e.g., regarding the use of Bradley-Terry model. For example, Rendle et al. [65] and Liu et al. [66] model pairwise comparisons of observed ratings using BT model with low-rank structure. Besides, in [89], the authors formulated an objective function based on the Bradley-Terry model and developed a large-scale non-convex implementation that trains a factored form of the matrix via alternating minimization (which reduces to alternating SVM problems). There are also many other interesting models for collaborative ranking: Cofirank [43] 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 [42] and optimizes pairwise surrogate ranking losses.

4.2.3 Bradley-Terry Model

In 1952, Bradley and Terry [96] 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,

$$P(i > j) = \frac{\exp(\beta_i)}{\exp(\beta_i) + \exp(\beta_j)} = \frac{\exp(\beta_i - \beta_j)}{1 + \exp(\beta_i - \beta_j)} \tag{4.1}$$

where β_i and β_j can be regarded as relevance scores for item i and item j (β_i and β_j may have different interpretations in different settings).

The Bradley-Terry model can also be easily extended to model the preferences among a set of objects, for example, let $P(i > \{j, k\})$ denote that i is preferred to $\{j, k\}$, then

$$P(i > \{j, k\}) = \frac{\exp(\beta_i)}{\exp(\beta_i) + \exp(\beta_j) + \exp(\beta_k)}$$

4.3 Methodology and Strategy

In this section, we first explain how the Bradley-Terry model can be applied in collaborative ranking, and point out two issues that need to be tackled when the Bradley-Terry model is combined with matrix factorization. We then present an effective strategy for this purpose.

4.3.1 Bradley-Terry Model Meets Matrix Factorization

In recommender systems, the relevance scores β_i and β_j in Eq. (4.1) often refer to rating values. Assuming that the rating matrix $\mathbf{R}$ has low rank, β_i and β_j can be obtained through matrix factorization (MF). Suppose that there are m rows and n columns in $\mathbf{R}$, then $\mathbf{R}$ can be approximated by the product of two matrices $\mathbf{U} \in \mathbb{R}^{m \times f}$ and $\mathbf{V} \in \mathbb{R}^{n \times f}$: $\hat{\mathbf{R}} = \mathbf{U}\,\mathbf{V}^\intercal$, where f is the dimensionality/rank of the approximation. Each row of $\mathbf{U}$ and $\mathbf{V}$ describe the latent features for a specific user u and an item i respectively. Therefore, the estimated rating a user u gives to an item i could be obtained by $\hat{r}_{ui} = \langle \mathbf{U}_u, \mathbf{V}_i \rangle$, where $\mathbf{U}_u$ is the uth row of $\mathbf{U}$, $\mathbf{V}_i$ is the ith row of $\mathbf{V}$ 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 $\mathbf{R}_{ui}$ is an observed rating. We replace the reference scores β_i and β_j in Eq. (4.1) with estimated rating values $\hat{r}_{ui}$ and $\hat{r}_{uj}$, and then the pairwise ranking aggregation on the observed pairs can be directly formulated as minimizing the negative log-likelihood function as follows:

$$\begin{aligned}
\text{BT-OPT} &:= -\log \prod_{(u,i,j)\in\Omega} P(\hat{r}_{ui} > \hat{r}_{uj}) \\
&= -\sum_{(u,i,j)\in\Omega} \log P(\hat{r}_{ui} > \hat{r}_{uj}) \\
&= -\sum_{(u,i,j)\in\Omega} \log \frac{\exp(\hat{r}_{ui})}{\exp(\hat{r}_{ui}) + \exp(\hat{r}_{uj})} \\
&= -\sum_{(u,i,j)\in\Omega} \log \frac{\exp(\mathbf{U}_u\,\mathbf{V}_i^\intercal)}{\exp(\mathbf{U}_u\,\mathbf{V}_i^\intercal) + \exp(\mathbf{U}_u\,\mathbf{V}_j^\intercal)}
\end{aligned} \tag{4.2}$$

The above objective function in Eq. (4.2) has been widely used in collaborative ranking for modeling pairwise preferences ([89, 56]). 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**.

ISSUE1: Identifiability issue (non-convex constraints)

In Eq. (4.2), 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 many common problems, such as multiclass logistic regression.

In order to deal with this issue, the usual approach is to impose a sum-to-zero constraint ([97, 90]) to the model. 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 $\mathbf{U}$ and $\mathbf{V}$, we accordingly obtain the constraints $\sum_{i=1}^{n} \mathbf{U}_u \mathbf{V}_i^{\intercal} = 0$ 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 $\mathbf{U}$ and $\mathbf{V}$ are no longer applicable. As a result, dealing with non-identifiability becomes more challenging when the Bradley-Terry model is combined with matrix factorization.

ISSUE2: Numerical challenge (no minimizer of objective function)

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 such that we can not obtain a minimizer of the negative log-likelihood in Eq. (4.2). For example, given a preference graph as shown in Fig. 4.1, 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.

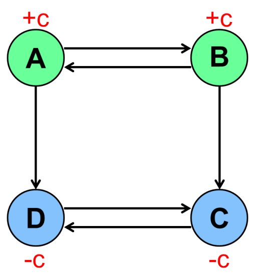


Figure 4.1: The preference graph of an example to show the numerical challenge.

4.3.2 Regularization

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

$$\text{BT-OPT}(1) := -\sum_{(u,i,j)\in\Omega} \log \frac{\exp(\mathbf{U}_u \mathbf{V}_i^\intercal)}{\exp(\mathbf{U}_u \mathbf{V}_i^\intercal) + \exp(\mathbf{U}_u \mathbf{V}_j^\intercal)} + \lambda_U \| \mathbf{U} \|_F^2 + \lambda_V \| \mathbf{V} \|_F^2$$

where $\| \cdot \|_F^2$ represents the squared Frobenius norm. Regularization is a good idea to prevent overfitting in matrix factorization, but it cannot directly deal with the issues we have discussed. Actually, both of the identifiability issue and numerical challenge result from the shift of the values $\mathbf{U}_u \mathbf{V}_i^\intercal$ and $\mathbf{U}_u \mathbf{V}_j^\intercal$, while regularization constrains $\mathbf{U}_u$ and $\mathbf{V}_i$ separately.

4.3.3 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., $\mathbf{U}_u \mathbf{V}_i^\intercal$ and $\mathbf{U}_u \mathbf{V}_j^\intercal$): (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 Fig. 4.1), however the likelihood can go to infinity. Hence, we propose to restrain the shift of $\mathbf{U}_u \mathbf{V}_i^\intercal$ and $\mathbf{U}_u \mathbf{V}_j^\intercal$. For this purpose, we reformulate the objective function by adding a penalty term as follows:

$$\begin{aligned}\text{BT-OPT(2)} := \sum_{(u,i,j)\in\Omega} \Big(& \underbrace{-\log \frac{\exp(\mathbf{U}_u \mathbf{V}_i^\intercal)}{\exp(\mathbf{U}_u \mathbf{V}_i^\intercal) + \exp(\mathbf{U}_u \mathbf{V}_j^\intercal)}}_{\text{pairwise loss}} \\ & + \gamma \underbrace{\left((r_{ui} - \mathbf{U}_u \mathbf{V}_i^\intercal)^2 + (r_{uj} - \mathbf{U}_u \mathbf{V}_j^\intercal)^2\right)}_{\text{penalty}} \Big) \\ & + \lambda_U \| \mathbf{U} \|_F^2 + \lambda_V \| \mathbf{V} \|_F^2 \end{aligned} \tag{4.3}$$

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 can't happen. We call this new method as improved Bradley-Terry model (**Improved-BT**).

The additive penalty term in Eq. (4.3) is similar to a squared regression loss even though they are slightly different in the form of accessing input data: in Eq. (4.3), it accesses observed ratings in the form of pairwise while regression losses usually access data in the form of single point. Even though in this chapter we mainly discuss the pairwise BT model, it should be mentioned that if we delete the pairwise term in Eq. (4.3), then we can still get reasonable ranked lists of items by solely optimizing the penalty term, since in collaborative filtering, it is quite common to set and optimize an regression-based objective function. Therefore, the Improved-BT can be interpreted as a combination of a pairwise model and a regression model.

4.4 Learning

Methods based on stochastic gradient descent (SGD) have become increasingly important, especially for large-scale industrial applications. In this chapter, 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 [87].

4.4.1 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:

$$\lambda_u = \frac{\lambda}{\text{\#observed pairs which contain user } u}$$
$$\lambda_v = \frac{\lambda}{\text{\#observed pairs which contain item } i}$$

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 $\hat{r}_{ui} = \mathbf{U}_u \mathbf{V}_i^\intercal$, $\hat{r}_{uj} = \mathbf{U}_u \mathbf{V}_j^\intercal$, for any training example $(u, i, j) \in \Omega$, partial derivatives of all the model parameters are calculated as:

$$\begin{aligned}
\frac{\partial \text{BT-OPT(2)}}{\partial \mathbf{U}_u} &= -\frac{1}{1+\exp(\hat{r}_{ui} - \hat{r}_{uj})}(\mathbf{V}_i - \mathbf{V}_j) + 2\lambda_u \mathbf{U}_u \\
&\quad + \gamma\big(- 2\,\mathbf{V}_i(r_{ui} - \hat{r}_{ui}) - 2\,\mathbf{V}_j(r_{uj} - \hat{r}_{uj})\big) \\
\frac{\partial \text{BT-OPT(2)}}{\partial \mathbf{V}_i} &= -\frac{1}{1+\exp(\hat{r}_{ui} - \hat{r}_{uj})}\,\mathbf{U}_u + 2\lambda_i \mathbf{V}_i + \gamma\big(- 2\,\mathbf{U}_u(r_{ui} - \hat{r}_{ui})\big) \\
\frac{\partial \text{BT-OPT(2)}}{\partial \mathbf{V}_j} &= \frac{1}{1+\exp(\hat{r}_{ui} - \hat{r}_{uj})}\,\mathbf{U}_u + 2\lambda_j \mathbf{V}_j + \gamma\big(- 2\,\mathbf{U}_u(r_{uj} - \hat{r}_{uj})\big)
\end{aligned}$$

Algorithm 3: Parallel SGD for Improved-BT

```
Input  : observed rating matrix $\mathbf{R}$; the set of observed rating pairs $\Omega$; balance factor $\gamma$;
         regularization parameter $\lambda$; learning rate $\eta$
Output: $\mathbf{U}$ and $\mathbf{V}$
1  while not converged do
2      for each thread $t \in \{1, 2, ..., T\}$ do
3          repeat
4              choose $(u, i, j) \in \Omega$ uniformly at random;
5              update $\mathbf{U}_u \leftarrow \mathbf{U}_u - \eta \frac{\partial \text{BT-OPT(2)}}{\partial \mathbf{U}_u}$;
6              update $\mathbf{V}_i \leftarrow \mathbf{V}_i - \eta \frac{\partial \text{BT-OPT(2)}}{\partial \mathbf{V}_i}$;
7              update $\mathbf{V}_j \leftarrow \mathbf{V}_j - \eta \frac{\partial \text{BT-OPT(2)}}{\partial \mathbf{V}_j}$;
8          until sampling $S$ times is done;
9      end
10     $\eta \leftarrow \frac{\eta}{2}$ ;                                // update learning rate
11 end
```

Let T be the number of threads and S be the sample size for each thread. In the chapter, we take the value $S = \frac{|\Omega|}{T}$. The full algorithm of parallel SGD is described in Algorithm 3.

```
Algorithm 4: Parallel SSGD for Improved-BT
Input  : observed rating matrix $\mathbf{R}$; the set of observed rating pairs $\Omega$; balance factor $\gamma$;
         regularization parameter $\lambda$; learning rate $\eta$
Output: $\mathbf{U}$ and $\mathbf{V}$
1  while not converged do
2      for each thread $t \in \{1, 2, ..., T\}$ do
3          repeat
4              choose $(u, i, j) \in \Omega$ uniformly at random;
5              sample $z \in [0, 1]$ uniformly at random;
6              if $z < \frac{1}{1+\gamma}$ then
7                  update $\mathbf{U}_u \leftarrow \mathbf{U}_u - \eta \frac{\partial \text{PairLoss}}{\partial \mathbf{U}_u}$;
8                  update $\mathbf{V}_i \leftarrow \mathbf{V}_i - \eta \frac{\partial \text{PairLoss}}{\partial \mathbf{V}_i}$;
9                  update $\mathbf{V}_j \leftarrow \mathbf{V}_j - \eta \frac{\partial \text{PairLoss}}{\partial \mathbf{V}_j}$;
10             else
11                 update $\mathbf{U}_u \leftarrow \mathbf{U}_u - \eta \frac{\partial \text{Penalty}}{\partial \mathbf{U}_u}$;
12                 update $\mathbf{V}_i \leftarrow \mathbf{V}_i - \eta \frac{\partial \text{Penalty}}{\partial \mathbf{V}_i}$;
13                 update $\mathbf{V}_j \leftarrow \mathbf{V}_j - \eta \frac{\partial \text{Penalty}}{\partial \mathbf{V}_j}$;
14             end
15         until sampling $S$ times is done;
16     end
17     $\eta \leftarrow \frac{\eta}{2}$ ;                              // update learning rate
18 end
```

Problem with SGD: In practice, we may need to investigate when γ is set to be very large. In such situation, the Improved-BT will solely recover a regression model. However, if we set a large value for γ (e.g., 1000000), then the entry values of the partial derivatives $\frac{\partial \text{BT-OPT(2)}}{\partial \mathbf{U}_u}$, $\frac{\partial \text{BT-OPT(2)}}{\partial \mathbf{V}_i}$, $\frac{\partial \text{BT-OPT(2)}}{\partial \mathbf{V}_j}$ become too large. As a result, after a few iterations of parameter update, the algorithm will encounter a numerical error such that the values of entries in $\mathbf{U}\,\mathbf{V}^\intercal$ may exceed the limitation of a real number that a machine can handle.

4.4.2 Parallel Sampling-based Stochastic Gradient Descent

In this part, we propose a sampling-based SGD method to tackle the problem with SGD.In the community of LTR, it is known that ranking error could be bounded by both of regression error [98] and pairwise error [68]. This fact motivates us to divide BT-OPT(2) into two partitions: a regularized pairwise loss, represented by:

$$\begin{aligned}\text{PairLoss} := -\sum_{(u,i,j)\in\Omega} \log \frac{\exp(\mathbf{U}_u \mathbf{V}_i^\intercal)}{\exp(\mathbf{U}_u \mathbf{V}_i^\intercal) + \exp(\mathbf{U}_u \mathbf{V}_j^\intercal)} \\ + \lambda_U \| \mathbf{U} \|_F^2 + \lambda_V \| \mathbf{V} \|_F^2\end{aligned} \tag{4.4}$$

and the penalty term which is similar to a regularized regression loss:

$$\begin{aligned}\text{Penalty} := \sum_{(u,i,j)\in\Omega} (r_{ui} - \mathbf{U}_u \mathbf{V}_i^\intercal)^2 + (r_{uj} - \mathbf{U}_u \mathbf{V}_j^\intercal)^2 \\ + \lambda_U \| \mathbf{U} \|_F^2 + \lambda_V \| \mathbf{V} \|_F^2\end{aligned} \tag{4.5}$$

We can calculate the partial derivatives of $\mathbf{U}_u$, $\mathbf{V}_i$ and $\mathbf{V}_j$ for these two objective function 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. The balance factor is introduced into our algorithm in the following way: 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 Alg. 4.

4.5 Experiments

4.5.1 Dataset and Setting

Dataset Our algorithms are tested on three popular datasets: MovieLens1M, MovieLens10M[1] and Netflix Prize dataset. MovieLens1M contains 1,000,209 anonymous ratings of 3,706 movies made by 6,040 MovieLens users who joined MovieLens in 2000. The ratings in both of these two datasets range from 1 to 5 (5 stars). The data in MovieLens10M were created by 71,567 users applied to 10,681 movies. Ratings in MovieLens10m are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars). Netflix Prize dataset consists of three parts: training set, probe set and quiz set. The Netflix dataset in this chapter refers to the first part.

Setting In this chapter, we partition each dataset into training and test parts following a popular setup in [43, 42, 89]. For each user, we randomly select N ratings as training samples

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

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%

and all the remaining observed ratings are used as test data. Since we will 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.

Ranking Metric In our experiments, we evaluate our proposed algorithms by Normalized Discounted Cumulative Gain (NDCG). 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 $\mathbf{U}$, 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.

Regularization parameter 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, i.e., 1 to 1000, the ranking performance is not sensitive to the change of λ. Hence, we select $\lambda = 100$ for all the experiments. In the experiment, learning rate is chosen from the set $\{0.1, 0.05, 0.01, 0.005, 0.001\}$. For the reproducibility of our proposed method, we recommend a setting: $\lambda = 100$ and $\eta = 0.01$.

4.5.2 Discussion

We discuss the Improved-BT model through comprehensive experimental study on the Movielens1M dataset. The results measured by NDCG@10 are reported in Fig. 4.2. Several model parameters are set as: regularization parameter $\lambda = 100$, learning rate $\eta = 0.01$, $rank = 100$. We vary the value of γ and the number of training samples N.

Impact of balance factor γ

The first interesting observation from Fig. 4.2 is that the best choice of γ is decided by the size of training samples N. It is shown that when N gets larger, the best choice of γ becomes smaller.

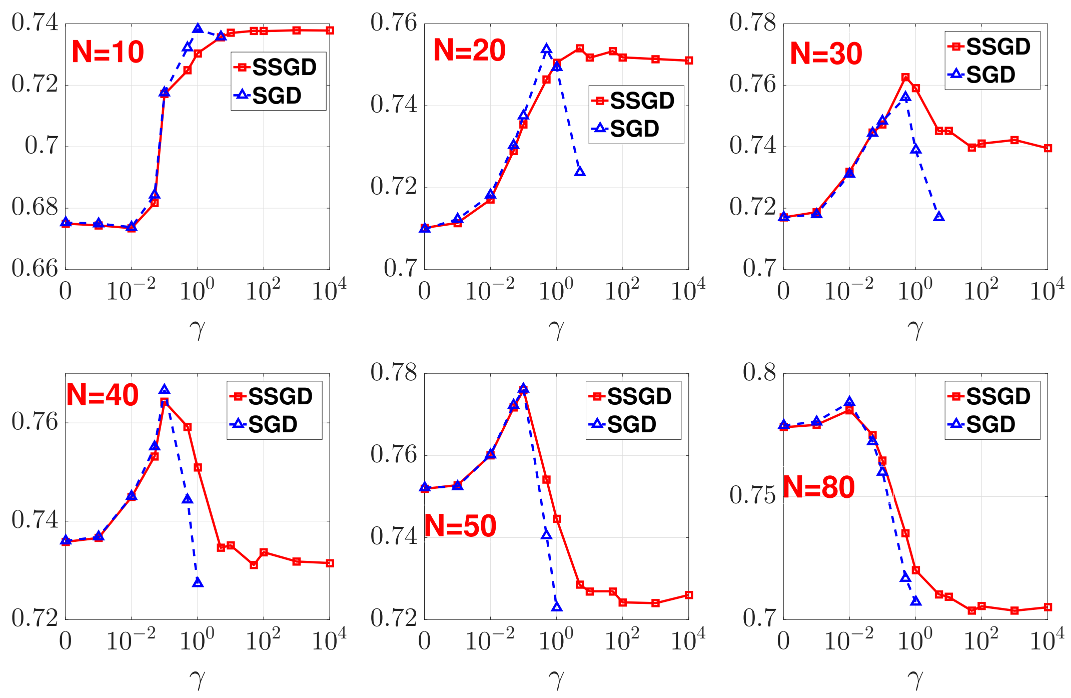


Figure 4.2: Experimental results measured by NDCG@10 on Movielens1M as a function of varying balance factor γ. The number of training samples N is chosen from {10, 20, 30, 40, 50, 80}. The pre-defined parameters include: regularization parameter $\lambda = 100$, learning rate $\eta = 0.01$, and $rank = 100$.

We know that in Eq. (4.3), the objective function recovers pairwise BT model when $\gamma = 0$ and recovers regression model when γ goes to infinity. Accordingly in Fig. 4.2, the 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. In this figure, we can observe that the regression model outperforms pairwise BT model when N is small, while pairwise BT performs better when N is large. This observation can 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. Hence, the pairwise model is given more training samples as the growth of N and more data often results in better performance.

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

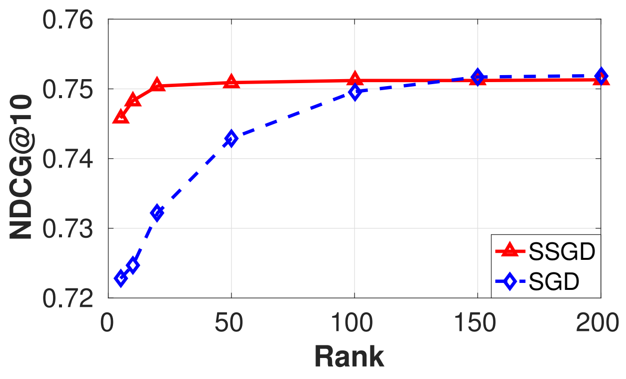


Figure 4.3: The ranking performance of SGD and SSGD on Movielens1M dataset when $N = 20$ and $\gamma = 1$ is reported.

over those two methods when best γ is selected.

Compare SGD with SSGD

In Fig. 4.2, we see that the ranking performance of general SGD and SSGD is almost the same when γ is relatively small (e.g., when $\gamma \leq 0.1$). From this empirical results, it shows to some extent that our proposed sampling SGD method which splits the Improved-BT model into two objective functions and optimizes these two objective functions separately could be considered as another effective SGD method for learning the user and item latent factors $\mathbf{U}$ and $\mathbf{V}$.

Moreover, we point out several advantages of SSGD over SGD: **(I)** SSGD decouples the learning rate with balance factor. As mentioned in section 4.4.1 that in the general SGD algorithm, the learning rate and balance factor are coupled, and hence when γ is large, the SGD algorithm may encounter a numerical error that the estimated rating value may exceed the limitation of the numerical value that a program can handle. This issue is demonstrated through the experiments in Fig. 4.2 that in all of the examples when $\gamma \geq 10$, there is no result reported for SGD since it will get into the numerical trouble. While as shown in the figure, SSGD never runs into such trouble and hence make it possible for us to investigate the ranking performance when γ is relatively large. **(II)** SSGD runs a little bit faster than SGD, because in each iteration of parameter update, SSGD only optimizes one part of the objective function in Eq. (4.3). We will show some results on the running time of these two algorithms in Table 4.1.

Table 4.1: Scalability of proposed methods.

Methods	Cores	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

4.5.3 Parallelization and Scalability

We also demonstrate the scalability of our proposed methods in Table 4.1 and Fig. 4.4. All the experiments were conducted on a single workstation: DELL Precision T7600 with two Intel@Xeon(R) E5-2687W 3.10GHZ 16-core CPUs. Fig. 4.4 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.

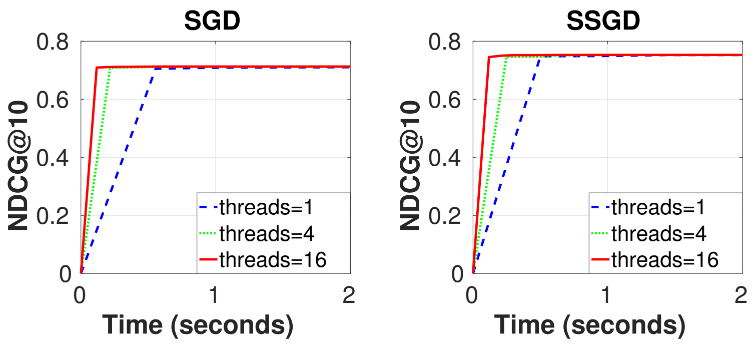


Figure 4.4: The performance of parallel SGD and SSGD on Movielens1M dataset. Parameters are selected as: $N = 20$, $\eta = 0.05$, $\lambda = 100$, and $rank = 100$.

4.5.4 Compare with other methods

Methods for comparison (1) **Factorization Machine (FM)** [44] is a notable rating prediction method. (2) **CofiRank** [43] also known as maximum margin matrix factorization is always considered as a strong baseline method for collaborative ranking. (3) Local Collaborative Ranking (**LCR**) [36] is a state-of-the-art collaborative ranking method where $\mathbf{R}$ is approximated by many locally low-rank matrices. (4) **AltSVM** [89] demonstrates very competitive ranking performance in the original paper, which reduces matrix factorization to alternating

Table 4.2: Comparisons of collaborative ranking methods.

Datasets	Settings	CofiRank	FM	LCR	AltSVM	Improved-BT
Movielens1M	N=10	0.7280 ± 0.0023	0.7166 ± 0.0024	0.6978 ± 0.0031	0.6694 ± 0.0014	**0.7407 ± 0.0012**
	N=20	0.7226 ± 0.0013	0.7194 ± 0.0033	0.7012 ± 0.0025	0.7154 ± 0.0032	**0.7538 ± 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**

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 best settings from their original papers or software. For our proposed Improved-BT, we fix $\lambda = 100$, $rank = 100$, 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 $N = 50$, $\gamma = 0.01$.

Performance comparisons. Table 4.2 shows the comprehensive comparisons of all the methods. We report the NDCG@10 in the settings of $N = 10, 20, 50$ and values in bold-face indicate the best performance. In the table, we can see that Improved-BT performs the best on all the datasets. Since some datasets (e.g., Netflix) are fairly large, the improvement of most cases in the Table 4.2 is fairly significant. Besides, the results show that our proposed approach always obtains the best ranking performance in different settings of N. In particular, when N locates within a specific sparsity range (e.g., $N = 20$ in the experiment), Improved-BT demonstrates more convincing performance such that it achieves 3% to 5% improvement on the ranking performance compared with the best performing baseline method.

Running time. We also report the running time for the comparison methods on Movielens1M dataset in Table 4.3. Our method runs almost 100 times faster than several algorithms, such as LCR and CofiRank, even in the 1-thread setting.

Table 4.3: Running time (seconds) of compared methods.

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

4.6 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 (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 chapter, we proposed an Improved-BT model to address the aforementioned issues. This model could also be interpreted as a combination of pairwise and regression models. 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.

Chapter 5

Collaborative Multi-objective Ranking

This chapter proposes to jointly resolve row-wise and column-wise ranking problems when an explicit rating matrix is given. The row-wise ranking problem, also known as *personalized ranking*, aims to build user-specific models such that the correct order of items (in terms of user preference) is most accurately predicted and then items on the top of ranked list will be recommended to a specific user, while column-wise ranking aims to build item-specific models focusing on targeting users who are most interested in the specific item (for example, for distributing coupons to customers).

In recommender systems, ranking-based collaborative filtering (known as collaborative ranking (CR)) algorithms are designed to solve the aforementioned ranking problems. The key part of CR algorithms is to learn effective user and item latent factors which are combined to decide user preference scores over items. In this chapter, we demonstrate that by individually solving row-wise or column-wise ranking problems using typical CR algorithms is only able to learn one set of effective (user or item) latent factors. Therefore, we propose to jointly solve row-wise and column-wise ranking problems through a parameter sharing framework which optimizes three objectives together: to accurately predict rating scores, to satisfy the user-specific order constraints on all the rated items, and to satisfy the item-specific order constraints. Our extensive experimental results on popular datasets confirm significant performance gains of our proposed method over state-of-the-art CR approaches in both of row-wise and column-wise ranking tasks.

5.1 Introduction

Recommender systems are by far one of the most successful applications of big data and machine learning algorithms. They are an integral part to the success of many giant web/Internet

Table 5.1: An empirical study of BPR for personalized ranking in terms of updating different set of user ($\mathbf{U}$) and item ($\mathbf{V}$) latent factors, evaluated by NDCG@10 for MovieLens1M data. "N" is the number of selected ratings per user for training.

Proposals in optimization	N=10	N=20	N=50
Update both of $\mathbf{U}$ and $\mathbf{V}$	0.6924	0.7168	0.7504
update $\mathbf{V}$, fix $\mathbf{U}$ (uniform initialization)	0.6972	0.7201	0.7527
update $\mathbf{V}$, fix $\mathbf{U}$ (normal initialization)	0.6968	0.7241	0.7522
update $\mathbf{U}$, fix $\mathbf{V}$	0.4831	0.4827	0.4832

companies, e.g., Amazon, Netflix, Google, where a large part of what customers purchase comes from recommendation. The goal of recommender systems is to find what is likely to be of interest to the user, thus enabling personalization and tailored services.

Collaborative filtering (CF) has been one of the most prominent algorithms to accurately predict the user preference. In CF, given a set of user feedback to items, the input data can be viewed as a sparse preference matrix, where rows represent users, columns represent items and entry values indicate the preference scores that users give to items. In this chapter, we study the problem in which preference scores are explicit, taking from a discrete set of values, e.g., a five-star rating system. Two problems are generally discussed in the context of recommendation. On the account of row-wise representation of given preference matrix, the *personalized ranking* task is formed which aims to meet the user-specific information needs, i.e., to obtain a set of interesting items that individual users like the most. On the other hand, if the problem is viewed in a column-wise manner, then the *user ranking* task is formed which aims to find users who are most interested in the specific item.

Ranking-based collaborative filtering, which is also called collaborative ranking (CR) [56], has been proposed for the purpose of generating accurate ranked list of items. The general idea of CR is to combine learning to rank techniques with matrix factorization [65, 42, 99, 86, 43, 100, 101]. The key task in CR algorithms is to learn both of effective user and item latent factors, which jointly generate estimated preference scores over unobserved items. Interestingly, through an experimental study (see Table 5.1), we observe that individually solving row-wise or column-wise ranking objectives is only able to learn one set of effective user or item latent factors. For example, when we apply one of the most popular CR algorithms, Bayesian Personalized Ranking (BPR) [65], to solve the row-wise personalized ranking problem, it is shown

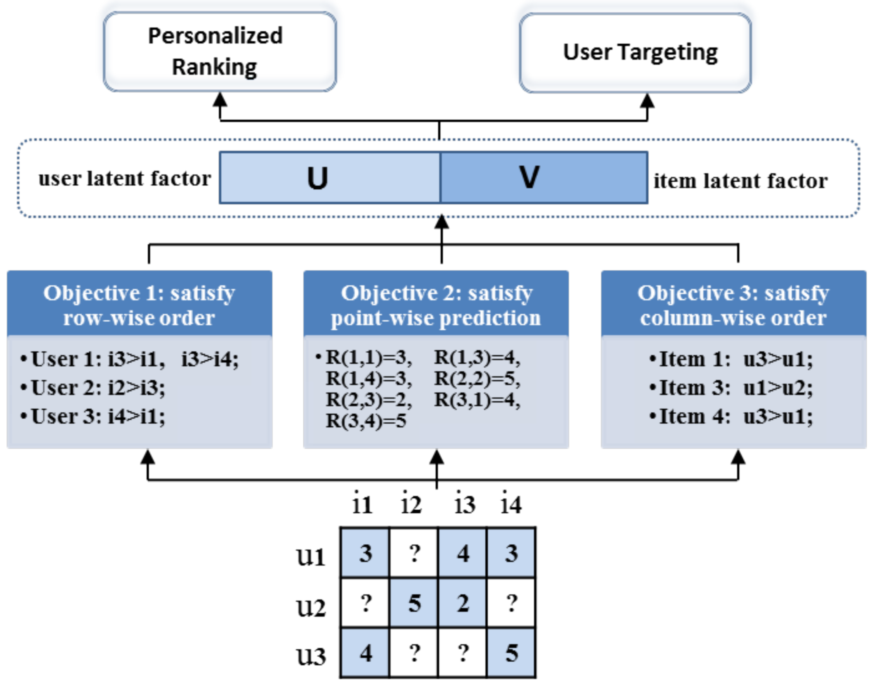


Figure 5.1: Our proposed framework. In this framework, three different objectives which share the same set of user and item latent factors are jointly optimized.

in Table 5.1 that no matter how we update/learn user latent factors $\mathbf{U}$ or even do not update them, the ranking performance is almost the same. This result suggests that we are unable to learn effective user latent factors through BPR. We also test many other CR algorithms (e.g., local collaborative ranking [42], Bradley-Terry model [59], AltSVM [99], etc) and observe that none of them learns effective user latent factors. In the meanwhile, we apply aforementioned pairwise CR algorithms to solve the column-wise ranking problem and also observe that none of them is able to learn effective item latent factors $\mathbf{V}$ for the user ranking task. In Sec. 5.3, we will discuss the problem why we cannot learn effective user latent factors through optimizing row-wise ordered rating pairs.

In order to learn both of effective user and item latent factors, we propose a unified framework (see Fig. 5.1) that simultaneously resolves row-wise and column-wise ranking problems by jointly optimizing three objectives, aiming to satisfy three different constraints: 1) row-wise order constraint: the order of any pair of rating scores a user gives to two items should be preserved; 2) column-wise order constraint: the order of any pair of rating scores an item received from two users should be estimated correctly; 3) pointwise prediction constraint: the prediction of a rating score should be close to its real value. The first and second constraints are apparently designed for personalized ranking and user ranking problems respectively. The third constraint is considered because if all the rating scores are perfectly predicted, then both of correct row-wise orders and column-wise orders can be obtained through comparing the order of predicted rating scores.

With our proposed framework, we are able to learn effective user and item latent factors though jointly optimizing column-wise and row-wise ranking objectives. In addition, by incorporating the pointwise rating regression objective, it brings in additional advantages. For example, tie relationship does provide rich information for ranking purpose, however popular CR models designed for the objectives 1 and 3 in Fig. 5.1 generally ignore tie pairs of rating scores. This issue can be resolved by combining pointwise regression objective with ranking objectives 1 and 3, since optimizing objective 2 will generate similar predicted values for any tie pairs of rating scores, and this result will be propagated to the optimization of pairwise ranking objectives because all the optimization problems share the same set of user and item latent factors.

Despite the potential advantages of the framework, it is a great challenge to jointly optimize multiple objectives than solving each one of them individually. In particular, we have to deal with heterogeneous data inputs during the learning process. To address this challenge, we propose a stochastic gradient method which sequentially optimizes three individual objectives following a simple schema. The learning procedure is explained in details in Sec. 5.5. Compared with state-of-the-art collaborative filtering/ranking algorithms, our proposed method is able to achieve significant performance gains on both row-wise and column-wise ranking tasks.

5.2 Related Work

The literature on recommender systems has been largely focused on the row-wise personalized ranking problem. The problem of column-wise ranking is rarely discussed, as they are two symmetric tasks and most of the row-wise personalized ranking algorithms can be applied to the other task. In the literature of personalized ranking, algorithms can be categorized into three classes: pointwise, pairwise, and listwise, in accordance with the different types of ranking objectives that they optimize [56, 86].

The principle of pointwise methods designed for ranking purpose is that popular ranking metrics (e.g., NDCG [54]) can be approximated by regression loss or classification loss [98, 60]. The set of pointwise approaches have been widely proposed and particularly popularized by the Netflix Prize competition [53] which granted 1 million dollar prize pool for the rating prediction task. As accurately predicting rating scores can generate an accurate ranked list of items, rating prediction algorithms have been widely used in recommender systems. Matrix factorization is probably the most popular one for rating prediction [38, 39, 40, 57, 41, 42, 43, 44, 81, 102], which models the user-item interaction as the inner product of their latent vectors. In particular, algorithms based on matrix factorization have attracted great attention because of their scalability and high prediction accuracy. Matrix factorization methods are viewed as pointwise methods as they generally optimize regression loss or classification loss where data are accessed in the form of single points. In the meanwhile, we should mention that another set of neighborhood-based CF methods are also popular in the task of item recommendation [78, 61, 62, 103, 104].

In recent years, the research for personalized ranking has been shifted from rating prediction to directly optimizing ranking based measurements defined on the observed user feedback. For example, in the literature of Bayesian personalized ranking [65] and other similar works [42, 99, 66, 105, 106, 107], it is proposed to model pairwise comparisons of observed ratings using popular pairwise models (e.g., Bradley-Terry model [96]). Algorithms are also proposed to optimize the surrogate function of listwise ranking measurements. For example, Cofirank [43] optimizes a surrogate convex upper bound of NDCG error. CLiMF [101] and xCLiMF [108] optimize (expected) mean-reciprocal rank (MRR), which has the tendency to obtain at least a few interesting items at the top of ranked list. ListCF [109] directly predicts a total order of items for each user based on similar users' probability distributions over permutations of the items. In addition, it is proposed to combine pointwise and pairwise CR methods to sort items for recommendation [59, 110].

All the aforementioned collaborative ranking algorithms aim to learn effective user and item latent factors and thus enable personalization in the recommendation. However, individually optimizing row-wise or column-wise ranking objective is only able to learn one set of effective latent factors. Therefore, in this chapter, we propose to combine different objectives for both of row-wise and column-wise ranking purpose. The proposed framework not only generates effective user and item latent factors but also resolve several typical issues existing in current popular pairwise CR models.

5.3 Modeling Row-wise Comparisons only generates effective "V"

In this section, we try to explain why popular pairwise CR models only learn one set of effective latent factors. Without loss of generality, we show that modeling row-wise comparisons is only able to learn effective item latent factors $\mathbf{V}$. User latent factors $\mathbf{U}$ learned through row-wise ranking optimization are actually not effective.

5.3.1 Zero-one Loss and Its Approximation

The zero-one error is in general used as the measurement for evaluating the comparisons of ordered pairs. In CR, the preference scores are often obtained through the dot product of user and item latent factors, and then the zero-one error can be formulated as:

$$\mathcal{E}_{\text{zero-one}} = -\sum_{(u,i_1,i_2)\in\Omega} \delta(\mathbf{U}_u\,\mathbf{V}_{i_1}^{\mathsf{T}} - \mathbf{U}_u\,\mathbf{V}_{i_2}^{\mathsf{T}}) \tag{5.1}$$

where $\Omega = \{(u, i_1, i_2) : r_{ui_1} > r_{ui_2}\}$ is the set of pairwise comparisons. $\delta(x)$ is a zero-one loss function (also known as Heaviside function): if $x > 0$, then $\delta(x) = 1$, otherwise $\delta(x) = 0$.

As the zero-one loss is not differentiable, the sigmoid function (i.e., $f(x) = \frac{1}{1+\exp(-x)}$) is routinely applied in CR algorithms to approximate the zero-one ranking loss [65, 42, 59, 89]. Given one pair of ordered rating scores $(u, i_1, i_2) : r_{ui_1} > r_{ui_2}$, the error can be rewritten as follows:

$$\mathcal{E}_{\text{zero-one-approximate}} = -\frac{1}{1+\exp(-\,\mathbf{U}_u\,(\mathbf{V}_{i_1} - \mathbf{V}_{i_2})^{\mathsf{T}})} \tag{5.2}$$

5.3.2 Discussion on User Latent Factors

In the optimization for row-wise comparisons, many ordered pairs of items are formed for each user, and all these ordered pairs from one user share the same user latent factors during optimization. It is obvious that updating $\mathbf{U}$ for one pair of rating scores may destroy the correct order of other pairs. For example, given two item pairs of user u: $\{(u, a, b) : r_{ua} > r_{ub}\} \in \Omega$, $\{(u, c, d) : r_{uc} > r_{ud}\} \in \Omega$, and currently we have item latent factors and their differences: $\mathbf{V}_a - \mathbf{V}_b = [0, -1]$, $\mathbf{V}_c - \mathbf{V}_d = [0, 1]$, then updating $\mathbf{U}_u$ in order to satisfy the correct order of one pair (i.e., updating $\mathbf{U}_u$ to make $\mathbf{U}_u(\mathbf{V}_a - \mathbf{V}_b)^\intercal > 0$ or $\mathbf{U}_u(\mathbf{V}_c - \mathbf{V}_d)^\intercal > 0$) will always destroy the correct order of the other pair. Since there are many ordered pairs of items rated by one user, this issue becomes even more severe. Therefore, it is a great challenge to learn useful user latent factors from row-wise ranking optimization.

5.3.3 More Insight

In this part, we give more insight into the approximation function in Eq. (5.2). We assume "rank=1" of latent factors, i.e., $r_{ui} = U_u \times V_i$, $U_u, V_i \in \mathbb{R}^1$ and compare Eq. (5.2) with the pure logistic function:

$$f(x) = \frac{1}{1 + \exp(-ax)}$$

In personalized ranking, all the ordered rating pairs are formed in the row-wise manner. When processing all the rating pairs given by user u, it will update U_u and the latent factors of items which have ratings assigned from user u. Comparing $f(x) = \frac{1}{1+\exp(-ax)}$ with the formulation in Eq. (5.2), it is intuitive to think of that the functionality of user factor U_u in Eq. (5.2) is similar to that of "a" in logistic function as U_u is shared over all rated items, which is similar to a (a weight parameter shared by all input x). The difference is that in logistic function the input feature x is fixed, while in CR item factors V are model parameters.

In logistic function, the value of "a" determines the shape of the function. In other words, it tells how close the approximation of logistic function to the zero-one loss (see Fig. 5.2). However, in the context of matrix factorization, the change of U_u doesn't necessarily contribute to the change of approximation to zero-one loss as any change to U_u (e.g., double U_u) can be

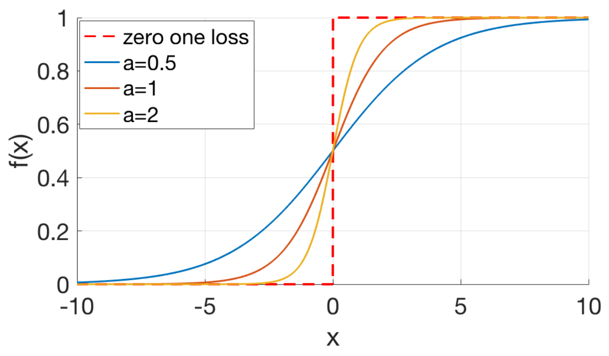


Figure 5.2: Approximation of zero-one loss using logistic function $f(x) = \frac{1}{1+\exp(-ax)}$, for $a = 0.5, 1, 2$.

compensated by changing all the item factors V_i accordingly (e.g., reduce V_i by half). Therefore, updating U_u is unnecessary, unless we set the constraint to V_i. The experimental results shown in Table 5.1 verify the above discussions.

5.3.4 Combining Row-wise and Column-wise Comparisons

Through the above discussion, we know that optimizing row-wise comparisons cannot learn effective user latent factors. Symmetrically, we cannot learn effective item latent factors by solely optimizing column-wise comparisons. In order to deal with this issue, we propose to combine row-wise and column-wise ranking problems together. The procedure is simple: when optimizing row-wise ranking objective, we fix $\mathbf{U}$ and only update $\mathbf{V}$; while during the optimization of column-wise ranking objective, we only update $\mathbf{U}$. This way we can learn both of effective user and item latent factors.

5.4 Proposed Method

In this chapter, we investigate our proposed model on explicit preference scores (e.g., five-star rating system). Given a set of observed rating scores from m users to n items, the data can be represented as a sparse rating matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$ where one observed rating from user u gives to item i is represented as r_{ui}. The task of *personalized ranking* boils down to imputing the missing entries in the sparse rating matrix $\mathbf{R}$, and then sorting items in reverse order of imputed values of each item in a row-wise manner (i.e., user-specific). Then items at the top of ranked lists will be recommended to users. The task of column-wise ranking is symmetric to the task of *personalized ranking*. It aims to target users who are most interested in an item (item-specific), e.g., distributing coupons to customers who are most likely to consume the products.

In order to learn both of effective user and item latent factors, we introduce a learning schema of jointly optimizing multiple objective functions which share the same set of model parameters. The diagram of our proposed method is shown in Fig. 5.1. Given a sparse rating matrix, we design multiple algorithms to deal with heterogeneous data inputs in order to satisfy three different constraints: 1) row-wise order constraint which includes rating pairs that one user gives to two items, 2) column-wise order constraint which introduces rating pairs that one item received from two users, and 3) pointwise constraint which treats the rating score of one user-item combination as a single data point. The key property of this framework is that all the individual models share the same set of model parameters (i.e., user and item latent factors). As this method simultaneously resolves both of row-wise and column-wise ranking tasks, we name the proposed framework to be **collaborative multi-objective ranking (CMR)**.

5.4.1 Rating Prediction through Matrix Factorization

Matrix factorization is probably the most widely used algorithm for the rating prediction task (i.e., objective 2 in Fig. 5.1). The general idea of matrix factorization is to assume that the rating matrix $\mathbf{R} \in \mathbb{R}^{m\times n}$ has low rank and thus it can be approximated by $\mathbf{R} = \mathbf{U}\,\mathbf{V}^{\mathsf{T}}$, where $\mathbf{U} \in \mathbb{R}^{m\times k}$ and $\mathbf{V} \in \mathbb{R}^{n\times k}$ respectively represent user latent factors and item latent factors, and k is the rank of approximation. In matrix factorization, the latent factors $\mathbf{U}$ and $\mathbf{V}$ are typically learned through minimizing a regularized squared loss on the observed training ratings. This prediction loss on the training set is formulated as:

$$L_{\text{pointwise}} = \sum_{u=1}^{m}\sum_{i=1}^{n}(r_{ui} - \hat{r}_{ui})^2 + \sum_{u} \lambda_{U,u} \| \mathbf{U}_u \|^2 + \sum_{i} \lambda_{V,i} \| \mathbf{V}_i \|^2 \tag{5.3}$$

where r_{ui} and $\hat{r}_{ui}$ are respectively the observed and estimated rating scores. The regularized term is set in order to prevent from overfitting. We use $L_{\text{pointwise}}$ to represent the objective function designed for solving objective 2 in our proposed framework (see Fig. 5.1). To optimize the above loss we can apply a general stochastic gradient descent (SGD) method. Given an observed rating score r_{ui}, the gradients of model parameters are calculated as follows:

$$\frac{\partial L_{\text{pointwise}}}{\partial \mathbf{U_u}} = -2(r_{ui} - \hat{r}_{ui})\mathbf{V_i} + 2\lambda_{U,u}\mathbf{U_u} \tag{5.4}$$

$$\frac{\partial L_{\text{pointwise}}}{\partial \mathbf{V_i}} = -2(r_{ui} - \hat{r}_{ui})\mathbf{U_u} + 2\lambda_{V,i}\mathbf{V_i} \tag{5.5}$$

where $\hat{r}_{ui} = \mathbf{U}_u\,\mathbf{V}_i^\intercal$. We then update $\mathbf{U}_u \leftarrow \mathbf{U}_u - \eta\frac{\partial L_{\text{pointwise}}}{\partial \mathbf{U_u}}$ and $\mathbf{V}_i \leftarrow \mathbf{V}_i - \eta\frac{\partial L_{\text{pointwise}}}{\partial \mathbf{V_i}}$ where η is the learning rate.

5.4.2 Pairwise Comparisons through Bradley-Terry Model

We combine matrix factorization with the Bradley-Terry (BT) model [96] for solving the objectives 1 and 3 in Fig. 5.1. The Bradley-Terry model is widely used for modeling pairwise preferences. Let $P(i_1 > i_2)$ denote the probability that item i_1 is preferred over item i_2 for user u. Typically, the model ignores ties and hence considers that $P(i_1 > i_2) + P(i_1 < i_2) = 1$ for all pairs. The Bradley-Terry model is given by:

$$P(i_1 > i_2) = \frac{\exp(\beta_{i_1})}{\exp(\beta_{i_1}) + \exp(\beta_{i_2})} = \frac{1}{1 + \exp(\beta_{i_2} - \beta_{i_1})} \tag{5.6}$$

where β_{i_1} and β_{i_2} are relevance values for items i_1 and i_2. Most of the existing pairwise CR methods (e.g., Bayesian personalized ranking [65]) apply BT to model the pairwise comparisons.

Row-wise comparisons: We resolve objective 1 in Fig. 5.1 by modeling row-wise comparisons of (user-specific) item pairs using Bradley-Terry model together with matrix factorization. The row-wise comparisons can be formulated as follows:

$$P(r_{ui_1} > r_{ui_2}) = \frac{\exp(\mathbf{U}_u\,\mathbf{V}_{i_1}^\intercal)}{\exp(\mathbf{U}_u\,\mathbf{V}_{i_1}^\intercal) + \exp(\mathbf{U}_u\,\mathbf{V}_{i_2}^\intercal)} \tag{5.7}$$

We then minimize negative log likelihood on all the comparisons of observed item pairs, obtaining the following objective function:

$$L_{\text{row-wise}} = -\sum_{(u,i_1,i_2)\in\Omega} \log P(r_{ui_1} > r_{ui_2}) + \sum_i \lambda_{V,i}\|\,\mathbf{V}_i\,\|^2 \tag{5.8}$$

where $\Omega = \{(u, i_1, i_2) : r_{ui_1} > r_{ui_2}\}$ is the set of observed row-wise rating pairs. It should be mentioned that in solving row-wise ranking problem, we only update $\mathbf{V}$.

Again, we apply SGD to learn the model parameters. Given an observed pair $(u, i_1, i_2) \in \Omega$, the derivatives of relevant model parameters are calculated as follows:

$$\frac{\partial L_{\text{row-wise}}}{\partial \mathbf{V}_{i_1}} = -\frac{1}{1+\exp(\hat{r}_{ui_1} - \hat{r}_{ui_2})}\mathbf{U}_u + 2\lambda_{V,i_1}\mathbf{V}_{i_1} \tag{5.9}$$

$$\frac{\partial L_{\text{row-wise}}}{\partial \mathbf{V}_{i_2}} = \frac{1}{1+\exp(\hat{r}_{ui_1} - \hat{r}_{ui_2})}\mathbf{U}_u + 2\lambda_{V,i_2}\mathbf{V}_{i_2} \tag{5.10}$$

where $\hat{r}_{ui_1} = \mathbf{U}_u \mathbf{V}_{i_1}^\intercal$ and $\hat{r}_{ui_2} = \mathbf{U}_u \mathbf{V}_{i_2}^\intercal$.

Column-wise comparisons: The method to resolve objective 3 in Fig. 5.1 is symmetric to that in modeling the row-wise comparisons. Concretely, we model the (item-specific) column-wise comparisons as follows:

$$P(r_{u_1 i} > r_{u_2 i}) = \frac{\exp(\mathbf{U}_{u_1}\mathbf{V}_i^\intercal)}{\exp(\mathbf{U}_{u_1}\mathbf{V}_i^\intercal) + \exp(\mathbf{U}_{u_2}\mathbf{V}_i^\intercal)} \tag{5.11}$$

Then, the objective function becomes:

$$L_{\text{column-wise}} = -\sum_{(u_1,u_2,i)\in O} \log P(r_{u_1 i} > r_{u_2 i}) + \sum_u \lambda_{U,u} \| \mathbf{U}_u \|^2 \tag{5.12}$$

where $O = \{(u_1, u_2, i) : r_{u_1 i} > r_{u_2 i}\}$ is the set of observed column-wise rating pairs. The optimization process is similar to that in the row-wise ranking problem. We also should mention that only $\mathbf{U}$ is updated in optimizing column-wise objective function.

5.4.3 Combining All Three Objectives

Similar to the work in [111] which linearly combines two objectives using one balance factor, we introduce two balance factors $\alpha \in [0, 1]$ and $\beta \in [0, 1]$, s.t., $\alpha + \beta \leq 1$, to combine aforementioned three losses. The final integrated loss is introduced in the following formulation:

$$L = \alpha L_{\text{row-wise}} + \beta L_{\text{column-wise}} + (1 - \alpha - \beta) L_{\text{pointwise}} \tag{5.13}$$

where balance factors α and β are set to model the importance of individual losses. Intuitively, the weight of each loss function should be set differently in solving different problems. For example, when we consider row-wise ranking problem, $L_{\text{row-wise}}$ and $L_{\text{pointwise}}$ are more important than $L_{\text{column-wise}}$, as individually optimizing $L_{\text{row-wise}}$ or $L_{\text{pointwise}}$ is able to provide a reasonable ranked list of items. The key property of our proposed model is that these three individual loss functions are able to communicate with each other during joint optimization as they share the same set of latent factors, i.e., $\mathbf{U}$ and $\mathbf{V}$. This optimization scheme brings in multiple advantages compared with learning three models separately.

5.4.4 The Advantages of CMR

In addition to learning both of effective user and item latent factors, there are other advantages of our proposed CMR model. This section discusses these advantages of learning multiple objectives together, compared with methods that solve each objective separately.

Before discussion on the advantages of our framework, we show some limitations or issues if we solely apply pointwise matrix factorization method or Bradley-Terry models for the personalized ranking or user ranking problem. We then show how our proposed method resolves the mentioned issues.

Limitation of matrix factorization: The major problem of matrix factorization as a pointwise collaborative filtering method is that rating scores in this method are generally considered as numerical values or nominal categorical labels. Both numerical view and the categorical view can't accurately reflect the intuition when users assign rating scores. For example, it is expected that for each user, the distances (in terms of user preferences) between different pairs of adjacent rating scores should not be the same, e.g., "5-star"-"4-star"$\neq$"4-star"-"3-star". Both numerical and nominal categorical views of rating scores ignore this information. As a result, pointwise methods generally perform worse than ranking-based methods for the task of personalized ranking [43, 56, 42].

Issues with pairwise model: There are two typical issues when we combine matrix factorization with the Bradley-Terry model (or the pairwise CR models) for the personalized ranking or user ranking problem.

(i) Ignore tie comparisons: Intuitively, tie comparisons in multi-level rating systems provide useful information for the ranking purpose. However, most pairwise collaborative ranking model, such as the Bradley-Terry model, ignore the tie pairs of rating scores.

(ii) Identifiability issue: This issue is apparent in the Bradley-Terry model [112]. If we add a same constant "c" to both of $\mathbf{U}_u \mathbf{V}_{i_1}^\intercal$ and $\mathbf{U}_u \mathbf{V}_{i_2}^\intercal$ in Eq. (5.7), then the likelihood

$P(r_{ui_1} > r_{ui_2})$ will remain the same, i.e.,

$$\frac{\exp(\mathbf{U}_u\,\mathbf{V}_{i_1}^{\intercal})}{\exp(\mathbf{U}_u\,\mathbf{V}_{i_1}^{\intercal})+\exp(\mathbf{U}_u\,\mathbf{V}_{i_2}^{\intercal})}=\frac{\exp(\mathbf{U}_u\,\mathbf{V}_{i_1}^{\intercal}+c)}{\exp(\mathbf{U}_u\,\mathbf{V}_{i_1}^{\intercal}+c)+\exp(\mathbf{U}_u\,\mathbf{V}_{i_2}^{\intercal}+c)}$$

At a technical level, one says that the parameters are not uniquely identifiable. Similar identifiability issues exist in other common problems, such as multi-class logistic regression. The common practice to deal with this issue is to add a sum-to-zero or sum-to-constant constraint, however, when we apply matrix factorization as the predictive model, the sum-to-zero constraint (i.e., $\sum_{i=1}^{n} U_u V_i^T = 0$) becomes non-convex and thus difficult to handle in the optimization process. In particular, popular optimization methods used in this context such as (stochastic) gradient descent or alternating optimization of $\mathbf{U}$ and $\mathbf{V}$ are no longer applicable.

The limitations or issues aforementioned can be easily resolved by combining the pointwise method with the Bradley-Terry model by sharing the same set of latent factors. By combining these two approaches, it not only incorporates the numerical interpretation of rating scores by optimizing the pointwise regression loss in Eq. (5.3) but also includes the ordinal interpretation of rating scores by optimizing the pairwise ranking loss. Then, tie comparisons will not be ignored in the integrated model, as optimizing pointwise regression loss will push the predictions of any tie pairs of rating scores to be the same, and this result will be propagated to the pairwise ranking loss as these two objectives share the same set of model parameters. Meanwhile, the identifiability issue existing in the Bradley-Terry model can also be resolved by combining it with a pointwise loss. Recall that identifiability issue occurs when adding a same constant "c" to both of the estimated rating values of two items. If we add a regression loss, then the optimization of regression loss will make the estimated rating values converge to the real observed rating scores. Therefore, the same constant can never be arbitrarily added to any estimated scores, as it may increase the regression loss.

The technique of combining pointwise and pairwise CR models is also mentioned in [110], but the underlying reason why the combined model outperforms individual models is not fully discussed.

5.5 Learning

The difficulty in jointly optimizing an integrated objective function (shown in Eq. (5.13)) is that we have to simultaneously deal with heterogeneous data inputs: regression loss requests input data as (numerical) rating values; Bradley-Terry model for personalized ranking requests row-wise rating pairs; while Bradley-Terry model for the task of user ranking requests column-wise rating pairs. In order to handle the heterogeneity of input data, we propose a stochastic gradient descent method by interactively updating the three individual objectives included in Eq. (5.13). The idea is summarized as follows:

(1) we firstly sample a random value a from the uniform distribution between 0 and 1;

(2) if $a < \alpha$, then we randomly sample a row-wise rating pair and update $L_{\text{row-wise}}$; else if $\alpha \leq a < \alpha + \beta$, then we randomly sample a column-wise rating pair and update $L_{\text{column-wise}}$; else, we randomly sample a rating score and update $L_{\text{pointwise}}$;

(3) repeat (1)(2) until the procedure converges or meets the end condition.

Note that in this algorithm, we only update $\mathbf{V}$ when optimizing $L_{\text{row-wise}}$ and only update $\mathbf{U}$ when optimizing $L_{\text{column-wise}}$.

Additionally, we choose to set user and item specific regularization parameters as follows:

$$\lambda_{U,u} = \frac{\lambda}{\text{\#observed pairs/ratings which contain user } u}$$
$$\lambda_{V,i} = \frac{\lambda}{\text{\#observed pairs/ratings which contain item } i}$$

Afterwards, we only have one tuning parameter λ in the regularization term. This is a popular strategy used in many collaborative ranking algorithms ([89, 44]) to reduce the number of tuning parameters. In addition, in order to efficiently learn model parameters, we apply a parallel framework called HOGWILD! ([87]). The concrete parallel SGD algorithm is introduced in Alg. 5 [1].

[1]Code is available at https://github.com/bssbbsmd/Collaborative_Multi-task_ranking

Algorithm 5: Parallel SGD for learning our model

```
Input  : observed rating matrix $\mathbf{R}$; observed (row-wise and column-wise) rating pairs $\Omega$
         and $O$; balance factor $\alpha$ and $\beta$; regularization parameter $\lambda$; initial learning rate
         $\eta$; $I_{max}$: maximum number of iterations
Output: user and item latent factors $\mathbf{U}$ and $\mathbf{V}$
1  while not converged and iteration $< I_{max}$ do
2      for each thread $t \in \{1, 2, ..., T\}$ do
3          repeat
4              sample $a \in [0, 1]$ randomly;
5              if $a < \alpha$ then
                   // optimize objective 1
6                  choose $(u, i_1, i_2) \in \Omega$ uniformly at random;
7                  calculate $\frac{\partial L_{\text{row-wise}}}{\partial \mathbf{V}_{i_1}}$, $\frac{\partial L_{\text{row-wise}}}{\partial \mathbf{V}_{i_2}}$ and update $\mathbf{V}_{i_1}$, $\mathbf{V}_{i_2}$ accordingly;
8              else if $\alpha < a < \alpha + \beta$ then
                   // optimize objective 3
9                  choose $(u_1, u_2, i) \in O$ uniformly at random;
10                 calculate $\frac{\partial L_{\text{column-wise}}}{\partial \mathbf{U}_{u_1}}$, $\frac{\partial L_{\text{column-wise}}}{\partial \mathbf{U}_{u_2}}$ and update $\mathbf{U}_{u_1}$, $\mathbf{U}_{u_2}$ accordingly;
11             else
                   // optimize objective 2
12                 randomly sample an observed rating $r_{ui}$;
13                 calculate $\frac{\partial L_{\text{pointwise}}}{\partial \mathbf{U_u}}$, $\frac{\partial L_{\text{pointwise}}}{\partial \mathbf{V_i}}$ and update $U_u$, $V_i$ accordingly;
14             end
15         until sampling $S$ times is done;
16     end
17     $\eta \leftarrow \frac{\eta}{2}$ ;                                   // update learning rate
18 end
```

5.6 Experiment

Data preparation. We test our proposed method on MovieLens100K [113], MovieLens1M, Netflix1M, Amazon Instant Video datasets [114]. MovieLens100K dataset contains 100,000 rating scores assigned from 943 users to 1,682 items; MovieLens1M dataset contains 1,000,209 rating scores collected from 6,040 users and 3,706 items. Netflix1M dataset contains 1,020,752 ratings assigned by 48,000 users give to 1800 items. Amazon Instant Video dataset is a subset of Amazon review dataset, which contains 583,933 ratings.

Evaluation metric. We apply the standard Normalized Discounted Cumulative Gain (**NDCG**) [54] as the metric for evaluating both personalized ranking and user ranking tasks. NDCG not only evaluates the ranking performance of a ranked list but also emphasizes more on the top of the ranked list. In the real world, most customers are only interested in the items recommended in the first one or two pages. Particularly, on the mobile devices, e.g., smartphones or pads, the performance at the top of the ranked list becomes even more crucial as users can only view few top-ranked items because of the limited size of a screen.

When we evaluate personalized ranking, NDCG score is defined as: $NDCG@K(u) = \frac{DCG@K(u,\pi_u)}{DCG@K(u,\pi_u^*)}$, 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 $DCG@K$. Symmetrically, when evaluating user ranking performance, the permutation is formulated in the column-wise manner.

Settings for experiments. We follow the general setting for testing CR algorithms [43, 42, 86, 99]: when testing the row-wise personalized ranking performance, we sample N rating scores from each row as the training set, and the remaining observed rating scores are used as test data. As NDCG@10 is used as the ranking metric, 10 rating scores must be kept in each row of the test data and thus rows containing less than $N + 10$ rating scores will be dropped. We use $N = 10, 20, 50$, as commonly used in other papers. Symmetrically, when testing the column-wise ranking performance, training pairs and test pairs are selected from each column.

5.6.1 Discussion of Results

To help explain the advantage of CMR (which combines three objectives), we first discuss the combinations of only two objectives.

(i) Combining $L_{\text{row-wise}}$ with $L_{\text{pointwise}}$ for the task of personalized ranking. If we optimize $L_{\text{row-wise}}$ or $L_{\text{pointwise}}$ individually, both methods are able to provide reasonable results for the task of personalized ranking. In this section, we investigate the idea by combining these two approaches. In order to provide an empirical study on the combination of these two methods, we set $\beta = 0$ in Eq. (5.13) and report the ranking results as a function of α, since only α balances the importance of these two individual objectives. We choose $\alpha \in \{0, 0.1, 0.2, ..., 0.9, 1\}$ and the experimental results on MovieLens100K dataset are reported in Fig. 5.3. Particularly, it only optimizes $L_{\text{pointwise}}$ when $\alpha = 0$ and only optimizes $L_{\text{row-wise}}$ when $\alpha = 1$. The results demonstrate that when we jointly learn pointwise and pairwise models (i.e., $\alpha \neq 0$ and

$\alpha \neq 1$), the ranking performance can be significantly improved, compared with solving them individually (i.e., $\alpha = 0$ or $\alpha = 1$).

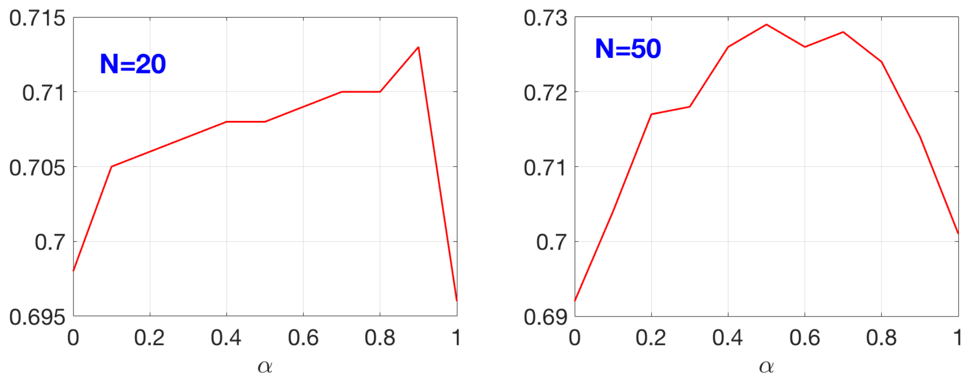


Figure 5.3: The performance of personalized ranking on MovieLens100K dataset in terms of different α when $\beta = 0$. It is evaluated using NDCG@10. N is the number of ratings used in the training per user.

(ii) Combining $L_{\text{column-wise}}$ with $L_{\text{pointwise}}$ for the task of user ranking. Symmetrically, we investigate the impact of combining $L_{\text{column-wise}}$ with $L_{\text{pointwise}}$, as individually optimizing either of these two objectives is able to provide reasonable results for the user ranking task. We set $\alpha = 0$ and tune the value of β in Eq. (5.13). Particularly, it only optimizes $L_{\text{pointwise}}$ when $\beta = 0$ and only optimizes $L_{\text{column-wise}}$ when $\beta = 1$. Experimental results on MovieLens100K dataset are reported in Fig. 5.4, which demonstrates that the combination of the pointwise and pairwise methods also significantly improves the performance of user ranking, compared with applying these two methods separately (i.e., when $\beta = 0$ or $\beta = 1$).

The reason why the combination of pointwise with pairwise objectives is able to improve the ranking performance is that the combination incorporates tie comparisons of rating scores into the pairwise model, as well as solving the identifiability issue. The concrete explanation is provided in Sec. 5.4.4.

(iii) Learning both of effective user and item latent factors by combining row-wise and column-wise comparisons.

Individually optimizing row-wise or column-wise ranking objective is only able to learn one

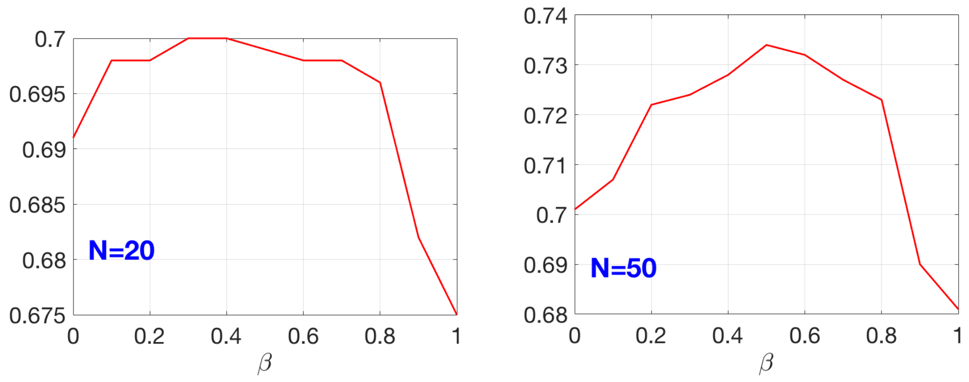


Figure 5.4: The performance of user ranking on MovieLens100K dataset in terms of different β when $\alpha = 0$. It is evaluated by NDCG@10. N is the number of ratings used in the training per item.

set of effective latent factors. In this section, we empirically study the problem that if combining row-wise and column-wise objectives can help to learn both effective user and item latent factors.

In the experiment, we choose balance factors α and β from $\{0, 0.1, 0.2, ..., 0.9, 1\}$, s.t., $\alpha+\beta \leq 1$ and conduct experiments on the MovieLens100K dataset. The experimental results of personalized ranking and user ranking by choosing different α and β combinations are reported in Fig. 5.5. In the figure, the brighter color represents the better performance.

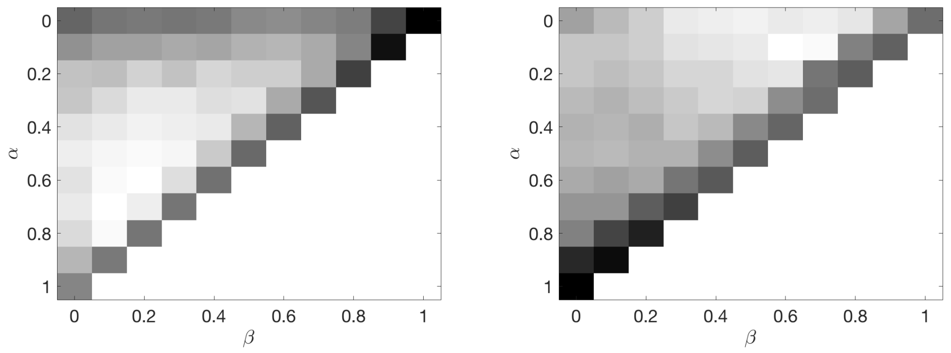


Figure 5.5: The ranking performance of personalized ranking (left panel) and user ranking (right panel) in terms of different α and β combinations ($\alpha + \beta \leq 1$). The brighter color represents the better performance. It is tested on the MovieLens100K dataset when $N = 50$.

On the left panel of Fig. 5.5, we observe that the setting of α and β which generates best personalized ranking performance is: $\alpha = 0.7$ and $\beta = 0.1$ or $\alpha = 0.6$ and $\beta = 0.2$. More specifically, when we combine $L_{\text{pointwise}}$ and $L_{\text{row-wise}}$ in optimization, the NDCG@10 score is 0.727 (see in Fig. 5.3), and after adding $L_{\text{column-wise}}$ in optimization, the ranking performance can be further improved by 2% (now 0.740). Similar results are observed when we focus on the user ranking task. Both of them tells that combing row-wise and column-wise objectives is able to improve individual row-wise or column-wise ranking task through learning both of effective user and item latent factors.

Another interesting observation from the experimental results is that the best setting of α and β combination for these two symmetric tasks is almost symmetric: the best performance of personalized ranking is obtained when we choose a relatively large α (e.g., 0.6, 0.7) and small β (e.g., 0.1), while the best performance of user ranking is obtained when we choose a relatively small α (e.g., 0.1) and large β (e.g., 0.6, 0.7). This fact may help to tune parameters when we need to solve both of these two ranking tasks.

5.6.2 Performance Comparisons

In this section, we compare our proposed method with many notable collaborative ranking algorithms on both of row-wise and column-wise ranking tasks.

Compared methods

. We use the following state-of-the-art CR methods as baselines:

- **CofiRank** [43]: It is also known as maximum margin matrix factorization which optimizes the surrogate function of NDCG error. CofiRank is always considered as a strong baseline method for collaborative ranking.

- **BPR-MF** [65]: Bayesian personalized ranking is initially proposed to model implicit feedbacks. The idea of BPR-MF is almost the same as the algorithm introduced in Sec. 5.4.2.

- **Factorization Machine (FM)** [44]: FM is known for performing well in the rating prediction task. It can mimic many rating prediction algorithms by adapting the feature representations. In this thesis, we apply the SVD++ [39] type of feature for the FM method, as SVD++ is considered as one of the most accurate rating prediction algorithms.

- **AltSVM** [89]: This method demonstrates very competitive ranking performance in the original paper if the size N of training samples is relatively large. It trains a factored form of the matrix via alternating minimization, which reduces to alternating SVM problems.

- **Decoupled Collaborative Ranking (DCR)** [86]: DCR is probably one of the best performing collaborative ranking methods, which combines ordinal regression with matrix factorization for solving the top-K ranking problem, which is defined in the rating matrix.

We compare the performance of all the methods on both of the row-wise and column-wise ranking problems and the results are reported in Table 5.2 and Table 5.3, where our proposed method is called Collaborative Multi-objective Ranking (CMR). For each method, we conduct 5 times of independent experiments and report the average and standard deviation of NDCG@10 value.

Comparisons on the performance of personalized ranking: In order to make fair comparisons, we set the dimension of latent factors as 50 for all the compared methods. In our proposed algorithm, we use the following parameter settings: $\alpha = 0.7, \beta = 0.1$, learning rate is chosen from {0.01, 0.03, 0.05, 0.1}, and regularization parameter λ is chosen from {1,10,100}. The row-wise and column-wise comparisons of rating scores are obtained from the training set, where for each user we select $N = 10, 20, 50$ rating scores. The empirical results in Table 5.2 show that our method outperforms all the baseline algorithms. Since all the compared baseline methods are learned by optimizing a single objective: CofiRank optimizes NDCG metric, BPR-MF and AltSVM optimize pairwise ranking aggregations, FM optimizes regression loss, and DCR optimizes ordinal classification loss, the superiority of our method in ranking performance demonstrates that learning multiple objectives together is an effective way to generate better performance than learning single models individually.

Comparisons on the performance of user ranking: The setting of parameters in our

Table 5.2: Performance comparisons on the row-wise personalized ranking task.

Datasets	Methods	N=10	N=20	N=50
MovieLens100K	CofiRank	0.6625 ± 0.0023	0.6933 ± 0.0018	0.7021 ± 0.0031
	BPR-MF	0.6432 ± 0.0045	0.6567 ± 0.0047	0.7101 ± 0.0032
	AltSVM	0.6295 ± 0.0075	0.6564 ± 0.0027	0.6960 ± 0.0021
	FM	0.6623 ± 0.0028	0.6680 ± 0.0028	0.6752 ± 0.0024
	DCR	0.6881 ± 0.0031	0.7018 ± 0.0044	0.7181 ± 0.0025
	CMR	$\mathbf{0.7121 \pm 0.0031}$	$\mathbf{0.7151 \pm 0.0023}$	$\mathbf{0.7341 \pm 0.0058}$
MovieLens1M	CofiRank	0.7041 ± 0.0023	0.7233 ± 0.0013	0.7256 ± 0.0042
	BPR-MF	0.6852 ± 0.0021	0.7111 ± 0.0035	0.7468 ± 0.0020
	AltSVM	0.6694 ± 0.0014	0.7154 ± 0.0032	0.7610 ± 0.0025
	FM	0.7143 ± 0.0028	0.7221 ± 0.0045	0.7245 ± 0.0024
	DCR	0.7261 ± 0.0034	0.7389 ± 0.0027	0.7601 ± 0.0033
	CMR	$\mathbf{0.7562 \pm 0.0025}$	$\mathbf{0.7558 \pm 0.0027}$	$\mathbf{0.7802 \pm 0.0013}$
Netflix1M	CofiRank	0.7077 ± 0.0025	0.7192 ± 0.0027	0.7092 ± 0.0045
	BPR-MF	0.7213 ± 0.0011	0.7224 ± 0.0015	0.7414 ± 0.0021
	AltSVM	0.6592 ± 0.0017	0.7013 ± 0.0046	0.7405 ± 0.0018
	FM	0.7190 ± 0.0027	0.7124 ± 0.0034	0.7384 ± 0.0031
	DCR	0.7332 ± 0.0037	0.7381 ± 0.0024	0.7522 ± 0.0032
	CMR	$\mathbf{0.7488 \pm 0.0014}$	$\mathbf{0.7565 \pm 0.0014}$	$\mathbf{0.7712 \pm 0.0033}$
Amazon Instant Video	CofiRank	0.7377 ± 0.0025	0.7382 ± 0.0021	0.7102 ± 0.0012
	BPR-MF	0.7151 ± 0.0008	0.7011 ± 0.0011	0.6922 ± 0.0024
	AltSVM	0.7242 ± 0.0018	0.6878 ± 0.0031	0.6841 ± 0.0020
	FM	0.7291 ± 0.0022	0.7172 ± 0.0008	0.6084 ± 0.0031
	DCR	0.7342 ± 0.0017	0.7281 ± 0.0011	0.7091 ± 0.0022
	CMR	$\mathbf{0.7511 \pm 0.0014}$	$\mathbf{0.7532 \pm 0.0027}$	$\mathbf{0.7231 \pm 0.0027}$

method for user ranking problem is similar to that in the personalized ranking task, except that we choose $\alpha = 0.1, \beta = 0.7$. The row-wise and column-wise comparisons of rating scores are obtained from the training set, where for each item we also choose $N = 10, 20, 50$ rating scores and report the results in different settings of N.

Table 5.3 reports similar results to that in the personalized ranking task: our proposed method significantly outperforms other baseline methods in all settings (i.e., $N = 10, 20, 50$). Another interesting observation is that when the same set of individual methods are applied for row-wise and column-wise ranking problems, they don't perform consistently well in different tasks. For example, DCR performs the best among all the compared methods in the personalized ranking task. However, when it is applied in the user ranking problem, some other single models outperform it, e.g., factorization machine outperforms DCR in all test datasets when $N = 10$. This observation indicates that individual row-wise or column-wise ranking algorithms may only be able to perform well in specific situations. From this point of view, combining models and objectives probably provides a reasonable way to improve the robustness of

Table 5.3: Performance comparisons on the column-wise user ranking task.

Datasets	Methods	N=10	N=20	N=50
MovieLens100K	CofiRank	0.6152 ± 0.0035	0.6691 ± 0.0027	0.6867 ± 0.0022
	BPR-MF	0.6475 ± 0.0043	0.6670 ± 0.0021	0.6935 ± 0.0015
	AltSVM	0.5632 ± 0.0023	0.6012 ± 0.0013	0.6832 ± 0.0031
	FM	0.6571 ± 0.0031	0.6802 ± 0.0017	0.7056 ± 0.0022
	DCR	0.6412 ± 0.0025	0.6745 ± 0.0021	0.7092 ± 0.0044
	CMR	**0.6778 ± 0.0023**	**0.7002 ± 0.0015**	**0.7261 ± 0.0031**
MovieLens1M	CofiRank	0.5927 ± 0.0045	0.6206 ± 0.0027	0.6427 ± 0.0035
	BPR-MF	0.6103 ± 0.0027	0.6341 ± 0.0056	0.6652 ± 0.0013
	AltSVM	0.5644 ± 0.0008	0.6136 ± 0.0023	0.6417 ± 0.0007
	FM	0.6115 ± 0.0022	0.6444 ± 0.0053	0.6736 ± 0.0041
	DCR	0.6002 ± 0.0013	0.6324 ± 0.0045	0.6722 ± 0.0031
	CMR	**0.6420 ± 0.0037**	**0.6668 ± 0.0013**	**0.6882 ± 0.0043**
Netflix1M	CofiRank	0.5924 ± 0.0013	0.6217 ± 0.0030	0.7034 ± 0.0051
	BPR-MF	0.6157 ± 0.0042	0.6355 ± 0.0028	0.6792 ± 0.0014
	AltSVM	0.6074 ± 0.0009	0.6350 ± 0.0011	0.6634 ± 0.0021
	FM	0.6122 ± 0.0021	0.6309 ± 0.0024	0.6953 ± 0.0011
	DCR	0.5720 ± 0.0016	0.6178 ± 0.0027	0.7107 ± 0.0038
	CMR	**0.6240 ± 0.0041**	**0.6692 ± 0.0025**	**0.7382 ± 0.0023**
Amazon Instant Video	CofiRank	0.7451 ± 0.0018	0.7571 ± 0.0022	0.7623 ± 0.0035
	BPR-MF	0.7521 ± 0.0023	0.7581 ± 0.0011	0.7610 ± 0.0015
	AltSVM	0.7292 ± 0.0017	0.7331 ± 0.0021	0.7399 ± 0.0028
	FM	0.7538 ± 0.0027	0.7610 ± 0.0014	0.7743 ± 0.0031
	DCR	0.7467 ± 0.0021	0.7581 ± 0.0011	0.7731 ± 0.0015
	CMR	**0.7610 ± 0.0013**	**0.7720 ± 0.0024**	**0.7841 ± 0.0033**

predictions, as we can see that our method consistently performs the best in all situations.

5.7 Conclusion

In this chapter, we propose a method which jointly resolves row-wise and column-wise ranking problems, in order to learn both of effective user and item latent factors, since individually solving row-wise or column-wise ranking task is only able to learn one set of effective latent factors. Our approach simultaneously optimizes three objectives through a parameter sharing framework: one aims to accurately predict rating scores, one aims to learn the correct row-wise order of rating scores and the other one aims to learn the correct column-wise order of rating scores. In addition, we propose a simple yet effective stochastic gradient descent method to deal with heterogeneous data inputs. Through the comprehensive empirical study, we demonstrate the effectiveness of our method over other collaborative ranking approaches.

Chapter 6

Summary and Future Work

In this thesis, we discuss collaborative ranking-based recommender systems. In previous chapters, we introduced our contributions to improving the ranking performance of recommender systems.

In Chapter 2, we discussed the ordinal view of user rating scores, which more accurately reflect users' internal preferences. In order to capture the ordinal nature of rating scores, we proposed a pointwise method that transfers cumulative probability distributions of rating scores to ranking scores. In addition, in order to improve the recommendation performance at the top of a ranked list, our proposed method adopts a weighted summation of the cumulative probabilities where higher rating scores are emphasized more than lower rating scores. Through extensive experimental evaluations, our proposed method is verified to significantly outperform state-of-the-art methods.

In chapter 3, we also discussed the ordinality of user ratings. Different from previous statistic ordinal regression methods which consider ordered rating scores as ordinal categorical labels, in Chapter 3 we directly calculate the magnitudes of user ratings in the ordinal scale, which quantitatively evaluate individual users' internal preferences. Through extensive experiments on popular datasets, we demonstrated that our proposed ordinal regression method is able to significantly improve the rating prediction accuracy over common collaborative filtering methods.

In Chapter 4 and Chapter 5, we discussed three issues existing in current pairwise collaborative ranking methods: (1) identifiability issues; (2) the pairwise ranking objective function has no minimizer; (3) current pairwise collaborative ranking methods fail to learn effective user latent factors. In order to address the above issues, we proposed to combine pointwise and pairwise ranking objectives together. We also proposed to combine row-wise and column-wise ranking problems. The idea of the ensemble method is a simple, quick, and effective mechanism to address all the aforementioned issues.

Even through collaborative ranking-based recommender systems have been investigated and improved extensively over the past years, there is still room for substantial improvement. In the following, we propose several interesting problems which deserve more research:

- *Understanding the relationship between data sparsity with ranking performance.* In the comparative study of collaborative filtering [115], the level of data sparsity significantly affects the performance of different rating prediction algorithms. In this thesis, we also observed that the ranking performance of various methods differs substantially based on the choice of different data sparsity levels. Specifically, we observed that pointwise methods perform relatively well when data are highly sparse, while pairwise methods perform better when data become less sparse. In order to investigate this kind of knowledge, we need a more comprehensively comparative study of collaborative ranking, (probably) including comparing the traditional collaborative filtering methods, ranking-based methods, neural collaborative filtering/ranking methods, etc. What is more, it could be very interesting if we are able to understand why certain kind of methods perform the best in a specific range of data sparsity. This knowledge is very helpful when we decide to choose an algorithm for a specific recommendation scenario.

- *More understanding on different kinds of collaborative filtering/ranking models.* While the comparative experimental study is very valuable for the real applications, we might still need a better understanding of models and their relationships. For example, the multilayer perceptron (MLP) rating prediction model can be mimicked by matrix factorization [46], which builds a simple connection between the neural network with matrix factorization. However, there are still many other collaborative filtering models, such as autoencoder, graph-based factorization methods, restricted Boltzmann machine, etc. It could be very interesting if we are able to understand the relationships between them, which may result in a proposal of a unified model or learning framework, such as factorization machine.

- *Fast Recommendation.* In this thesis, we mainly discussed the offline ranking performance of recommender systems. While in the real applications, it is vital to provide a fast real-time recommendation of items, which are selected from a very large dataset. To that end, a common approach is to learn a binary representation of user and item latent factors through discrete matrix factorization. However, discrete optimization on matrix factorization problem is relatively time-consuming and does not scale well. In addition, the matrix factorization approach is not able to handle the cold-start problem (e.g., meet a new user) well, compared with neighbor-hood based methods. It could be interesting to develop an effective unsupervised hashing schema for the fast and accurate recommendation, which are suitable for different recommendation algorithms.

References

[1] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, “Using collaborative filtering to weave an information tapestry,” *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[2] S. S. Anand and B. Mobasher, “Intelligent techniques for web personalization,” in *Proceedings of the 2003 international conference on Intelligent Techniques for Web Personalization*. Springer-Verlag, 2003, pp. 1–36.

[3] T. Mahmood and F. Ricci, “Improving recommender systems with adaptive conversational strategies,” in *Proceedings of the 20th ACM conference on Hypertext and hypermedia*. ACM, 2009, pp. 73–82.

[4] F. McSherry and I. Mironov, “Differentially private recommender systems: Building privacy into the netflix prize contenders,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 627–636.

[5] P. Resnick and H. R. Varian, “Recommender systems,” *Communications of the ACM*, vol. 40, no. 3, pp. 56–58, 1997.

[6] R. Burke, “Hybrid web recommender systems,” in *The adaptive web*. Springer, 2007, pp. 377–408.

[7] F. Ricci, L. Rokach, and B. Shapira, “Recommender systems: introduction and challenges,” in *Recommender systems handbook*. Springer, 2015, pp. 1–34.

[8] J. B. Schafer, J. Konstan, and J. Riedl, “Recommender systems in e-commerce,” in *Proceedings of the 1st ACM conference on Electronic commerce*. ACM, 1999, pp. 158–166.

[9] T. Lee, J. Chun, J. Shim, and S.-g. Lee, “An ontology-based product recommender system for b2b marketplaces,” *International Journal of Electronic Commerce*, vol. 11, no. 2, pp. 125–155, 2006.

[10] J. L. de la Rosa, N. Hormazábal, S. Aciar, G. A. Lopardo, A. Trias, and M. Montaner, “A negotiation-style recommender based on computational ecology in open negotiation environments,” *IEEE Transactions on Industrial Electronics*, vol. 58, no. 6, pp. 2073–2085, 2011.

[11] K. Sugiyama, K. Hatano, and M. Yoshikawa, “Adaptive web search based on user profile constructed without any effort from users,” in *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004, pp. 675–684.

[12] O. C. Santos, *Educational Recommender Systems and Technologies: Practices and Challenges*. IGI Global, 2011.

[13] O. R. Zaíane, “Building a recommender agent for e-learning systems,” in *Computers in education, 2002. proceedings. international conference on*. IEEE, 2002, pp. 55–59.

[14] C.-M. Chen, L.-J. Duh, and C.-Y. Liu, “A personalized courseware recommendation system based on fuzzy item response theory,” in *null*. IEEE, 2004, pp. 305–308.

[15] L. Terán and A. Meier, “A fuzzy recommender system for eelections,” in *International Conference on Electronic Government and the Information Systems Perspective*. Springer, 2010, pp. 62–76.

[16] X. Guo and J. Lu, “Intelligent e-government services with personalized recommendation techniques,” *International Journal of Intelligent Systems*, vol. 22, no. 5, pp. 401–417, 2007.

[17] P. De Meo, G. Quattrone, and D. Ursino, “A decision support system for designing new services tailored to citizen profiles in a complex and distributed e-government scenario,” *Data & Knowledge Engineering*, vol. 67, no. 1, pp. 161–184, 2008.

[18] R. D. Burke, K. J. Hammond, and B. C. Young, “Knowledge-based navigation of complex information spaces,” in *Proceedings of the national conference on artificial intelligence*, vol. 462, 1996, p. 468.

[19] H.-W. Tung and V.-W. Soo, “A personalized restaurant recommender agent for mobile e-service,” in *e-Technology, e-Commerce and e-Service, 2004. EEE’04. 2004 IEEE International Conference on*. IEEE, 2004, pp. 259–262.

[20] A. García-Crespo, J. Chamizo, I. Rivera, M. Mencke, R. Colomo-Palacios, and J. M. Gómez-Berbís, “Speta: Social pervasive e-tourism advisor,” *Telematics and informatics*, vol. 26, no. 3, pp. 306–315, 2009.

[21] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, “Collaborative filtering recommender systems,” in *The adaptive web*. Springer, 2007, pp. 291–324.

[22] M. J. Pazzani and D. Billsus, “Content-based recommendation systems,” in *The adaptive web*. Springer, 2007, pp. 325–341.

[23] P. Lops, M. De Gemmis, and G. Semeraro, “Content-based recommender systems: State of the art and trends,” in *Recommender systems handbook*. Springer, 2011, pp. 73–105.

[24] S. Trewin, “Knowledge-based recommender systems,” *Encyclopedia of library and information science*, 2000.

[25] J. He and W. W. Chu, “A social network-based recommender system (snrs),” in *Data mining for social network data*. Springer, 2010, pp. 47–74.

[26] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, “Recommender systems with social regularization,” in *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 2011, pp. 287–296.

[27] Z. Zhang, H. Lin, K. Liu, D. Wu, G. Zhang, and J. Lu, “A hybrid fuzzy-based personalized recommender system for telecom products/services,” *Information Sciences*, vol. 235, pp. 117–129, 2013.

[28] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in *Recommender systems handbook*. Springer, 2011, pp. 217–253.

[29] K. Verbert, N. Manouselis, X. Ochoa, M. Wolpers, H. Drachsler, I. Bosnic, and E. Duval, "Context-aware recommender systems for learning: a survey and future challenges," *IEEE Transactions on Learning Technologies*, vol. 5, no. 4, pp. 318–335, 2012.

[30] J. Masthoff, "Group recommender systems: Combining individual models," in *Recommender systems handbook*. Springer, 2011, pp. 677–702.

[31] H. Yin, B. Cui, J. Li, J. Yao, and C. Chen, "Challenging the long tail recommendation," *Proceedings of the VLDB Endowment*, vol. 5, no. 9, pp. 896–907, 2012.

[32] A. Elberse, "Should you invest in the long tail?" *Harvard business review*, vol. 86, no. 7/8, p. 88, 2008.

[33] R. Armstrong, "The long tail: Why the future of business is selling less of more," *Canadian Journal of Communication*, vol. 33, no. 1, 2008.

[34] D. M. Fleder and K. Hosanagar, "Recommender systems and their impact on sales diversity," in *Proceedings of the 8th ACM conference on Electronic commerce*. ACM, 2007, pp. 192–199.

[35] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, J. Riedl *et al.*, "Combining collaborative filtering with personal agents for better recommendations," in *AAAI/IAAI*, 1999, pp. 439–446.

[36] 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)*, vol. 47, no. 1, p. 3, 2014.

[37] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1998, pp. 43–52.

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

[39] 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*. ACM, 2008, pp. 426–434.

[40] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *Proceedings of KDD cup and workshop*, vol. 2007, 2007, pp. 5–8.

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

[42] 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*. International World Wide Web Conferences Steering Committee, 2014, pp. 85–96.

[43] 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*, 2007, pp. 1593–1600. [Online]. Available: http://papers.nips.cc/paper/3359-cofi-rank-maximum-margin-matrix-factorization-for-collaborative-ranking

[44] S. Rendle, "Factorization machines with libfm," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 3, no. 3, p. 57, 2012.

[45] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 2016, pp. 7–10.

[46] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

[47] X. Wang, X. He, L. Nie, and T.-S. Chua, "Item silk road: Recommending items from information domains to social users," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 185–194.

[48] F. Strub, R. Gaudel, and J. Mary, "Hybrid recommender system based on autoencoders," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 2016, pp. 11–16.

[49] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1235–1244.

[50] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 2016, pp. 153–162.

[51] H. Wang, X. Shi, and D.-Y. Yeung, "Relational stacked denoising autoencoder for tag recommendation." in *AAAI*, 2015, pp. 3052–3058.

[52] H. Wang and D.-Y. Yeung, "Towards bayesian deep learning: A framework and some existing methods," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 12, pp. 3395–3408, 2016.

[53] J. Bennett, S. Lanning *et al.*, "The netflix prize," in *Proceedings of KDD cup and workshop*, vol. 2007. New York, NY, USA, 2007, p. 35.

[54] K. Järvelin and J. Kekäläinen, "IR evaluation methods for retrieving highly relevant documents." in *SIGIR*, Athens, Greece, 2000, pp. 41–48.

[55] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan, "Expected reciprocal rank for graded relevance," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 621–630.

[56] S. Balakrishnan and S. Chopra, "Collaborative ranking," in *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 2012, pp. 143–152.

[57] 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*. ACM, 2011, pp. 117–124.

[58] 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*, 2015, pp. 1907–1916. [Online]. Available: http://jmlr.org/proceedings/papers/v37/park15.html

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

[60] 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*, 2007, pp. 897–904.

[61] 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*, vol. 17, no. 6, pp. 734–749, 2005.

[62] 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*. ACM, 2009, pp. 125–132.

[63] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

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

[65] 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*. AUAI Press, 2009, pp. 452–461.

[66] 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*. ACM, 2009, pp. 759–766.

[67] A. Agresti, *Categorical Data Analysis*, 2nd ed. Hoboken, NJ: John Wiley & Sons, Inc., 2002.

[68] 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*, 2009, pp. 315–323.

[69] D. Cossock and T. Zhang, "Statistical analysis of bayes optimal subset ranking," *Information Theory, IEEE Transactions on*, vol. 54, no. 11, pp. 5140–5154, 2008.

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

[71] C. C. Aggarwal, *Recommender Systems*. Springer, 2016.

[72] F. Ricci, L. Rokach, and B. Shapira, *Introduction to recommender systems handbook*. Springer, 2011.

[73] P. McCullagh, "Regression models for ordinal data," *Journal of the royal statistical society. Series B (Methodological)*, pp. 109–142, 1980.

[74] A. Agresti, "Introduction to generalized linear models," *Categorical Data Analysis, Second Edition*, pp. 115–164, 2002.

[75] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: an open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, 1994, pp. 175–186.

[76] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "Grouplens: applying collaborative filtering to usenet news," *Communications of the ACM*, vol. 40, no. 3, pp. 77–87, 1997.

[77] M. D. Ekstrand, J. T. Riedl, J. A. Konstan *et al.*, "Collaborative filtering recommender systems," *Foundations and Trends® in Human–Computer Interaction*, vol. 4, no. 2, pp. 81–173, 2011.

[78] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 285–295.

[79] A. Mnih and R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in neural information processing systems*, 2007, pp. 1257–1264.

[80] S. Rendle, "Factorization machines," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, 2010, pp. 995–1000.

[81] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, 2008, pp. 263–272.

[82] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 4, no. 1, p. 1, 2010.

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

[84] Y. Koren and J. Sill, "Collaborative filtering on ordinal user feedback," in *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, 2013, pp. 3022–3026.

[85] U. Paquet, B. Thomson, and O. Winther, "A hierarchical model for ordinal matrix factorization," *Statistics and Computing*, vol. 22, no. 4, pp. 945–957, 2012.

[86] J. Hu and P. Li, "Decoupled collaborative ranking," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 1321–1329.

[87] 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*, 2011, pp. 693–701.

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

[89] 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.

[90] A. Agresti and M. Kateri, *Categorical data analysis*. Springer, 2011.

[91] T. Joachims, "Optimizing search engines using clickthrough data," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002, pp. 133–142.

[92] C. J. Burges, "From ranknet to lambdarank to lambdamart: An overview," *Learning*, vol. 11, pp. 23–581, 2010.

[93] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 89–96.

[94] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *Journal of machine learning research*, vol. 4, no. Nov, pp. 933–969, 2003.

[95] M. N. Volkovs and R. S. Zemel, "Boltzrank: learning to maximize expected ranking gain," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 1089–1096.

[96] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.

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

[98] D. Cossock and T. Zhang, "Subset ranking using regression," in *Learning theory*. Springer, 2006, pp. 605–619.

[99] 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*, 2015, pp. 1907–1916.

[100] Y. Shi, M. Larson, and A. Hanjalic, "List-wise learning to rank with matrix factorization for collaborative filtering," in *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010, pp. 269–272.

[101] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic, "Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering," in *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 2012, pp. 139–146.

[102] J. Hu and P. Li, "Collaborative filtering via additive ordinal regression," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 2018, pp. 243–251.

[103] J. Zhang and P. Pu, "A recursive prediction algorithm for collaborative filtering recommender systems," in *Proceedings of the 2007 ACM conference on Recommender systems*. ACM, 2007, pp. 57–64.

[104] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, no. 1, pp. 76–80, 2003.

[105] F. Zhuang, D. Luo, N. J. Yuan, X. Xie, and Q. He, "Representation learning with pairwise constraints for collaborative ranking," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 2017, pp. 567–575.

[106] W. Pan and L. Chen, "Cofiset: Collaborative filtering via learning pairwise preferences over item-sets," in *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 2013, pp. 180–188.

[107] S. Rendle and C. Freudenthaler, "Improving pairwise learning for item recommendation from implicit feedback," in *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, 2014, pp. 273–282.

[108] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, and A. Hanjalic, "xclimf: optimizing expected reciprocal rank for data with multiple levels of relevance," in *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 2013, pp. 431–434.

[109] S. Huang, S. Wang, T.-Y. Liu, J. Ma, Z. Chen, and J. Veijalainen, "Listwise collaborative filtering," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2015, pp. 343–352.

[110] Y. Lei, W. Li, Z. Lu, and M. Zhao, "Alternating pointwise-pairwise learning for personalized item ranking," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 2155–2158.

[111] D. Sculley, "Combined regression and ranking," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 979–988.

[112] A. Agresti, *Categorical data analysis*. John Wiley & Sons, 2013.

[113] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5, no. 4, p. 19, 2016.

[114] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *proceedings of the 25th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 2016, pp. 507–517.

[115] J. Lee, M. Sun, and G. Lebanon, "A comparative study of collaborative filtering algorithms," *arXiv preprint arXiv:1205.3193*, 2012.