LEARNING TOP-N RECOMMENDER SYSTEMS WITH IMPLICIT FEEDBACKS

A Dissertation
Submitted to
the Temple University Graduate Board

In Partial Fulfillment
of the Requirements for the Degree
DOCTOR OF PHILOSOPHY

---

by
Feipeng Zhao
May 2017

Examining Committee Members:

Yuhong Guo, Advisory Chair, Dept. of Computer and Information Sciences
Justin Shi, Dept. of Computer and Information Sciences
Eduard Dragut, Dept. of Computer and Information Sciences
Yuexiao Dong, External Member, Dept. of Statistical Science, Temple University

ABSTRACT

Top-N recommender systems automatically recommend N items for users from huge amounts of products. Personalized Top-N recommender systems have great impact on many real world applications such as E-commerce platforms and social networks. Sometimes there is no rating information in user-item feedback matrix but only implicit purchase or browsing history, that means the user-item feedback matrix is a binary matrix, we call such feedbacks as implicit feedbacks. In our work we try to learn Top-N recommender systems with implicit feedbacks. First, we design a heterogeneous loss function to learn the model. Second, we incorporate item side information into recommender systems. We formulate a low-rank constraint minimization problem and give a closed-form solution for it. Third, we also use item side information to learn recommender systems and in this model we use gradient descent method to learn our model.

Most existing methods produce personalized top-N recommendations by minimizing a specific uniform loss such as pairwise ranking loss or pointwise recovery loss. In our first model, we propose a novel personalized Top-N recommendation approach that minimizes a combined heterogeneous loss based on linear self-recovery models. The heterogeneous loss integrates the strengths of both pairwise ranking loss and pointwise recovery loss to provide more informative recommendation predictions. We formulate the learning problem with heterogeneous loss as a constrained convex minimization problem and develop a projected stochastic gradient descent optimization algorithm to solve it.

Most previous systems are only based on the user-item feedback matrix. In many applications, in addition to the user-item rating/purchase matrix, item-based side information such as product reviews, book reviews, item comments, and movie plots can be easily collected from the Internet. This abundant item-based information can be used for recommendation systems. In the second model, we propose a novel predictive collaborative filtering approach that exploits both the partially observed user-item recommendation matrix and the item-based side information to produce top-N recommender systems. The proposed approach automatically identifies the most interesting items for each user from his or her non-recommended item pool by aggregating over his or her recommended items via a low-rank coefficient matrix. Moreover, it also simultaneously builds linear regression models from the item-based side information such as item reviews to predict the item recommendation scores for the users. The proposed approach is formulated as a rank constrained joint minimization problem with integrated least squares losses, for which an efficient analytical solution can be derived.

In the third model, we also propose a joint discriminative prediction model that exploits both the partially observed user-item recommendation matrix and the item-based side information to build top-N recommender systems. This joint model aggregates observed user-item recommendation activities to predict the missing/new user-item recommendation scores while simultaneously training a linear regression model to predict the user-item recommendation scores from auxiliary item features. We evaluate the proposed approach on a variety of recommendation tasks. The experimental results show that the proposed joint model is very effective for producing top-N recommendation systems.

To my parents.

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Dr. Yuhong Guo. She introduced me to the field of Machine Learning. Her attitude to research really inspired me a lot. She liked to discuss with me and help me improve myself. Many times Dr.Guo helped me revised papers and gave me lots of suggestions. I feel that I owe her a lot. Without her help, I cannot finish this book.

Second, I would like to thank my other committee members, Dr. Justin Shi, Dr. Eduard Dragut for giving me very useful feedback towards my work. I am also grateful to Dr. Yuexiao Dong for being my external committee member.

I would also like to thank my lab-mates, Min Xiao, Xin Li. Whenever I met any difficulties in the research, they are always beside me and ready to help. I really learned a lot from them. They gave me a lot of suggestions. They also helped me avoid mistakes.

Other friends such as Chen Shen, Meng Ye, Dawei Li, Yu Pang and Liang Du also gave me a lot of help. Discussing with them always help me solve the problems.

Finally, I would like to thank my parents. Ph.D. career is difficult, They always encourage me not to give up.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Top-N Recommender Systems

Recommender Systems are information filtering systems that seek to predict the "rating" or "preference" that users would assign to an item. In the era of Internet, online shopping has become a close part of peoples daily life. To facilitate consumers' online shopping experience, help consumers to find their interested items from the huge amount of online items and hence encourage online item purchases, effective Top-N commercial recommendation systems, which suggest a small list of items that best match each consumers taste from the large amount of product items, have become increasingly important. Top-N recommendation systems automatically predict the recommendation scores over each item in the whole product item pool for each consumer, and recommend top-N items with high scores to the consumers. They can be widely used in many real world applications. Recommender systems can be applied to recommend different kinds of things. Such as music [89, 29, 30, 80], movies [23, 17, 19], news [37, 18, 42, 36], products [40, 45], articles [84], blogs [9, 88]. Furthermore, people also try to recommend personalized social links [92, 13, 47, 96] and point-of-interest [41, 93, 35, 39, 22]. Amazon.com has a huge amount of online products, while consumers may not know all the special items they would like. A Top-N recommendation system, which can effectively suggest a set of personalized

Recommender Systems Application	Related Company
Movie Recommendation	Netflix Hulu
Music Recommendation	Apple Spotify Yahoo!
Products	Amazon Ebay
News	Google Yahoo!
Articles	Google Springer
Blogs	Tumblr Twitter
Social Links	Facebook LinkedIn Twitter
Points-of-Interests	Yelp FourSquare

Table 1.1: Recommender Systems Applications and Related Companies

selected products for each consumer, can significantly increase the user-item purchase probability. Similarly, Netflix Inc. has thousands of online movies and TV shows, a personalized recommendation for users can help them to find the videos they may like. Yahoo! and Apple Itunes have millions of musics, Top-N recommendation systems can help users to find the musics they may like and provide a recommendation list to each user. In summary, an effective Top-N personalized recommendation system can have significant real world commercial impacts. Google and Yahoo! can recommend personalized news to the users based on their click history. Yelp can help users to find their favorite restaurants, hotels and so on. In Table 1.1 we give a summary of different recommender system applications and related companies.

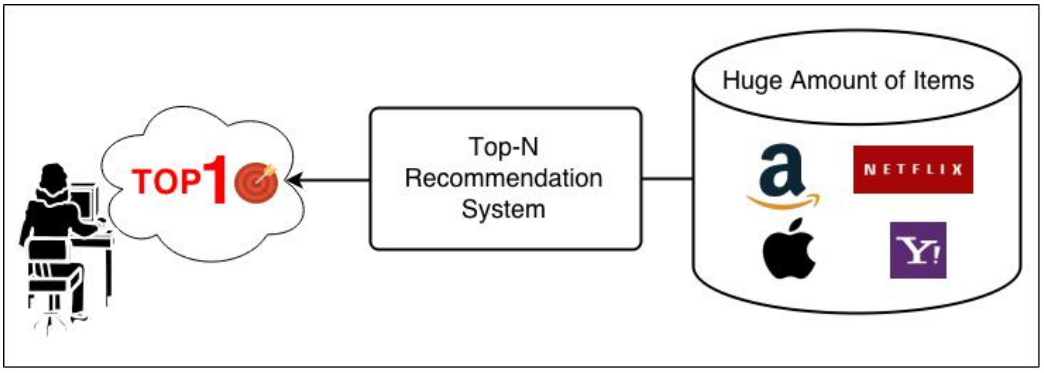

FIGURE 1.1: Top-N Recommender Systems

1.2 Top-N Recommender Systems with Implicit Feedbacks

1.2.1 Explicit Feedbacks vs Implicit Feedbacks

Traditional Recommender Systems rely on explicit feedbacks, that means the input is a sparse rating matrix $M \in R^{n \times d}$, where n is user numbers and d is item number. For example, in Netflix challenge [6], Netflix collects ratings(from 1 to 5) for movies of each user. Yelp provides ratings from users to restaurants. However, sometimes explicit feedbacks are not available. For example, when we purchase products from Amazon, sometimes we wont give a rating for them. When we browser from Internet, we only leave the click history without any explicit feedbacks. In such situations, the user-item feedback matrix is a binary matrix with values 0 and 1. We call such feedbacks implicit feedbacks. In our works, we try to build Top-N recommender systems based on implicit Feedbacks.

In figure 1.2, we show the explicit and implicit feedbacks. In the figure, each row u represents one user and each column I represents one item. We can see that in explicit feedbacks, the entries have values from 1 to 5 while in implicit feedbacks, the entries only have values 0 and 1.

1.3 Importance of Top-N Recommender Systems

Nowadays, with the information explosion, it is necessary to filter the huge amount of information. Recommender systems are very popular in recent years. In this chapter

Explicit Feedbacks

	I_1	I_2	I_3	I_4	I_5
u_1	5	1	0	0	5
u_2	3	0	4	0	0
u_3	2	2	3	0	0
u_4	5	4	0	3	2

Implicit Feedbacks Matrix

	I_1	I_2	I_3	I_4	I_5
u_1	1	1	0	0	1
u_2	1	0	1	0	0
u_3	1	1	1	0	0
u_4	1	1	0	1	1

FIGURE 1.2: Implicit and Explicit Feedbacks

we briefly explain why recommender systems are very important for companies and users.

1.3.1 Importance of Top-N Recommender Systems for Companies

Apple sells billions of songs every year by using iTunes. There are millions of items in Amazon. There are huge amounts of news every day from Yahoo!. People upload lots of videos to Youtube. If there is no personalized recommender systems, those companies can not push items efficiently to users. With the help of recommender systems, users can get personalized recommendations easily from huge amount items, people may buy more items, click more news links or videos. This will help companies get more revenue.

1.3.2 Importance of Top-N Recommender Systems for Users

Top-N recommender systems also bring values to users. Think about such a situation, when you want to explore new musics you never heard before from iTunes, it's hard

for you to get musics of your taste easily. One Top-N recommender system can recommend you new musics based on your history and other users history. It will save you a lot of time and you can find good musics recommender systems learned for you. Similarly, when you want to go to a restaurant when you travel, you are not familiar with the new place and it's very difficult for you to find a good restaurant. Recommender systems can learn your taste and recommend restaurants you may like. It really makes your life easier.

1.4 Design Top-N Recommender Systems

Since Top-N recommender systems is very important, in this book, we try to design novel Top-N recommender systems. Different recommender systems algorithms are designed for special usages. In our recommender systems, we try to make the recommendation results better than state-of-the-art models. Particularly, we will design new loss functions to learn a better Top-N recommender system. In addition, besides user-item feedback matrix(Figure 1.2), we also try to use item side information to improve the performance of our model.

1.5 Organization

The remainder of this work is organized as follows. In the following chapter, we give a brief review of existing methods for Top-N Recommender Systems. After that, we present top-N Recommendation with Heterogeneous Loss in Chapter 3. In this chapter, a new loss function is designed and the stochastic gradient descent algorithm is applied to optimize . We then show the work Predictive Collaborative Filtering with Side Information in Chapter 4 and Chaper 5. Finally we conclude the work and discuss the future work in Chapter 6.

CHAPTER 2

BACKGROUND

A variety of works have been developed in the literature to address recommender systems. We provide a brief review below on the most related collaborative filtering approaches and side information assisted approaches. Collaborative filtering approaches make recommendations by using both the target users' previous recommendation activities and other users' history behaviors [79], which include two groups of methods, neighborhood-based methods [66, 20, 28] and latent factor-based methods [7, 25, 62, 64, 86, 32, 17, 75].

2.1 Neighborhood-based Collaborative Filtering

Implicit Feedbacks Matrix

	I_1	I_2	I_3	I_4	I_5
u_1	1	1	0	0	1
u_2	1	0	1	0	0
u_3	1	1	1	0	0
u_4	1	1	0	1	1

FIGURE 2.1: Implicit Feedbacks. Each column is treated as item feature to calculate item-item similarity

Neighborhood-based collaborative filtering methods make recommendations by exploiting similarities among items within the user-item recommendation matrix. As showed in Figure 2.1, In the neighborhood-based Collaborative Filtering models, each item column is treated as one instance vector, each user is one feature dimension. Once item-item similarities are computed, the prediction on a new item for a user is obtained by considering the particular users past recommendation activities on similar items. For example, the work in [66] proposed to use similarity weighted sum prediction to recommend new items for a target user. In order to measure the item-item similarities, various similarity schemes have been investigated, including cosine similarity, correlation similarity, adjusted cosine similarity [66], and conditional probability similarity [20]. Suppose we have a binary feedback matrix $Y \in \{0,1\}^{n \times d}$, d items, we use $W \in \mathbb{R}^{d \times d}$ to represent the similarity matrix. Take the implicit feedback matrix in Figure 2.1 as example, the similarity between item 1 and item 2 W_{12} can be calculated as:

$$W_{12} = \frac{Y_{:,1}^T Y_{:,2}}{|Y_{:,1}||Y_{:,2}|} = (1,1,1,1) \times (1,0,1,1)/(2 * \sqrt{3}) = 0.866 \tag{2.1}$$

2.2 Latent Factor-based Methods

Unlike the neighborhood-based methods, which directly employ the user-item recommendation matrix for prediction, the latent factor-based collaborative filtering methods assume that the true feedback matrix $\hat{Y}$ has a low rank k. They first identify a set of latent factors for users $U \in \mathbb{R}^{n \times k}$ and items $V \in \mathbb{R}^{d \times k}$ respectively. They then train a prediction model on the latent space to make recommendations. In order to identify the latent factors, a variety of models have been employed.

For example, the pure singular value decomposition (**PureSVD**) method [17] use SVD to factorize the feedback matrix. After they get U, V from the factorization, they use top-K component to reconstruct the matrix and get a low rank matrix $\tilde{Y}$.

$$\tilde{Y} = U_k * V_k^T \tag{2.2}$$

Similarly, in [25], LSA [7] is used to decompose the feedback matrix. In [62, 86] the maximum margin matrix factorization methods are used to learn the user and item latent features.

2.3 Other Methods

In addition to these two groups, some other collaborative filtering recommender systems have also been developed in the literature [31, 50]. Koren [31] proposed an asymmetricSVD based hybrid approach to address recommendation problems, which combines the similarity-based neighborhood method and the latent factor-based method together for recommendations. Ning and Karypis [50] proposed a sparse linear method to address Top-N recommendation problems, which computes the recommendation score on a non-purchased/non-recommended item for a target user using sparse aggregation of purchased/recommended items of the particular user. Shi et al. [73] proposed a collaborative less-is-more filtering approach, which maximizes the mean reciprocal rank to address recommendation problems in social networks.

In addition to the traditional matrix factorization methods, Neural Network models [34, 5] have also been used for collaborative filtering. For example, in [64, 81], the authors use a restricted Boltzmann Machine(RBM) to build collaborative filtering models. In [70], the authors use an autoencoder framework to learn a collaborative filtering model. In [85], the authors use the Bayesian stacked denoising autoencoder and deep learning model to learn the side information and rating matrix together for collaborative filtering. In [38], the authors use marginalized denoising autoencoder(mDAE) [12] to learn side information and then incorporate the learned representation into matrix factorization model to learn better user and item representations. In [52], convolutional neural networks(CNN) [33] is used on music content to learn new representations. In [16], deep learning is used to recommend youtube contents to users.

2.4 Analysis Top-N Recommender Systems based on Loss Function

Existing top-N recommendation methods have mostly focused on minimizing a uniform type of recommendation recovery loss to predict the recommendation scores. Based on the type of the losses, these recommendation systems can be categories into two main groups. The first group exploits point-wise comparison losses and the second group exploits pairwise ranking losses. Pointiwise comparison methods. Many traditional Top-N recommendation algorithms perform learning by minimizing the point-wise (entry-wise) divergence of the reconstructed recommendation matrix from the original observation matrix. These include the collaborative filtering methods [68, 79, 26, 86, 94], and the sparse linear aggregation methods [50, 14, 28, 15]. For example, a weighted regularized matrix factorization (WRMF) model has been developed in [26], which minimizes the point-wise difference between the reconstructed matrix produced by the inner product of the user and item latent representations and the training matrix. Similarly, the work in [94] enforces the pointwise similarity between the training matrix and reconstructed matrix. It uses a non-locking parallel computing algorithm to perform matrix completion. In [50], a sparse linear method (SLIM) was proposed to recover the missing recommendation entries. It exploits the item-item correlations to reconstruct the incomplete recommendation matrix and minimizes the pointwise reconstruction loss between the reconstructed matrix and the input training matrix. Pairwise ranking methods. Given a pair of items, a user naturally will prefer one to another, which forms the important pairwise preference structure of the recommendation data. The pointwise comparison methods however ignored this user personalized preference information. Recently, pairwise ranking methods are proposed to produce top-N recommendation systems by optimizing the preference structure consistency between the original matrix and the reconstructed recommendation matrix [86, 77, 11, 3, 56, 60]. Pairwise ranking methods treat training data as a set of triplet instances; for example, the triplet (i, j, k) is an instance

that encodes the i-th users preference to item j over item k. Different pairwise ranking losses have been exploited in these works. For example, the pairwise ranking methods in [60, 3] optimize AUC scores; the work in [86] optimizes a normalized discounted cumulative gain; Yun et al. [94] explored the connection between the metric discounted cumulative gain and the binary classification to change the ranking problem into binary classification problems; Park et al. [56] proposed a large-scale collaborative ranking method that exploits a max-margin hinge ranking loss to minimize the ranking risk in the reconstructed recommendation matrix. Nevertheless, these pairwise ranking methods ignored the entrywise consistency between the reconstructed matrix and the original matrix.

2.5 Recommender Systems with Side Information

Besides user-item feedback matrix. Other information can be collected easily from Internet. As showed in Table 2.1, user age, education, salary et.al are useful information to improve the performance of recommender systems. Similarly, item side information such as item price, description, category, reviews and so on can also be incorporated into recommender systems. In fact lots of works use user/item side information when learning recommender systems. In [51], they use item reviews as item side information to learn a Top-N recommender system. In [74], Shin et.al use tumblr tags and text as side information to recommend blogs to users. In [38], Denoising Autoencoder(DAE) [83] is used to extract item side information as better representations to improve the recommender systems performance.

2.6 Recommender Systems with other information

Besides side information, other information are also used in recommender systems. For example, in [44, 91], social networks are exploited to enhance the performance of recommender systems. In [1, 72, 101, 100], context-aware recommender systems are learned. When they recommend items to users, real-time context such as weather,

Table 2.1: User and Item Side Information Example

User Side Information	Item Side Information
Age	Price
Education	Description
Hobbies	Category
Salary	Reviews
Gender	...
...	...

location, time and so on are considered to help the recommender systems to decide what items to be introduced to people.

2.7 Challenge of Top-N Recommender Systems

Recommender Systems has always been a hot topic. People try to use Recommender Systems in different situations and always meet challenges. In our work, we mainly try to solve two challenges: the first one is designing a loss function to improve the performance of top-N recommender systems. The second one is that we try to use a better way to incorporate side information into recommender systems to improve the performance.

2.7.1 Loss function design

As discussed before,there are two kinds of different loss functions to learn the recovery matrix: point-wise consistency and pair-wise consistency. Different kinds of loss functions are designed to learn recommender systems. In [73, 72, 71, 3, 56], different loss functions are designed to optimize special metrics. How to design a good loss function is very important to the performance of recommender systems.

2.7.2 Side Information Incorporation

User/item side information are used in recommender systems. In [51, 74], algorithms are proposed. In [38], Li et.al try to use deep learning methods to improve the side information representation. However, these methods improve the performance

slightly. In order to make the performance better,more smart algorithms still need to be proposed.

2.7.3 Large Scale Recommender Systems

In real-world recommender system applications, user size and item size are very large. For example, Amazon has more than 200 million users, Apple iTunes has more than 800 million users. Recommender systems in real-world applications should be designed to be suitable for large-scale problems. Several works tried to solve large-scale problems. In [67, 61], people try to design large-scale recommender systems. In [16], large-scale recommender system is designed to recommend youtube contents to users.

2.7.4 Cold-Start Problem

User-item feedback history is a sparse matrix. Sometimes users do not have any feedbacks on any items or new item do not have any feedbacks from users. When lots of users and items do not have records in the feedback matrix, collaborative filtering models will fail to work in such a case. For example in Figure 2.2. user 4 and item 4 do not have any record. Though different algorithms try to use various information besides feedbacks to enhance the performance [69, 27, 57, 102, 2], cold-Start is still a challenge problem that need to be explored.

Implicit Feedbacks Matrix Y

	I_1	I_2	I_3	I_4	I_5
u_1	1	1	0	0	1
u_2	1	0	1	0	0
u_3	1	1	1	0	0
u_4	0	0	0	0	0

FIGURE 2.2: Cold-Start Problem Example

2.8 Contributions

Traditional methods have two problems. First, they only use point-wise comparison methods or pairwise ranking methods to learn Top-N recommender Systems. Second, they only use user-item feedback matrix to recommend items for users. Our contributions are trying to solve two problems.

First, Different from all these existing methods, we propose a novel personalized top-N recommendation approach that uses a combined heterogeneous loss function based on linear self-recovery models [97]. The heterogeneous loss integrates the strengths of pairwise ranking loss and point-wise recovery loss to enforce both entry-wise consistency and preference structure consistency between the recommendation recovery and the observations, which leads to improved recommendation predictions. We formulate the learning problem with this heterogeneous loss as a constrained convex minimization problem and develop a projected stochastic gradient descent optimization algorithm to solve it. The proposed approach is evaluated on a set of personalized top-N recommendation tasks. The experimental results show the proposed approach outperforms a number of state-of-the-art methods on top-N recommendation.

Second, we propose a novel predictive collaborative filtering approach to generate a list of top-N items with the highest recommendation scores for each target user [99]. The proposed approach exploits the item-based side information in addition to the conventional user-item recommendation matrix. To be more specific, it automatically identifies the recommendation scores for non-recommended items via low-rank aggregation over the users already-recommended items to recover the full recommendation matrix for all items. Simultaneously, it incorporates the item-based side information as an item-feature matrix and trains a regularized linear regression model to predict the recommendation scores from the item features for each user. We formulate this simultaneous user-item matrix self-recovering and predictive regression model training process as a joint low-rank constrained minimization problem, and derive an efficient

analytical solution for it. To evaluate the proposed approach, we conduct a set of experiments on a number of real world Amazon datasets with recommendation items such as books, beauty products, etc. The experimental results demonstrate that the proposed approach is very effective and efficient in producing top-N recommender systems, and achieves superior recommendation performance than the comparison methods.

Finally, we propose another recommender system model which also use side information [98]. We propose a joint discriminative prediction model that exploits both the partially observed user-item recommendation matrix and the item-based side information to build top-N recommendation systems. This joint model aggregates observed user-item recommendation activities to predict the missing/new user-item recommendation scores while simultaneously training a linear regression model to predict the user-item recommendation scores from auxiliary item features. We evaluate the proposed approach on a variety of recommendation tasks. The experimental results show that the proposed joint model is very effective for producing top-N recommendation systems.

CHAPTER 3

IMPROVING TOP-N RECOMMENDATION WITH HETEROGENEOUS LOSS

In this chapter, we propose our work Improving Top-N Recommendation with Heterogeneous Loss. Different from all the existing methods our proposed approach integrates the strengths of both point-wise comparison loss and pair-wise ranking loss.

3.1 Introduction

In the era of Internet, online shopping has become a close part of people's daily life. To facilitate consumers' online shopping experience, help consumers to find their interested items from the huge amount of online items and hence encourage online item purchases, effective top-N commercial recommendation systems, which suggest a small list of items that best match each consumers taste from the large amount of product items, have become increasingly important. Top-N recommendation systems automatically predict the recommendation scores over each item in the whole product item pool for each consumer, and recommend the items with high scores to the consumers. They can be widely used in many real world applications. For example, Amazon.com has a huge amount of online products, while consumers may not know

all the special items they would like. A top-N recommendation system, which can effectively suggest a set of personalized selected products for each consumer, can significantly increase the user-item purchase probability. Similarly, Netflix Inc. has thousands of online movies and TV shows, a personalized recommendation for users can help them to find the videos they may like. Yahoo! and Apple Itunes have millions of musics, Top-N recommendation systems can help users to find the musics they may like and provide a recommendation list to each user. In summary, an effective top-N personalized recommendation system can have significant real world commercial impacts.

Many methods have been developed in the literature to build top-N recommendation systems [63]. One classical technique is collaborative filtering (CF) [68, 79] which models the relationships between users and the correlations between items to identify new user-items relationship scores. Two main types of collaborative filtering methods include neighborhood-based CF [65, 66, 20, 82] and model-based CF [62, 60, 17, 75, 26, 73, 43]. Standard model-based CF methods perform matrix factorization to complete the missing recommendation entries, which exploits the low-dimensional subspace representations of the users and items [26, 86, 95]. Besides CF, sparse aggregation methods that exploit the linear correlations between items have also been explored in a few works [50, 14, 28, 15] to improve top-N recommendation performance. These methods have mainly pursued recommendation score recovery by minimizing the pointwise comparison loss between the reconstructed user-item matrix and the observed incomplete user-item matrix.

3.2 Related Work

Existing top-N recommendation methods have mostly focused on minimizing a uniform type of recommendation recovery loss to predict the recommendation scores. Based on the type of the losses, these recommendation systems can be categories into two main groups. The first group exploits pointwise comparison losses and the second

group exploits pairwise ranking losses.

3.2.1 Pointwise Comparison Methods

Many traditional top-N recommendation algorithms perform learning by minimizing the pointwise (entrywise) divergence of the reconstructed recommendation matrix from the original observation matrix. These include the collaborative filtering methods [68, 79, 26, 86, 95] and the sparse linear aggregation methods [50, 14, 28, 15]. For example, a weighted regularized matrix factorization (WRMF) model has been developed in [26], which minimizes the pointwise difference between the reconstructed matrix produced by the inner product of the user and item latent representations and the training matrix. Similarly, the work in [95] enforces the pointwise similarity between the training matrix and reconstructed matrix. It uses a non-locking parallel computing algorithm to perform matrix completion. In [50], a sparse linear method (SLIM) was proposed to recover the missing recommendation entries. It exploits the item-item correlations to reconstruct the incomplete recommendation matrix and minimizes the pointwise reconstruction loss between the reconstructed matrix and the input training matrix.

3.2.2 Pairwise Comparison Methods

Given a pair of items, a user naturally will prefer one to another, which forms the important pairwise preference structure of the recommendation data. The pointwise comparison methods however ignored this user personalized preference information. Recently, pairwise ranking methods are proposed to produce top-N recommendation systems by optimizing the preference structure consistency between the original matrix and the reconstructed recommendation matrix [86, 77, 11, 3, 56, 60]. Pairwise ranking methods treat training data as a set of triplet instances; for example, the triplet (i, j, k) is an instance that encodes the i-th user's preference to item j over item k. Different pairwise ranking losses have been exploited in these works. For

example, the pairwise ranking methods in [60, 3] optimize AUC scores; the work in [86] optimizes a normalized discounted cumulative gain; [94] explored the connection between the metric discounted cumulative gain and the binary classification to change the ranking problem into binary classification problems; [56] proposed a large-scale collaborative ranking method that exploits a max-margin hinge ranking loss to minimize the ranking risk in the reconstructed recommendation matrix. Nevertheless, these pairwise ranking methods ignored the entry-wise consistency between the reconstructed matrix and the original matrix. Different from all these existing methods, our proposed approach will perform top-N recommendations by optimizing a heterogeneous loss that integrates the strengths of both a pointwise comparison loss and a pairwise ranking loss.

3.3 Approach

We assume a partially observed user-item recommendation/purchase matrix $X \in R^{n \times m}$ over n users and m items is given. For implicit feedbacks, the recommendation/rating values are within $\{0, 1\}$, where the entry $X_{ij} = 1$ indicates a transaction or purchase record for the i-th user on the j-th item. The entry $X_{ij} = 0$ on the other hand means either the i-th user never purchased the j-th item or the transaction record has been removed and need to be predicted. We aim to identify the most interesting items for each user from his unrecommended/un-purchased list of items. Below we first we first introduce the general learning framework for Top-N recommendations and then instantiate it with novel objective losses. We finally present a stochastic gradient descent algorithm to solve the recommendation problem formulated.

3.3.1 A General Learning Framework for Top-N Recommendation

Given the input user-item recommendation/purchase matrix X, a general framework for Top-N recommendation performs learning by minimizing a regularized reconstruc-

tion loss function:

$$\underset{W}{\text{minimize}} \quad L(X, \hat{X}(W)) + R(W) \tag{3.1}$$

where $\hat{X}$ denotes the reconstructed recommendation matrix with a parametric model with parameter matrix W, $L(\cdot)$ denotes a convex reconstruction loss function and R denotes a regularization function. By substituting $L(\cdot)$ with different loss functions and employing different reconstruction models (denoted by the parameter W), many existing methods can be produced from this general framework as specific examples. For example, by using a matrix factorization model that reconstructs X as $\hat{X} = UV$ and a least squares point-wise loss function $L(X, \hat{X}) = ||X - \hat{X}||_F^2$, we can produce the simple pure singular value decomposition based (PureSVD) matrix factorization method [17] with proper constraints. Similarly, by reconstructing X with a linear aggregation model $\hat{X} = XW$ and using a least squares point-wise loss function $L(\cdot)$ and an integrated l_2 and l_1 norm regularization function $R(W) = \beta||W||_F^2 + \lambda||W||_1$, the sparse linear method (SLIM) [50] can be produced from the framework with additional proper constraints.

Moreover, the state-of-the-art pairwise ranking methods can also be produced as specific examples from this framework. For example, given the set of preference triplets, by using a matrix factorization model to reconstruct X as $\hat{X} = UV$, a max-margin hinge loss function $L_{rank}(X) = max(0, 1 - x)^2$, and $l2$ norm regularizers on U and V , we can produce the large scale pairwise ranking method (AltSVM) in [56]:

$$\underset{U,V}{\text{minimize}} \quad \sum_{(i,j,k)\in\Omega} L(Y_{ijk} \cdot U_i(V_j - V_k)^T) + \frac{\beta}{2}(||U||_F^2 + ||V||_F^2) \tag{3.2}$$

where Y_{ijk} is a function of X and such that $Y_{ijk} = 1$ if user i prefers item j over item k in X and $Y_{ijk} = 1$ otherwise; $||\cdot||_F$denotes the matrix Frobenius norm; U_i denotes the i-th row of U and V_j denotes the j-th row of V.

Our proposed approach can be conveniently formulated within this general learning framework as well. We integrate two types of losses into a novel heterogeneous loss

based on linear self-recovery models.

3.3.2 Novel Heterogeneous Loss

Pairwise ranking methods have demonstrated great top-N recommendation performance in the literature [56]. However, with the matrix factorization reconstruction function, the scalable learning problem produced in Eq 3.2 is a non-convex optimization problem. To introduce a convenient convex formulation with pairwise ranking losses, we propose to adopt a linear self-recovery model to reconstruct X. In particular, we use a linear reconstruction function $\hat{X} = XW$ with constraints W 0 and $diag(W) = 0$. This linear function exploits the statistical positive item-item correlations distributed among all the users to reconstruct the missing recommendation scores. The constraints are used to avoid trivial solutions. Then with a pair-wise ranking loss function $L_{rank}(\cdot)$, we can formulate the following convex pair-wise ranking problem to perform Top-N recommendation:

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & \sum_{(i,j,k)\in\Omega} L(Y_{ijk} \cdot X_i(W_j - W_k)^T) + \frac{\beta}{2}||W||_F^2 \\ \text{s.t.} \quad & W \geq 0, diag(W) = 0. \end{aligned} \tag{3.3}$$

Note for each user i, if he/she purchased item j but there is no purchase record for item k in the training matrix X, we say that user i prefers item j over item k and we have $X_{ij} > X_{ik}$. The set in the formulation above contains all the triplets (i, j, k) where $X_{ij} > X_{ik}$. To maintain the same pairwise preference structure, we assume that if $X_{ij} > X_{ik}$ in the original matrix X, one should also have $\hat{X}_{ij} > \hat{X}_{ik}$ in the reconstructed matrix $\hat{X}$. The loss function $L_{rank}(\cdot)$ aims to encode the pairwise preference ranking divergence between the given matrix X and the reconstructed matrix $\hat{X}$. In our implementation, we use the max-margin squared hinge loss function as the ranking loss, i.e., $L_{rank}(x) = max(0, 1 - x)^2$. By minimizing such a ranking loss function, the model will increase the consistency of the pair-wise preference rank-

ing between the reconstructed matrix $\hat{X}$ and the original matrix X. However, the consistency is only maintained at the pair-wise relative level. It does not necessarily guarantee the good quality of point-wise reconstruction, which might consequently hamper the accurate inference of unseen pairwise relationships. We nevertheless have an easy solution. The linear self-recovery model $\hat{X} = XW$ naturally allows one to encode the point-wise recovery loss between the entries of the reconstructed matrix $\hat{X}$ and the original matrix X, e.g., $L_{point}(X, \hat{X}) = ||X - \hat{X}||_F^2$. Previous works [50] have also demonstrated good Top-N recommendation performance with a least squares point-wise recovery loss. Hence we propose to combine both the pairwise ranking loss and the point-wise recovery loss to produce a new heterogeneous loss function for Top-N recommendation. Our learning problem with the heterogeneous loss function can be formulated as below:

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & \sum_{(i,j,k)\in\Omega} L(Y_{ijk} \cdot X_i(W_j - W_k)^T) + \frac{\alpha}{2}||X - XW^T||_F^2 + \frac{\beta}{2}||W||_F^2 \\ \text{s.t.} \quad & W \geq 0, diag(W) = 0. \end{aligned} \tag{3.4}$$

3.3.3 Optimization Algorithm

Due to the large size of the user-item matrix, standard gradient descent algorithms are not efficient for solving the minimization problem in Eq. 3.4. Moreover, though the learning problem is convex, the hinge loss function for pairwise ranking is non-differentiable and subgradients are typically needed. Hence in this work, we develop a projected stochastic gradient descent (SGD) algorithm to solve our target learning problem. The SGD technique has been widely used in recommendation systems with pairwise ranking [60, 11, 94], and it can also be extended to perform parallel computations [103, 59, 95].

With the pair-wise ranking loss, we use the user-item-item triplet (i, j, k) instances for the stochastic gradient descent procedure. In each iteration, we randomly choose

a triplet $(i, j, k) \in \Omega$ and make a SGD update:

$$
\begin{aligned}
w_j^{T+} &\leftarrow w_j^T - \eta\{L^{'}(w_j^T) + \frac{\alpha}{|\Omega^j|}X^T(Xw_j^T - X_{:,j}) + \frac{\beta}{|\Omega^j|}w_j^T\} \\
w_k^{T+} &\leftarrow w_k^T - \eta\{L^{'}(w_k^T) + \frac{\alpha}{|\Omega^k|}X^T(Xw_k^T - X_{:,k}) + \frac{\beta}{|\Omega^k|}w_k^T\}
\end{aligned}
\tag{3.5}
$$

where $|\Omega^j|$ and $|\Omega^k|$ denote the number of comparisons in that involve item j and item k respectively. $L^{'}_{w_j^T}$ and $L^{'}_{w_k^T}$ mean the derivation on w_j^T and w_k^T respectively, such that

$$
L^{'}(W_j^T) = \begin{cases} 2X_i^T(X_i(W_j^T - W_k^T) - Y_{ijk}), & Y_{ijk} \cdot X_i(W_j^T - W_k^T) \leq 1 \\ 0, & Y_{ijk} \cdot X_i(W_j^T - W_k^T) > 1 \end{cases} \tag{3.6}
$$

$$
L^{'}(W_k^T) = \begin{cases} -2X_i^T(X_i(W_j^T - W_k^T) - Y_{ijk}), & Y_{ijk} \cdot X_i(W_j^T - W_k^T) \leq 1 \\ 0, & Y_{ijk} \cdot X_i(W_j^T - W_k^T) > 1 \end{cases} \tag{3.7}
$$

The η parameter in the SGD update is the step size of gradient descent. We used a fixed small step size value in our experiments. The overall learning algorithm is given in Algorithm 1.

Algorithm 1 Projected Gradient Descent Algorithm to Learn W

Input: initialize $\alpha > 0, \beta > 0, \eta > 0$; $W^0 = 0^{m \times m}$.
Set $W = W^0$
for iter = 1 to MaxIter **do**
  Randomly select $(i, j, k) \in \Omega$
  Update w_j by using (3.5)
  Project updated entries into feasible set: $W_j(:) = max(W_j(:), 0)$, $W_{jj} = 0$;
  Update w_k by using (3.5)
  Project updated entries into feasible set: $W_k(:) = max(W_k(:), 0)$, $W_{kk} = 0$
  if converge **then**
    break
  end if
end for
return W

3.4 Experiments

In this section, we describe the setup of the experiments in detail. We report the experiment settings of our proposed method. We compare our method with other

state-of-the-art methods using standard measurement. Experiment results prove the efficiency of the proposed method. Moreover, We also evaluate parameter sensitivity. Finally, performance at different iteration times is also reported.

3.4.1 Setup

Dataset We use $Yahoo!Webscope^{TM}$ music ratings v1.0 and movie ratings v1.0 dataset, $MovieLens100k(ml-100k)$, $MovieLens1M(ml-1m)$ and $Netflix$ in our experiements. $Yahoo!Music$ dataset contains ratings for songs, the ratings are supplied by users during normal interaction with Yahoo! Music services. Smilarly, $Yahoo!Movie$ dataset contains ratings for different movies. In all datasets, we change the entries with positive values to 1 and convert the user-item matrix to implicit feedback matrix. Since the original rating matrix is sparse, for $Yahoo!Movie$ and $Yahoo!Music$ we keep the users with more than 20 ratings and items with more than 5 ratings. For $ml-100k$ we use original implicit feedback matrix. For $ml-1m$ and $Netflix$ we use the same dataset as [3]. For $ml-1m$, each user rate from 20 to 500 items and items are rated by 20 to 500 users, for $Netflix$, each user rate from 20 to 500 items and items are rated by 5 to 500 users. The statistic summary of the dataset is in Table 3.1.

Table 3.1: Statistic summary for experimental datasets

dataset	#user	#item	#trns	density
Yahoo! Music	2689	994	86907	3.95%
Yahoo! Movie	2382	924	104459	4.75%
ml-100k	943	1682	100000	6.3%
ml-1m	3850	2273	315869	3.6%
netflix	2979	2544	114865	1.5%

Statistic summary for experimental datasets. The columns of #users, #users and #trns show the number of users,items, and non-zero transactions respectively in each dataset. The columns of #rsize and #csize show the average number of transactions for each user and each item respectively. The column density shows the density of non-zero transactions for each dataset.

Evaluation Methods For each dataset, we split it into a training and a testing set. For each user, we randomly selected ten feedbacks and place them to the test

set, the rest are used as training set. Training set is used to build a model to learn the item-item correlation matrix. After training a ranked list of items are generated according to their scores in reconstruction matrix. We get Top-N(Default value is 10) recommendation list for each user from our ranked list and compare it to the testing set for this user. We evaluate the performance of proposed method and compare it with other methods by using two standard performance metrics: Top-N prediction precision Precision@N and the mean average precision MAP@N. The definition of Precision@K is as follows:

$$Precision@N(i) = \frac{1}{N}\sum_{j=1}^{N} T(i, P_i(j)) \tag{3.8}$$

where T denotes the test matrix and $P_(i)$. Precision@K is average of Precision@N(i) over all the users.

MAP@N detotes the mean average precision ot the Top-N predictions. The average precision (AP) of Top- prediction for the i-th user can be defined as:

$$AP@N(i) = \frac{\sum_{j=1}^{N} Precision@j(i) \times T(i, P_i(j))}{\sum_{j=1}^{N} T(i, P_i(j))} \tag{3.9}$$

MAP@N can be computed as the mean of the average precision of the Top-N predictions for all users, such that

$$MAP@N(i) = \frac{1}{n}\sum_{i=1}^{n} AP@N(i) \tag{3.10}$$

In our experiment, we randomly split each dataset into training and test matrices for five times and report the average test results in terms of these two evaluation metrics.

Compared Methods We compared the proposed method with four methods developed in the literature: WRMF [26], SLIM [50], RobiRank [94] and AltSVM [56].

Parameter Selection For the proposed method, we have two parameters: the

weight control parameter α and the regularization parameter β. α is selected from $\{10, 100, 1000, 5000, 10000\}$ and β is selected from $\{1e{-}3, 1e{-}2, 1e{-}1, 1, 10, 100, 1000\}$. For WRMF, the latent factor dimension f is selected from $\{10, 20, 50, 100, 200, 500, 1000\}$ and regularization parameter λ is selected from $\{0.1, 1, 10, 100, 1000\}$. For SLIM, $l2$ regularization parameter β is selected from $\{1, 10, 100, 200, 500, 1000\}$ and $l1$ regularization parameter λ is selected from $\{1e-3, 1e-1, 1, 10\}$. For RobiRank, the latent feature dimension f is selected from $\{10, 20, 50, 100, 200, 500, 1000\}$ and the regularization parameter λ is selected from $\{1, 10, 100, 200, 500, 1000\}$. For AltSVM, latent dimension is selected from $\{100, 200, 500, 1000\}$ and regularization parameter λ is selected from $\{100, 1000, 10000\}$. For each approach we report the best results and the corresponding parameter settings.

3.4.2 Experimental Results

The selected parameter settings and the experimental results in terms of Precision@N and MAP@N with $N \in \{5, 10\}$ for all the comparison approaches are reported in Table3.2. We can see that among the four comparison methods, *RobiRank* has poor performance and it produces the most inferior results on four datasets, $Yahoo!Music$, $Yahoo!Movie$, $Netflix$ and $ml-1m$. $SLIM$ performs much better than *RobiRank* and it even outperforms *RobiRank* on the remaining dataset ml-100k across three measurements except Precision@10. The pairwise preference ranking method, $AltSVM$, produces the best results on $Yahoo!Movie$ in terms of all the measurements among the four methods, and has similar performance as $SLIM$ on other datasets. The $WRMF$ method, which uses regularization control and applies different weights for the entries, outperforms all the other three methods on both $Yahoo!Music$ and $Nexflix$ in terms of the four measurements. But it has poor performance on $ml-100k$. Our proposed approach on the other hand outperforms all these four methods with remarkable margins on all the datasets across the four

measurements. These results suggest the proposed approach is indeed an effective method for Top-N recommendation.

3.4.3 Parameter Sensitivity Analysis

We have also conducted parameter sensitivity analysis for the proposed method on $Yahoo!Movie$ and $Yahoo!Music$ datasets. First, we tested different α values from $\{0, 10, 100, 1000, 5000, 10000\}$ by setting β to the fixed value used before. The experimental results are reported in Figure 3.1. We can see that for $Yahoo!Movie$ dataset, the precision values increase when α increases from 0 to 5000. When $\alpha = 5000$, the model performs the best. Similar results can be observed on $Yahoo!Music$ dataset. According to our objective function in Eq 3.4, if $\alpha = 0$, the model will only use the pair-wise ranking loss and will not combine the point-wise recovery loss. On the other hand, if α is too big, the contribution of the pairwise ranking loss can be diminished and the model will become a special version of the SLIM method. Our parameter analysis results suggest that we need a parameter in the middle to keep both types of losses, which indicates the two types of losses can complement each other. To further validate this, we have also compared our proposed approach to its variants that only use each individual loss. We call the variant that drops the point-wise recovery loss($\alpha = 0$) as pureRanking and the variant that drops the pair-wise ranking loss as variant of SLIM. The comparison results are reported in Figure 3.3. We can see that the proposed approach with integrated heterogeneous loss outperforms both variants that only use a uniform type of loss component. This validated our algorithm design of using heterogeneous losses. We also conducted parameter sensitivity analysis on the $l2-norm$ regularization parameter β. We fixed the value of α as 5000 and choose β value from $\{0, 1e-3, 1, 10, 1e3, 1e6\}$. The experiment results with different values are reported in Figure 3.2. We can see that the Top-N recommendation performance is not sensitive to the values when $\beta \leq 1000$.

Table 3.2: Comparison of Performance of Top-N Recommendation Algorithms

method	**Yahoo! Music**						
	params			Precision@5	Precision@10	MAP@5	MAP@10
WRMF	200	100	-	0.282 ± 0.002	0.226 ± 0.001	0.521 ± 0.002	0.482 ± 0.002
SLIM	100	1	-	0.273 ± 0.002	0.213 ± 0.001	0.504 ± 0.003	0.472 ± 0.002
RobiRank	10	100	-	0.261 ± 0.001	0.209 ± 0.001	0.461 ± 0.002	0.430 ± 0.002
AltSVM	500	1000	-	0.275 ± 0.003	0.219 ± 0.002	0.504 ± 0.003	0.470 ± 0.002
Proposed	5000	0.01	-	0.318 ± 0.001	0.244 ± 0.001	0.572 ± 0.002	0.526 ± 0.002
method	**Yahoo! Movie**						
	params			Precision@5	Precision@10	MAP@5	MAP@10
WRMF	200	100	-	0.390 ± 0.001	0.316 ± 0.001	0.595 ± 0.003	0.554 ± 0.003
SLIM	100	1	-	0.378 ± 0.002	0.303 ± 0.001	0.577 ± 0.003	0.539 ± 0.003
RobiRank	10	100	-	0.358 ± 0.002	0.290 ± 0.002	0.520 ± 0.003	0.483 ± 0.002
AltSVM	500	1000	-	0.391 ± 0.002	0.316 ± 0.001	0.608 ± 0.003	0.561 ± 0.003
Proposed	5000	0.01	-	0.430 ± 0.001	0.337 ± 0.001	0.657 ± 0.001	0.606 ± 0.001
method	**ml-100k**						
	params			Precision@5	Precision@10	MAP@5	MAP@10
WRMF	200	100	-	0.299 ± 0.001	0.243 ± 0.001	0.515 ± 0.003	0.481 ± 0.002
SLIM	200	1	-	0.323 ± 0.002	0.247 ± 0.001	0.553 ± 0.003	0.516 ± 0.003
RobiRank	20	100		0.317 ± 0.003	0.254 ± 0.001	0.520 ± 0.003	0.477 ± 0.002
AltSVM	200	1000	-	0.316 ± 0.002	0.245 ± 0.002	0.552 ± 0.003	0.516 ± 0.003
Proposed	5000	0.01	-	0.342 ± 0.001	0.269 ± 0.001	0.574 ± 0.002	0.524 ± 0.002
method	**ml-1m**						
	params			Precision@5	Precision@10	MAP@5	MAP@10
WRMF	1000	100	-	0.193 ± 0.002	0.160 ± 0.002	0.365 ± 0.002	0.355 ± 0.002
SLIM	200	1	-	0.193 ± 0.003	0.156 ± 0.002	0.372 ± 0.003	0.359 ± 0.002
RobiRank	20	1000		0.188 ± 0.002	0.154 ± 0.002	0.358 ± 0.003	0.349 ± 0.002
AltSVM	200	1000	-	0.189 ± 0.002	0.159 ± 0.002	0.362 ± 0.002	0.352 ± 0.002
Proposed	5000	1	-	0.211 ± 0.002	0.167 ± 0.001	0.397 ± 0.003	0.382 ± 0.002
method	**Netflix**						
	params			Precision@5	Precision@10	MAP@5	MAP@10
WRMF	200	100	-	0.199 ± 0.002	0.163 ± 0.002	0.374 ± 0.002	0.364 ± 0.002
SLIM	100	0.1	-	0.192 ± 0.002	0.153 ± 0.002	0.372 ± 0.002	0.363 ± 0.002
RobiRank	20	100		0.143 ± 0.002	0.115 ± 0.001	0.261 ± 0.002	0.267 ± 0.001
AltSVM	500	1000	-	0.195 ± 0.003	0.159 ± 0.002	0.367 ± 0.002	0.358 ± 0.002
Proposed	5000	0.1	-	0.220 ± 0.002	0.173 ± 0.002	0.416 ± 0.003	0.396 ± 0.002

Average test results of Top-N recommendations for all the comparison methods. The params columns contain the parameter settings for each approach. $WRMF$, *RobiRank* and $AltSVM$ have two parameters, the latent factor dimension and the regularization parameter. SLIM has two parameters, $l2$ and $l1$ norm regularization parameters. The proposed approach has two parameters, the weight control parameter and the regularization parameter. Bold-font indicates the best results.

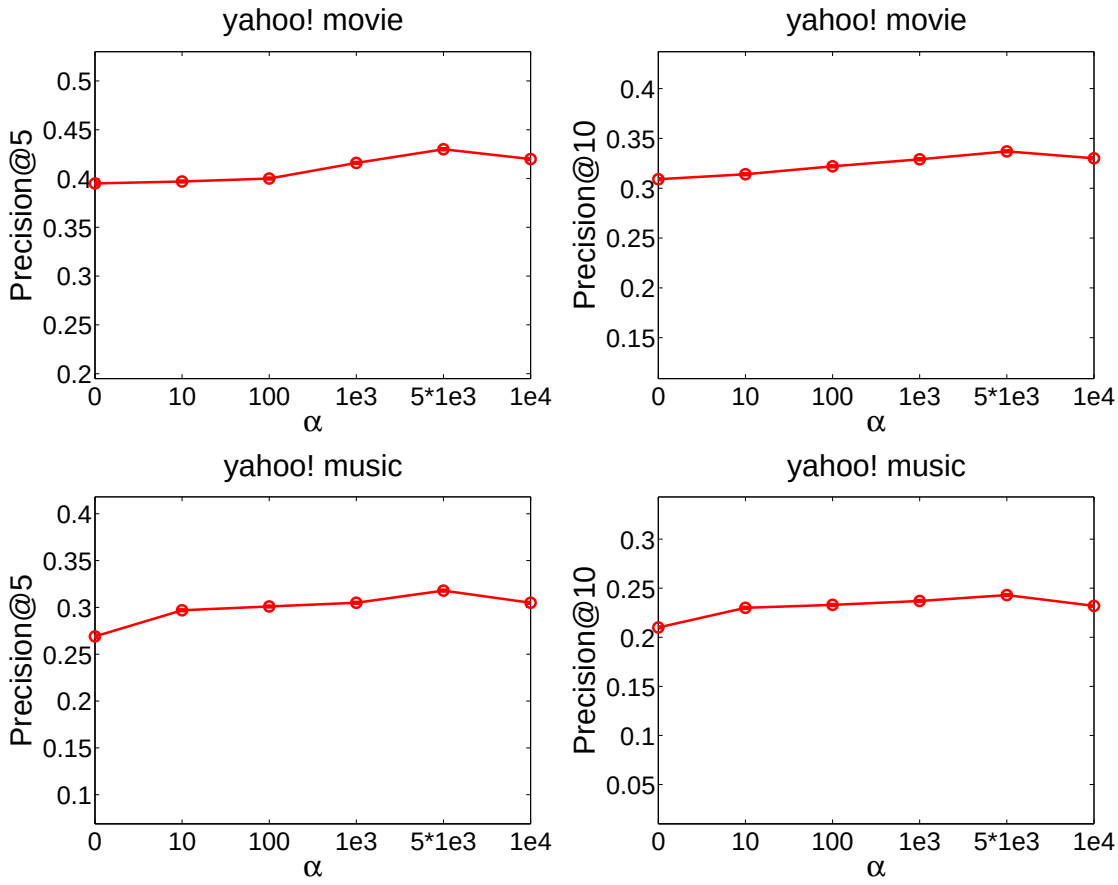

FIGURE 3.1: Analysis of parameter sensitivity of α on Yahoo! Movie and Yahoo! Music datasets

3.4.4 Iteration Number Analysis

There are more than 100 million pairwise comparisions for all datasets. In our algorithm, it will cost too much time if we try to use every comparision. In our SGD optimization algorithm, we randomly select one comparision from set $\Omega$for 1 million times. Since each comparision can be selected repeatedly, there are less than 1 million pairwise comparisions be selected. This raises the question: Is that enough for us to train a good model? Can the model perform better if we use more iterations? In order to answer this question, we choose different iteration number: The Iteration number$MaxIter$ is selected from $\{1e3, 1e4, 5e4, 1e5, 5e5, 1e6\}$. We record the Prediction Accuracy of different iterations on $Yahoo$! movie and music datasets. The results are in Figure 3.4. From the figure we can find that accuracy increases as the iteration

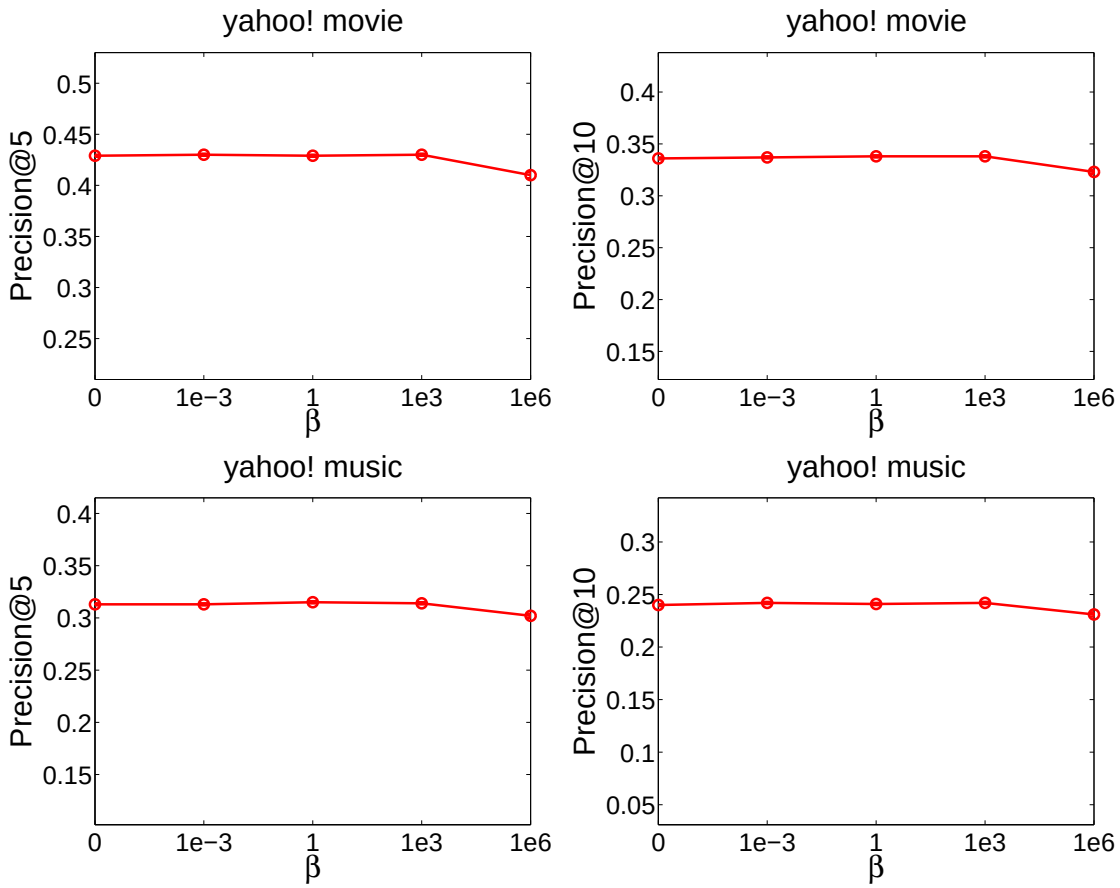

FIGURE 3.2: Analysis of parameter sensitivity of β on Yahoo! Movie and Yahoo! Music datasets

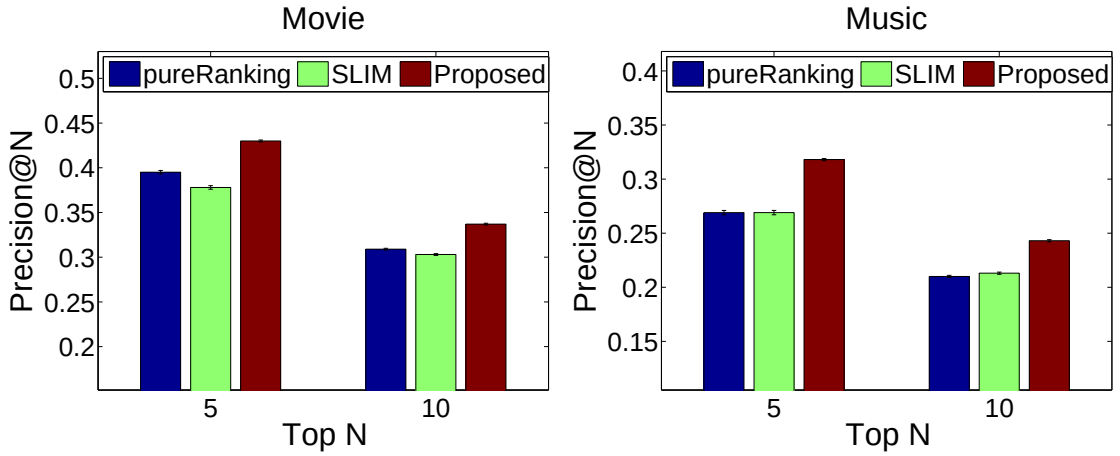

FIGURE 3.3: Performance of proposed method, SLIM and pureRanking($\alpha = 0$)

number increases from $1e3$ to $5e5$. After that, the prediction results become stable. This means it is not necessary to use more iterations to pick more comparisions to train the model.

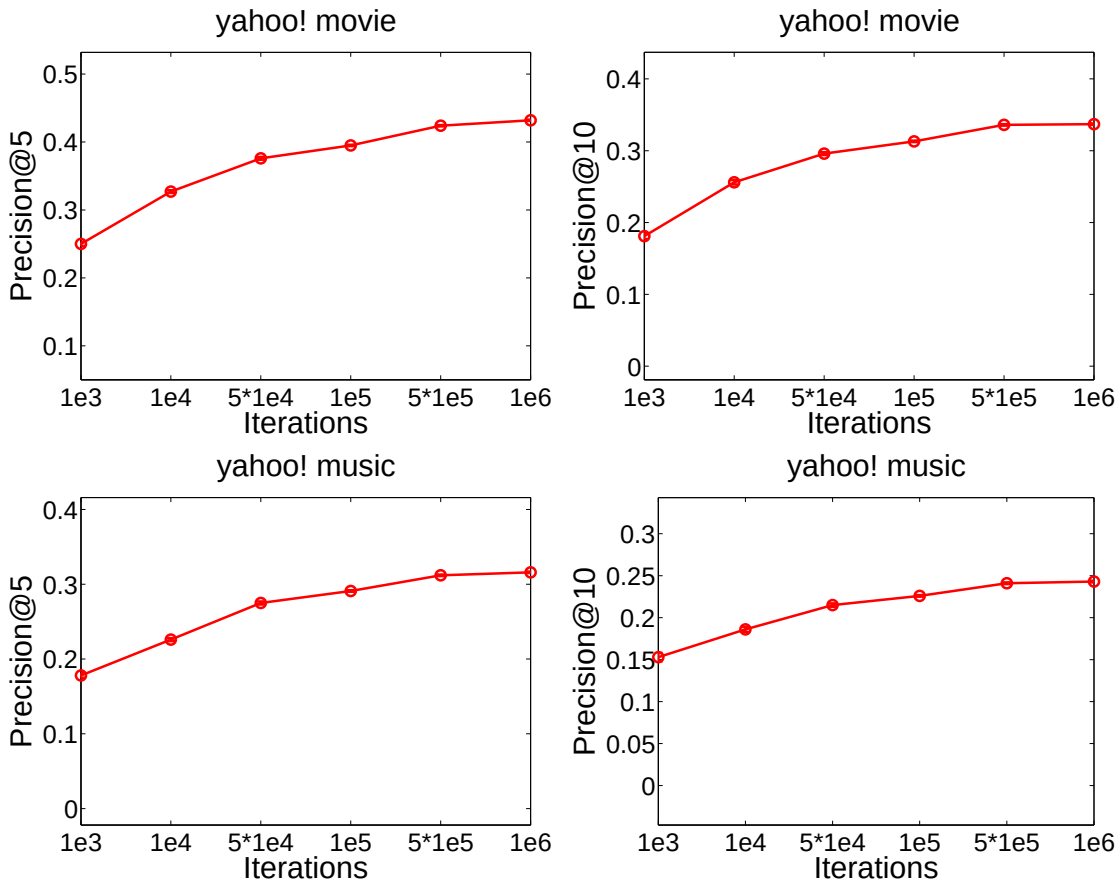

FIGURE 3.4: Analysis of Iteration Numbers on Yahoo! Movie and Yahoo! Music datasets

3.5 Conclusion

In this paper, we proposed a novel personalized Top-N recommendation approach that exploits novel heterogeneous loss functions based on linear self-recovery models. The heterogeneous loss integrates the strengths of both pairwise ranking loss and point-wise recovery loss to enforce both entry-wise consistency and pairwise preference structure consistency between the reconstructed recommendation matrix and the original observation matrix. We formulated the training problem with the hetero-

geneous loss as a constrained convex minimization problem and develop a projected stochastic gradient descent optimization algorithm to solve it. The proposed approach was evaluated on a set of real world personalized Top-N recommendation tasks. The experimental results showed that the proposed approach not only outperforms its two variants that only used either the comparison recovery loss or the pair-wise ranking loss, but also outperforms a number of state-of-the-art methods on Top-N recommendation.

CHAPTER 4

PREDICTIVE COLLABORATIVE FILTERING WITH SIDE INFORMATION

In this paper, we propose a novel predictive collaborative filtering approach to generate a list of Top-N items with the highest recommendation scores for each target user. The proposed approach exploits the item-based side information in addition to the conventional user-item recommendation matrix. To be more specific, it automatically identifies the recommendation scores for non-recommended items via low-rank aggregation over the users already-recommended items to recover the full recommendation matrix for all items. Simultaneously, it incorporates the item-based side information as an item-feature matrix and trains a regularized linear regression model to predict the recommendation scores from the item features for each user.

4.1 Introduction

Item-based recommender systems aim to recommend new items to a target user based on the user's previous recommendation activities (e.g., previous purchases, ratings, or clicks) [66, 7, 20, 53]. Recommending a ranked list of new items, which may be very attractive to the user but have not been observed in the user's previous recommendation activities, can encourage additional purchase or visits, which is very

important in real application scenarios. Recommender systems have been popularly studied due to their important impacts in E-commerce platforms such as Amazon and eBay, social networks such as Facebook and LinkedIn, and search engines such as Google and Bing [63]. A variety of works have been developed in the literature to address different recommender systems such as movie recommendations [55, 32, 60, 28], book recommendations [51], music recommendations [50], advert recommendations [78], news recommendations [54] and channel/program recommendations [26]. There are many different recommendation approaches, including content filtering-based approaches [48, 58] and collaborative filtering-based approaches [66, 20, 25]. The content filtering based approaches make recommendations to a target userby only using this particular users profile information (e.g., previous recommendation activities), while the collaborative filtering-based approaches also exploit other users' profile information. Usually, these recommender systems work on the user-item recommendation matrix and generate a list of top-N potential interesting items with the highest recommendation scores for each target user from the users non-recommended item pool. They typically retrieve relevant interesting items for target users by only using users history recommendation activities on items, but ignoring the existing valuable side information such as the reviews for books, or the plots for the movies, which limits their recommendation capacity and performance. In this paper, we propose a novel predictive collaborative filtering approach to generate a list of top-N items with the highest recommendation scores for each target user. The proposed approach exploits the item-based side information in addition to the conventional user-item recommendation matrix. To be more specific, it automatically identifies the recommendation scores for non-recommended items via low-rank aggregation over the users' already-recommended items to recover the full recommendation matrix for all items. Simultaneously, it incorporates the item-based side information as an item-feature matrix and trains a regularized linear regression model to predict the recommendation scores from the item features for each user. We formulate this simultaneous user-item matrix

self-recovering and predictive regression model training process as a joint low-rank constrained minimization problem, and derive an efficient analytical solution for it. To evaluate the proposed approach, we conduct a set of experiments on a number of real world Amazon datasets with recommendation items such as books, beauty products, etc. The experimental results demonstrate that the proposed approach is very effective and efficient in producing top-N recommender systems, and achieves superior recommendation performance than the comparison methods.

4.2 Related Work

A variety of works have been developed in the literature to address item-based recommender systems. We provide a brief review below on the most related collaborative filtering approaches and side information assisted approaches. Collaborative filtering approaches make recommendations by using both the target user's previous recommendation activities and other users' history behaviors [79], which include two groups of methods, neighborhood-based methods [66, 20, 28] and latent factor-based methods [7, 25, 62, 64, 86, 32, 17, 75]. Neighborhood-based collaborative filtering methods make recommendations by exploiting similarities among items within the user-item recommendation matrix. Once item-item similarities are computed, the prediction on a new item for a user is obtained by considering the particular user's past recommendation activities on similar items. For example, the work in [66] proposed to use similarity weighted sum prediction to recommend new items for a target user. In order to measure the item-item similarities, various similarity schemes have been investigated, including cosine similarity, correlation similarity, adjusted cosine similarity [66], and conditional probability similarity [20]. Unlike the neighborhood-based methods, which directly employ the user-item recommendation matrix for prediction, the latent factor-based collaborative filtering methods first identify a set of latent factors for users and items respectively. They then train a prediction model on the latent space to make recommendations. In order to iden-

tify the latent factors, a variety of models have been employed, including the pure singular value decomposition (PureSVD) method [17], the Gaussian probabilistic latent semantic analysis method [25], the latent Dirichlet allocation method [7], the matrix factorization method [32], the maximum margin matrix factorization methods [62, 86], the pairwise comparison method [60], the restricted Boltzmann machines [64], and the weighted non-negative matrix factorization method [75]. In addition to these two groups, some other collaborative filtering recommender systems have also been developed in the literature [31, 50, 73]. Koren [31] proposed an asymmetric-SVD based hybrid approach to address recommendation problems, which combines the similarity-based neighborhood method and the latent factor-based method together for recommendations. Ning and Karypis [50] proposed a sparse linear method to address top-N recommendation problems, which computes the recommendation score on a non-purchased/non-recommended item for a target user using sparse aggregation of purchased/recommended items of the particular user. Shi et al. [73] proposed a collaborative less-is-more filtering approach, which maximizes the mean reciprocal rank to address recommendation problems in social networks. Different from collaborative filtering approaches, which only exploit previous recommendation activities in user-item matrix, side information assisted approaches also exploit additional meta-data information (e.g., product reviews, film plots, etc) to recommend new items for a target user [78, 24, 19, 51]. Stern et al. [78] proposed to use both user meta-data such as a user's job, age, gender and id, and item meta-data such as a movie's genre and id. They represented these meta-data as feature vectors for the users and items respectively and incorporated the meta-data information into the collaborative filtering recommender system. The work in [24] incorporated both the user-item recommendation information and the item meta-data into a unified Boltzmann machine to form a hybrid recommender system. Similarly, the work in [19] developed a hybrid recommender system based on Bayesian networks over user nodes, item nodes, and item-feature nodes. The approach propagates content-based information from recom-

mended items to nonrecommended items via the item-feature nodes, and propagates collaborative-based information via item nodes. [51] proposed to incorporate item-based side information into the sparse linear method [50] to learn the aggregation coefficient matrix and address top-N recommendation problems, which has demonstrated superior performance over conventional recommender systems without side information. Different from these previous methods, our proposed approach performs a joint predictive recovery of the user-item matrix for top-N recommendations by integrating the partially observed user-item matrix and the item-based side information such as reviews in a discriminative and mutually complementary manner. A few recent works [4, 45, 87] have exploited user reviews for building recommender systems, while [10] provides a survey of such recent attempts. However most of these methods require the reviews to be directly associated with the user ratings, while our approach treats reviews as side information and can exploit reviews from auxiliary resources that have no direct relationships with the target users.

4.3 Approach

In this section, we present the proposed predictive collaborative filtering approach for item-based Top-N recommender systems. We assume an observed item-user recommendation matrix $Y \in R^{n \times t}$, which contains the observed recommendation (or purchase) activities of t users over n product items. In this matrix, an observed entry Y_{ij} (with an indicator value 1 or a positive recommendation score) indicates the i-th product item has been recommended (or purchased) by the j-th user, while an unobserved entry $Y_{ij} = 0$ only indicates an unknown relationship between the j-th user and the i-th product item, i.e., the recommendation (or purchase) action of the j-th user over the i-th item has not been observed in the users past recommendation activities, but it is possible that the j-th user can be interested in the i-th item. As item-based side information in the form of product reviews for E-commerce applications or film plots for movie recommendations exists in many application domains, we take the

item-based side information into account as well. In particular, we assume there is an item-feature matrix $X \in R^{n \times d}$, whose each row contains a feature vector for the corresponding product item and presents the item-based side information. We aim to automatically identify the most promising interesting product items, in particular the Top-N items, for each target user from his non-recommended items by using both the observed item-user recommendation matrix Y and the item-feature matrix X. Below we first present a matrix self-completion component based on the item-user recommendation matrix, and then incorporate the predictions from the item-feature matrix to form a joint prediction model. We finally derive a convenient analytical solution for the proposed joint learning method. In the rest of the paper, we use $\mathbf{1}$ to denote a column vector of all 1s, while assuming the size of the column vector can be inferred from the context. We use I_n to denote an identity matrix with size n. For any matrix X, $||X||_F$ denotes its Frobenius norm, and $X^\dagger$ denotes its pseudo-inverse.

4.3.1 A joint Prediction Model

Given the past recommendation/purchase activities in the observed item-user recommendation matrix Y , the recommendation score on a non-recommended/non-purchased item v_i for a user u_j can be calculated as a linear aggregation of the items that have already been recommended/purchased by u_j , that is, $\tilde{Y}_{ij} = w^T Y_{:j}$, where $Y_{:j}$ denotes the j-th column of Y. Such a linear recovery model can be built for all items with $Y : j = WY_{:j}$, while the linear functions have to be learned from the statistical item aggregation relationships existed in the observed data for all the users. This leads to a matrix self-recovery (self-completion) formulation $\tilde{Y} = WY$, where W is an $n \times n$ linear aggregation coefficient matrix that is shared across all the t users. A meaningful matrix self-recovery should automatically complete the unobserved recommendation scores in the matrix without affecting much the observed recommendation scores. We hence enforce the self-recovered item-user matrix to maximally preserve the original matrix values and formulate the self-recovery model as the following learning problem:

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & ||Y - WY||_F^2 \\ \text{s.t.} \quad & rank(W) \leq \lambda \end{aligned} \tag{4.1}$$

To avoid the trivial solution of setting W as an identity matrix, we assume the coefficient matrix W should be low rank and encode this low rank property using a rank constraint with a hyper parameter λ. This low rank property can also allow the model to implicitly exploit the latent low-dimensional representations of the items.

In addition to the item-user recommendation matrix Y, we also have explicit item-based side information available as an item-feature matrix X, where each item feature vector describes the properties of a given product item. Note that this product item side information is common to all users. But different users may have different tastes over the same product item. The recommendation scores of each user over the set of product items depend on the properties of the product items. Hence we propose to predict the recommendation score of each user over a product item from the feature vector of this product item. In particular, we can use a linear regression model, $f_j(x) = x^T q + b$, for the j-th user to predict his or her recommendation score y from the item feature vector x. Given the observed item-feature matrix X and the recovered item-user recommendation matrix $\tilde{Y}$, the linear regression model f_j for the j-th user can be trained on the data by minimizing a least squares regression loss $||Xq + b\mathbf{1} - \tilde{Y}_{:j}||^2$ over the model parameters q and b. Thus the linear regression model is specifically tailored to the j-th user and can help to capture user-specific preferences of the product items when making recommendations. For all the t users, we will train t linear regression models from the data by minimizing the following regularized least squares loss function:

$$\underset{W}{\text{minimize}} \quad ||XQ + \mathbf{1}\mathbf{b}^T - \tilde{Y}||_F^2 + \beta||Q||_F^2 \tag{4.2}$$

where $Q \in R^{d \times t}$ and $\mathbf{b} \in R^t$ are the model parameters for the t linear regression models, such that $f_j \mathbf{x} = \mathbf{x}^T Q_{:j} + \mathbf{b}_j$ for the j-th user.

Since $\tilde{Y} = WY$ is a self-recovered matrix via Eq.4.1, by integrating the recommendation matrix recovery in Eq.4.1 and the linear regression model training in Eq.4.2 together, we formulate a joint recommendation prediction model as follows to perform joint recommendation score prediction in a mutually complementary manner:

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & ||XQ + \mathbf{1}\mathbf{b}^T - \tilde{Y}||_F^2 + \beta||Q||_F^2 + \gamma||Y - WY||_F^2 \\ \text{s.t.} \quad & rank(W) \leq \lambda \end{aligned} \tag{4.3}$$

where β and γ are the trade-off parameters. By solving this joint minimization problem, we expect to integrate both the existing recommendation activity information and the item-based side information to predict the recommendation scores of each user over his or her non-recommended/non-purchased items. The Top-N recommendations for a user u_j can be produced by sorting u_j's non-recommended/non-purchased items in a decreasing order based on their predicted recommendation scores.

4.3.2 Learning Algorithm

The learning problem in Eq.4.3 is a joint minimization problem over both the linear regression model parameters $\{Q, \mathbf{b}\}$ and the linear aggregation coefficient matrix W. We first perform minimization over the linear regression model parameters assuming fixed W, which produces the following closed-form solutions for Q and $\mathbf{b}$:

$$Q = (X^T H X + \beta I_d)^{-1} X^T H W Y \tag{4.4}$$

$$\mathbf{b} = \frac{1}{n}(WY - XQ)^T \mathbf{1} \tag{4.5}$$

where $H = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T$ is centering matrix.

By plugging the solutions for the regression model parameters Q in Eq.4.4 and $\mathbf{b}$ in

Eq.4.5 back into the objective function in Eq.4.3, the joint learning problem can be equivalently reformulated into the following minimization problem over W,

$$\underset{W:rank(W)\leq\lambda}{\text{minimize}} \quad ||BWY||_F^2 + \beta||AWY||_F^2 + \gamma||Y - WY||_F^2 \tag{4.6}$$

where $A = (X^THX + \beta I_d)^{-1}X^TH$ and $B = H(XA - I_n)$. By further introducing the following replacement matrices,

$$F = \begin{bmatrix} B \\ \sqrt{\beta}A \\ \sqrt{\gamma}I_n \end{bmatrix}, S = \begin{bmatrix} O_{(n+d)\times t} \\ \sqrt{\gamma}Y \end{bmatrix} \tag{4.7}$$

where $O_{(n+d)\times t}$ denotes a $(n+d)\times t$ matrix with all 0s, the problem 4.6 can be rewritten as:

$$\underset{W:rank(W)\leq\lambda}{\text{minimize}} \quad ||FWY - S||_F^2 \tag{4.8}$$

Though the optimization problem in Eq.4.8 is still a non-convex problem due to the rank constraint, below we will derive a convenient analytical solution for it.

For matrix F, we denote its thin SVD as $F = U_F\Sigma_F V_F^T$. Two projections can be defined consequently as $P_{F,L} := U_F U_F^T$ and $P_{F,R} := V_F V_F^T$. Similarly, we can define projections $P_{Y,L}$ and $P_{Y,R}$ for matrix Y. We then have the following theorem.

Theorem 1 For $N \in \lambda \leq rank(P_{F,l}SP_{Y,R})$, let $M = P_{F,l}SP_{Y,R}$, then

$$W^* = F^\dagger M_{(\lambda)} Y^\dagger \tag{4.9}$$

is a solution to the minimization problem in 4.8, where the operation $M_{(\lambda)}$ extracts the truncation of M by setting all of its singular values as zeros except the λ largest ones, i.e.,

$$M = U_M\Sigma_M V_M^T, M_{(\lambda)} = U_M\hat{\Sigma}_M(\lambda)V_M^T \tag{4.10}$$

and $hat\Sigma_M(\lambda)$ is the truncation of Σ by keeping only the largest λ diagonal values of Σ and setting all others as zeros. The solution is unique iff the λ-th and $(\lambda+1)$-th largest singular values of M are different.

Theorem 1 was proved by [76] and [21]. It provides an analytical closed-form solution for our rank constrained minimization problem 4.8. The overall learning algorithm is summarized in Algorithm 2.

Algorithm 2 Learning Algorithm

Input: X, Y, $\beta > 0$, $\gamma > 0$, $\lambda > 0$.
Procedure:
1. Compute auxiliary matrices F and S with Eq. 4.7
2. Perform singular value decomposition over F and Y, compute $P_{F,L}$and $P_{Y,R}$, and let $M = P_{F,L}SP_{Y,R}$.
3. Compute a solution $\tilde{W}^*$ using Eq. 4.9, and set $\tilde{Y}^* = WY$.

The complexity of Algorithm 2 is dominated by the three singular value decomposition operations on F, Y and M respectively. This leads to a computational complexity of $O((2n+d)n^2 + min(n^2t, nt^2) + min((2n+d)^2t, (2n+d)t^2))$ for the proposed algorithm when a standard full SVD algorithm is used [8]. Moreover, based on the definitions of F and S in Eq. 4.7 and the definition of M in Theorem 1, it is easy to conclude that $rank(F) = n$ and $rank(M) \leq rank(S) = rank(Y)$. The item-user recommendation matrix Y is typically low rank due to the existence of dependencies and correlations among the users activities. Hence if $r = rank(Y)$, one could have $r \ll min(n, t)$. In this case, the computational complexity of Algorithm2 can be further reduced to $O((2n+d)n^2 + r(3n+d)t)$ by using a fast SVD algorithm in [8].

After obtaining the solution W^*in Eq.4.9 for the coefficient matrix, we can get the recovered item-user recommendation matrix $\tilde{Y}^*$ by setting $\tilde{Y}^* = WY$, which contains the predicted recommendation scores for the originally non-recommended/non-purchased items of each user. We then perform Top-N recommendation in the following way. For the j-th user, $\tilde{Y}^*_{:j}$ denotes his predicted recommendation scores over all the product items. We rank the scores for non-recommended items in a non-increasing

Table 4.1: Statistic summary for the experimental datasets.

dataset	#items	#users	#trns	rsize	csize	density
Beauty	7034	5294	23576	4.45	3.35	0.063%
Office	6167	2389	10988	4.60	1.78	0.075%
Sports	7085	7971	32742	4.10	4.62	0.058%
Gourmet Foods	8322	4780	23553	4.93	2.83	0.059%
Health	7877	9121	38215	4.19	4.85	0.053%

order and consider the first N items as the predicted recommendation items.

4.4 Experiments

In this section, we present our empirical evaluations and discussions of recommender systems on real-world datasets.

Datasets We conducted experiments on five real-world Amazon datasets[45], $Beauty$, $Office$, $Sports\&Outdoors$, $GourmetFoods$ and $Health$, each of which corresponds to one category of Amazon products. For each category, the original dataset contains transactions between different product items and users, indicated with non-zero multivariate rating values. We converted the multivariate rating values to 1s and filtered those less popular product items and users that appeared less than three transactions to construct the item-user recommendation matrix. The statistic summary of the produced datasets is presented in Table 4.1. For each dataset, we present the number of items (#items), the number of users (#users), the number of transactions (#trns), the average number of transactions for each item (#rsize), the average number of transactions for each user (#csize) and the density value (#trns/(#users×# items)).

Item-based Side Information For each dataset, we used the product reviews as the item based side information for the product items. We extracted the unigram features from the review articles with stop words being removed. We further selected the most frequent 5000 unigram features as the item features and represented each product item as a bag-of-word feature vector with term frequency feature values(F_{tf}).

We denote the recommender systems without item-based side information as F_{no}.

Comparison Approaches We compared our proposed approach with the following approaches for recommender systems: (1) Pure Singular Value Decomposition (PureSVD), which is a baseline collaborative filtering method based on singular value decomposition [17]; (2) Bayesian Personalized Ranking for matrix factorization (BPRMF), which exploits pairwise comparison [60]; (3) Sparse Linear Model (SLIM), which learns a sparse coefficient matrix to induce recommendation scores [50]; and (4) Collective Sparse Linear Model (cSLIM), which incorporates item-based side information into the SLIM model to generate the Top-N recommendations [51]. These methods address collaborative filtering from different aspects and have demonstrated effective performance in the literature.

Evaluation Criteria We used the commonly used leave-one-out cross validation to evaluate the performance of each recommendation algorithm over five runs. For each run, we randomly chose one transaction for each user and put it into the test set. The training set is the masked item-user transaction matrix with the transactions of the test set removed. We ran each recommender system five times based on random partitions of the training and test sets. The evaluation is conducted on the top-N items recommended for each user by the recommender system. We set N=10 and used the following two criteria to evaluate the top-N recommendation performance: $HitRate(HR)$ and $AverageReciprocalHitRank(ARHR)$ [20],

$$HR = \frac{\#\text{hits}}{\#\text{users}}, \quad ARHR = \frac{1}{\#\text{users}} \sum_{i=1}^{\#\text{hits}} \frac{1}{p_i}, \tag{4.11}$$

where #users is the number of users, #hits is the number of users with the test item recommended in the Top-N list, and p_i is the position of the item in the Top-N ranked list. Larger HR and ARHR values indicate better performance.

4.4.1 Experimental Results

We compared the five comparison methods $PureSVD$, $BPRMF$, $SLIM$, $cSLIM$, and *Proposed* on the five datasets using the evaluation criteria presented above. The average comparison results for all these methods in terms of HR and ARHR are presented in Table 4.2. We can see that the baseline PureSVD method does not work well on all the datasets comparing to the other systems, which shows that the pure singular value decomposition-based collaborative filtering method is far from enough to develop a good recommender system without using other information. The BPRMF and SLIM methods on the other hand work much better than the PureSVD method on all the datasets in both measurements. The cSLIM method, which utilizes item-based side information, outperforms PureSVD, BPRMF and SLIM across all cases except on Health in terms of ARHR. This shows that the item-based side information is useful for improving the recommendation performance. Our proposed approach on the other hand consistently produced the best results among all the comparison methods on all the five datasets in terms of both measurements. This demonstrates that our proposed approach provides an effective mechanism to integrate information from the item-user matrix and the item-based side information for producing Top-N recommender systems. Moreover, we also recorded the model learning time and the recommendation test time for all the comparison approaches in all these experiments. These running times are reported in Table 4.2 as well. We can see that with an analytical solution our proposed method is very efficient in learning. It significantly reduced the training time by comparing not only to the cSLIM method which also utilizes item-based side information, but also to SLIM and BPRMF that are not able to incorporate side information. On the datasets with larger recommendation matrices such as Gourmet Foods and Health, the differences between the model learning times of the proposed method and the cSLIM method are quite dramatic. Though the PureSVD method runs faster than our proposed approach,

this baseline cannot exploit side information and produced very poor performance. Our proposed approach on the other hand maintains both efficiency and effectiveness. All these results demonstrate that the proposed approach is an effective and efficient state-of-the-art solution for Top-N recommender systems.

4.4.2 Integration Model Study

To verify the effectiveness of the integration mechanism of the proposed joint model, we also compared the two individual components, the self-completion component and the sideinformation based prediction component, to the proposed joint model in terms of top-N recommendation performance. The comparison results in terms of HR and ARHR are reported in Table 3. We can see that the proposed joint model consistently outperforms both of the individual components. With each component alone, the model cannot achieve a state-of-the-art performance. This suggests that our proposed model provides an effective integration over the individual information sources.

4.4.3 Parameter Sensitivity Analysis

The proposed approach has three parameters (β, γ, λ), we then conduct sensitivity analysis for them. We used the same experimental setting as before and conducted five-time leave-one-out cross-validation. For the item side information, we used term frequencies (F_{tf}). Note that the parameter γ weights the component of using the conventional item-user recommendation matrix. We chose γ from $\{1e-3, 1e-2, 0.05, 0.1, 1, 10\}$ while fixing the other two parameters. We then present the average test results (HR and ARHR) over five runs on three datasets in Figure 4.1 and 4.2. As we can see, the performance varies as we change γ, which is understandable since it controls the importance of the two components of our recommender system.

feature	method	Beauty						
			params		HR	ARHR	mt	tt
F_{no}	PureSVD	100	-	-	0.214 ± 0.005	0.178 ± 0.005	112s	0.5s
	BPRMF	500	0.01	-	0.264 ± 0.001	0.234 ± 0.002	1900s	0.6s
	SLIM	10	0.01	-	0.289 ± 0.001	0.258 ± 0.001	3500s	5s
F_{tf}	cSLIM	1e-6	10	0.01	0.299 ± 0.001	0.263 ± 0.001	38000s	5s
	Proposed	200	0.01	2000	0.346 ± 0.001	0.284 ± 0.002	1800s	10s
feature	method	**Office**						
			params		HR	ARHR	mt	tt
F_{no}	PureSVD	100	-	-	0.095 ± 0.005	0.075 ± 0.005	21s	0.3s
	BPRMF	200	0.01	-	0.132 ± 0.003	0.095 ± 0.001	800s	0.1s
	SLIM	100	0.001	-	0.166 ± 0.003	0.110 ± 0.002	1200s	2s
F_{tf}	cSLIM	1e-6	100	0.001	0.177 ± 0.002	0.111 ± 0.002	2800s	2s
	Proposed	100	0.1	2000	0.219 ± 0.001	0.151 ± 0.001	600s	6s
feature	method	**Sports&Outdoors**						
			params		HR	ARHR	mt	tt
F_{no}	PureSVD	500	-	-	0.154 ± 0.005	0.126 ± 0.005	200s	2s
	BPR	500	0.01	-	0.186 ± 0.002	0.158 ± 0.001	3500s	0.3s
	SLIM	10	0.01	-	0.221 ± 0.001	0.185 ± 0.001	4300s	7s
F_{tf}	cSLIM	1e-6	10	0.01	0.224 ± 0.001	0.187 ± 0.001	46000s	5s
	Proposed	200	0.05	2000	0.238 ± 0.001	0.187 ± 0.001	1000s	10s
feature	method	**Gourmet Foods**						
			params		HR	ARHR	mt	tt
F_{no}	PureSVD	500	-	-	0.069 ± 0.005	0.045 ± 0.005	150s	0.5s
	BPRMF	200	0.01	-	0.076 ± 0.002	0.053 ± 0.001	2100s	0.4s
	SLIM	10	0.01	-	0.122 ± 0.002	0.074 ± 0.001	4800s	6s
F_{tf}	cSLIM	1e-6	10	0.01	0.129 ± 0.002	0.081 ± 0.001	52000s	5s
	Proposed	200	0.05	2000	0.173 ± 0.001	0.089 ± 0.001	1400s	9s
feature	method	**Health**						
			params		HR	ARHR	mt	tt
F_{no}	PureSVD	100	-	-	0.122 ± 0.005	0.095 ± 0.003	250s	2s
	BPRMF	500	0.01	-	0.162 ± 0.001	0.121 ± 0.001	4500s	1s
	SLIM	1	0.01	-	0.191 ± 0.001	0.159 ± 0.001	8000s	12s
F_{tf}	cSLIM	1e-7	1	0.01	0.209 ± 0.001	0.155 ± 0.001	70000s	10s
	Proposed	200	0.05	2000	0.211 ± 0.001	0.161 ± 0.001	1400s	14s

Table 4.2: Performance for all approaches in Sports&Outdoor, Office, and Beauty. The params columns contain the parameter setting for each approach. *PureSVD* has one parameter f, indicating the number of latent factors; *SLIM* method has two parameters, the ℓ_2-norm regularization parameter β and the ℓ_1-norm regularization parameter λ; *cSLIM* method has three parameters, the side-information weighting parameter α, the ℓ_2-norm regularization parameter β and the ℓ_1-norm regularization parameter λ. The proposed model has three parameters, β, γ, λ. The mt-column records the model learning time and tt-column records the test time in seconds (s). Bold-format indicates the best results.

Table 4.3: Performance of two seperate components compared to proposed Integrated method

method	**Beauty**	
	HR	ARHR
Side Infromation	0.048 ± 0.000	0.022 ± 0.000
Purchase Matrix	0.286 ± 0.002	0.256 ± 0.002
Proposed	0.346 ± 0.002	0.284 ± 0.001
method	**Office**	
	HR	ARHR
Side Infromation	0.057 ± 0.000	0.019 ± 0.000
Purchase Matrix	0.152 ± 0.001	0.102 ± 0.001
Proposed	0.219 ± 0.001	0.151 ± 0.001
method	**Sports&Outdoors**	
	HR	ARHR
Side Infromation	0.034 ± 0.000	0.011 ± 0.000
Purchase Matrix	0.216 ± 0.001	0.183 ± 0.001
Proposed	0.238 ± 0.001	0.187 ± 0.001
method	**Gourmet Foods**	
	HR	ARHR
Side Infromation	0.048 ± 0.000	0.022 ± 0.000
Purchase Matrix	0.113 ± 0.001	0.070 ± 0.001
Proposed	0.173 ± 0.001	0.089 ± 0.001
method	**Health**	
	HR	ARHR
Side Infromation	0.036 ± 0.000	0.0138 ± 0.000
Purchase Matrix	0.185 ± 0.003	0.154 ± 0.002
Proposed	0.211 ± 0.001	0.161 ± 0.001

Average performance in terms of HR and ARHR for different components on *Beauty*, $Office$, *Sports&Outdoors*, *GourmetFoods* and $Health$.

A large γ favours the recommender system based on the conventional item-user recommendation matrix while a small γ favours the item side information component. Our recommender system achieves the best prediction results when it finds a good balance between those two components. Those results do demonstrate the usefulness of both components in the proposed top-N recommender system.

The parameter λ is directly related to the recovery matrix M, which indicates the rank of the induced matrix should below λ. The dimensionality of M is $n \times n$, then we chose {100, 500, 1000, 1500, n} while fixing the other two parameters. Similarly,

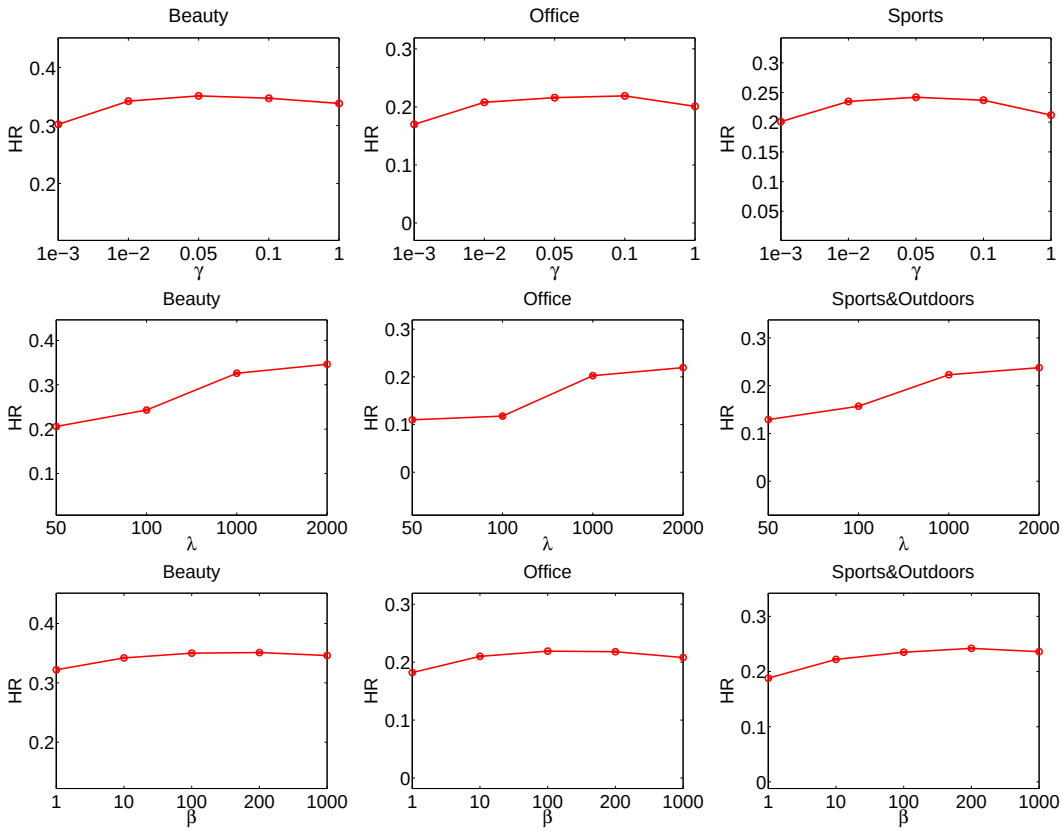

FIGURE 4.1: Parameter sensitivity analysis in terms of HR scores for γ, λ, β on Office, Beauty and Sports&Outdoor datasets.

we conducted five-time leave-one-out cross-validation and used term frequency (F_{tf}) for the item-feature matrix X. We then present the average performance in Figure 4.1 and 4.2. As we can see that, the performance improves as we start to increase λ from 100. When reaching a certain range (about 1000), its performance does not change too much if we continue to increase λ.

Finally, we investigate the parameter β, which controls the regularizer of the linear regression model parameter matrix W. We chose β from {1, 10, 100, 200, 1000} and fixed the other two parameters. After running each task over five times, we then provide experimental results in Figure 4.1 and 4.2. As we can see, the performance varies as we change its value. It achieves the best results within the range of 100 to 200.

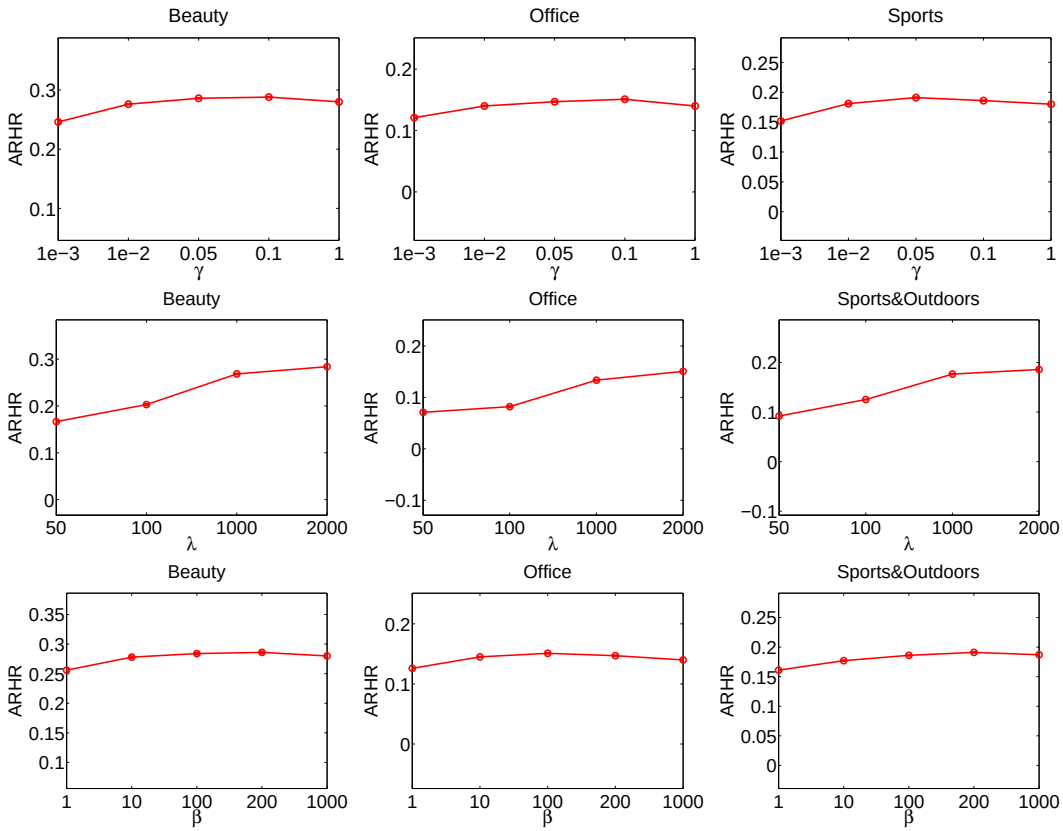

FIGURE 4.2: Parameter sensitivity analysis in terms of ARHR for γ, λ, β on Office, Beauty and Sports&Outdoor datasets.

4.5 Conclusion

In this paper, we presented a novel joint prediction approach that exploits both the conventional user-item recommendation matrix and the item-based side information in a complementary manner to generate Top-N recommendations for target users. We formulated the simultaneous user-item matrix recovering and item-based linear regression model training process as a joint low-rank constrained minimization problem, and derived an analytical closed-form solution for it. We conducted a set of experiments on five real world Amazon datasets with different recommendation items. The experimental results demonstrated that the proposed approach is effective and efficient in producing Top-N recommender systems, and outperforms the comparison methods.

CHAPTER 5

LEARNING DISCRIMINATIVE RECOMMENDATION SYSTEMS WITH SIDE INFORMATION

In this chapter, we propose a joint discriminative prediction model that exploits both the partially observed user-item recommendation (rating/purchase) matrix and the item-based side information such as item contents (product reviews and movie plots) to build top-N recommendation systems. The item information used in this work can be viewed as descriptions of the items and hence can produce feature representation vectors for the items. We propose a novel component that predicts the user-item recommendation (purchase) scores discriminatively from the item features with linear regression models, and integrate this component together with an aggregation model on the user-item matrix to jointly predict the unknown user-item recommendation (purchase) scores. The joint prediction of the recommendation scores from both information sources is expected to increase the performance of top-N recommendation systems. We formulate this joint prediction model as a convex minimization problem, which can be solved using a projected gradient descent algorithm. We evaluate the proposed approach by conducting experiments on several real world datasets. The experimental results show that the proposed approach can outperform both popular top-N recommendation methods that only use user-item matrix and state-of-the-art

algorithms that exploit additional side information. Moreover, our empirical study also reveals an interesting observation: the discriminative prediction component that predicts the recommendation scores from the item features (i.e., side information) contributes to most of the recommendation capacity of our joint prediction model, and the prediction component itself already outperform other comparison methods. This observation encourages researchers to explore prediction models for building top-N recommendation systems.

5.1 Introduction

Top-N recommendation systems automatically predict the missing recommendation scores over the set of product items for each consumer, and recommend a short list of N items with highest scores to each consumer. With the increasing popularity of online shopping, effective top-N commercial recommendation systems become of great importance in helping consumers to find their interested items and hence encouraging online item purchases. For example, a top-N recommendation system can effectively help consumers to find their potential interested products from Amazon.com and buy them online, and help users to find movies that best match their interests on Netflix. In such real world application domains, effective top-N recommendation systems can make significant commercial impacts.

Many algorithms have been developed in the literature to build top-N recommendation systems [63]. Most of them use a user-item rating matrix to build the recommendation models. A classical technique is collaborative filtering (CF) [68], which models the relationships between users and the correlations between items to identify new user-items relationship scores. Ranking methods have also demonstrated good performance for top-N recommendations [86, 77, 11, 3, 56]. In addition to CF and ranking, sparse aggregation methods that exploit linear correlations between items have been explored in a few works to improve top-N recommendation performance [50, 14, 28, 15]. Nevertheless, all these methods focus solely on the user-item rating

(or purchase) matrix, without exploring any additional information.

In many applications, in addition to the user-item rating/purchase matrix, item-based side information such as product reviews, book reviews, item comments, and movie plots can be easily collected from the Internet. This abundant item-based information can be used for recommendation systems. The work in [48] used books titles, reviews, and comments as item features; the work in [46] used item contents to enrich the sparse rating matrix. These recommendation systems however either only focus on user profiles or item contents without fully considering the user-item rating matrix or simply use item contents to pre-process the sparse rating matrix to produce small improvements. Moreover, they have not addressed the top-N recommendation problems. A recent work [51] incorporated item-based side information to improve top-N recommender systems, which however assumes that user-item matrix and item-based side information matrix are reproduced by the same linear aggregation model and does not exploit the discriminative power of the side information.

In this paper, we propose a joint discriminative prediction model that exploits both the partially observed user-item recommendation (rating/purchase) matrix and the item-based side information such as item contents (product reviews and movie plots) to build top-N recommendation systems. The item information used in this work can be viewed as descriptions of the items and hence can produce feature representation vectors for the items. We propose a novel component that predicts the user-item recommendation (purchase) scores discriminatively from the item features with linear regression models, and integrate this component together with an aggregation model on the user-item matrix to jointly predict the unknown user-item recommendation (purchase) scores. The joint prediction of the recommendation scores from both information sources is expected to increase the performance of top-N recommendation systems. We formulate this joint prediction model as a convex minimization problem, which can be solved using a projected gradient descent algorithm. We evaluate the proposed approach by conducting experiments on several real world datasets. The

experimental results show that the proposed approach can outperform both popular top-N recommendation methods that only use user-item matrix and state-of-the-art algorithms that exploit additional side information. Moreover, our empirical study also reveals an interesting observation: the discriminative prediction component that predicts the recommendation scores from the item features (i.e., side information) contributes to most of the recommendation capacity of our joint prediction model, and the prediction component itself already outperform other comparison methods. This observation encourages researchers to explore prediction models for building top-N recommendation systems.

5.2 Related Work

In this section, we review related works, including standard top-N recommendation systems and recent recommendation approaches that exploit item-based side information.

Collaborative filtering (CF) technology is widely used in top-N recommendation systems. CF methods explore the purchase history of all the users, relationships between the users and correlations between the items, and can be divided into two types, neighborhood-based CF and modelbased CF. Neighborhood-based CF methods exploit similarities between all the items and users to make recommendations [20, 82]. Model-based CF methods learn latent factors for users and items to reconstruct the user-item matrix. For example, the work in [17] developed a singular value decomposition (SVD) method to directly learn latent user factors and item factors. Though simple, this method has good performance for top-N recommendations. In [26, 75], a weighted regularized matrix factorization (WRMF) model is formulated to learn the latent factors based on implicit user-item feedbacks. The work in [73] proposed to learn the latent factors by directly maximizing the mean reciprocal rank. The work in [43] addressed top-N recommendation in dynamic situations. The authors of [60] proposed a Bayesian personalized ranking optimization method for personalized

ranking.

Weimer et.al [86] considered the top-N recommendation problem as a ranking problem, and used maximum margin matrix factorization to optimize ranking scores instead of rating scores. Following this work, a number of methods have been developed in this direction. The work in [77] addressed top-N item recommendation in the non-random missing situation by optimizing the metric of area under top-N curve. The work in [11] assumed that a user's preference is on a set of products instead of only one, and proposed to learn pairwise preferences over item-sets. The work in [3] focused on implicit feedbacks where preferences are given in the form of binary ratings, and proposed to optimize the ranking score within a margin maximization paradigm. The authors of [56] developed a large-scale optimization algorithm for ranking based matrix completion. In addition to collaborative filtering and ranking methods, a few sparse linear aggregation methods have recently been developed in the literature. The work in [50] proposed a novel sparse linear method (SLIM) to perform top-N recommendation. It learns a sparse aggregation coefficient matrix for items by solving an l_1-norm and l_2-norm regularized optimization problem, and aggregates the user purchase/rating profiles to produce the top-N recommendations. The work in [28] developed a factored item similarity model to tackle the situation where user-item matrix is very sparse and the SLIM method may fail to capture the item-item similarity. It learns the similarity matrix as the product of two low dimensional latent factor matrices and overcomes the sparsity problem. The work in [14] proposed a low rank SLIM method to improve SLIM, which adds a nuclear norm constraint to enforce a low rank aggregation matrix. The methods in [15] extend item pairwise similarities to higher orders, and capture more information with high order item correlations to better reconstruct the user-item matrix.

Instead of only focusing on the user-item matrix, a few works in the literature have incorporated contextual information [101, 73], user social networks [90], and information from linked open data [53] to improve recommendation performance. Moreover,

a number of methods have used more easily obtained item-based information to design recommendation systems. For example, the work in [48] used a Bayesian learning algorithm to build a content-based recommender system based on the item-based features, without using the user-item recommendation matrix. The work in [46] used item contents to pre-process the sparse rating matrix to overcome the sparsity of user-item purchase matrix. These two methods though are not designed for top-N recommendations, they verified the usefulness of item contents for item recommendation systems. More recently, the authors of [51] proposed a collective SLIM (cSLIM) method to incorporate item-based side information such as the reviews of the items into the SLIM to improve top-N recommender systems. This method assumes that user-item purchase matrix and itembased side information matrix are reproduced by the same sparse linear aggregation matrix, which is a very strong assumption in many real applications. A few recent works [4, 45, 87] have also exploited user reviews for building recommender systems. However these methods require the reviews to be directly associated with the user ratings, while our approach can exploit reviews from auxiliary resources that have no direct relationships with the target users.

5.3 Approach

In this section, we present a novel joint prediction model to produce personalized top-N recommendation systems. We assume an item purchase matrix $Y = \{0, 1\}^{n \times t}$ over n product items and t users is given, where $Y_{ij} = 1$ indicates an observed purchase relationship between the i-th product item and the j-th user while $Y_{ij} = 0$ indicates an unknown relationship between them. Note that an unobserved entry does not mean that the corresponding user is not interested in the particular product item. It only shows that the purchase record is not observed in the given user's past purchase history. The recommendation system will predict the unobserved entries and recommend to each user the top-N items with highest prediction scores from the unobserved item pool. We also assume the existence of item-based side information,

which could be in the form of product reviews for ecommerce applications or film plots for movie recommendations. Different from some literature works [4, 87] that exploited product reviews to build direct relationships between the considered users and products, we only require reviews provided as "side-information", which have no assumed connections to the target group of users. For simplicity, we assume the side-information will be presented as an item-feature matrix $X \in \mathbb{R}^{n \times d}$, whose each row contains features for a product item. We will use both X and Y to jointly recover an item-user recommendation matrix $\hat{Y}$, which contains the predicted recommendation scores for all product items. We then can produce a list of top-N most interesting items for each target user based on the predicted scores.

In the rest of the paper, we will use $\mathbf{1}$ to denote a column vector of all 1s, assuming the size of the column vector can be inferred from the context. We use I_n to denote an identity matrix with size n; use $||X||_F$ to denote the Frobenius norm of matrix X; use $X_{:j}$ to denote the j-th column and X_{ij} to denote the $(i; j)$-th entry of matrix X respectively.

5.3.1 A joint Prediction Model

We aim to predict the unobserved recommendation entries in the item-user matrix by exploiting both the given item-user matrix and the item-based side information matrix. The proposed joint recommendation prediction model hence has two components, a self-recovery component that recovers a full item-user recommendation matrix $\hat{Y}$ from Y and a linear regression prediction component that predicts $\hat{Y}$ from the item-based side information matrix X. We will first describe each component below and then present the joint model.

Self-Recovery of the Recommendation Matrix

Given the partially observed item-user recommendation matrix Y, we propose to automatically recover the full recommendation matrix $\hat{Y}$ and fill the recommendation

scores in the unobserved entries of Y by aggregating the existing item-user recommendations. Specifically, we assume an unobserved recommendation score for the j-th user on the i-th item, Y_{ij}, can be calculated as a nonnegative linear aggregation of the observed item recommendations for the j-th user; that is

$$\hat{Y}_{ij} = \mathbf{w}^T Y_{:j} \tag{5.1}$$

for $mathbfw \geq 0$. Such a linear recovery model can be built for any items, $\hat{Y}_{:j} = WY_{:,j}$. The principal behind the linear recovery models is that one assumes the items chosen by the same user have nonnegative linear correlation patterns. Assuming the same linear aggregation pattern for each item is shared across different users, we then have a matrix self-recovery formulation $\hat{Y} = WY$ with $W \geq 0$, where W is an $n \times n$ linear aggregation coefficient matrix that is shared across all the t users. The matrix self-recovery aims to recover the unobserved recommendation entries without affecting much the observed recommendation scores. Hence, to pursue a meaningful matrix self-recovery we should minimize the changes of the recovered matrix $\hat{Y}$ from the pre-given item-user recommendation matrix Y. Moreover, to avoid the trivial solution of setting W as an identity matrix, we enforce the diagonal of W to be zeros. This leads to the following matrix self-recovery problem:

$$\underset{W}{\text{minimize}} \quad ||Y - WY||_F^2 + \beta||Q||_F^2 \quad \text{s.t.} \quad W \geq 0, diag(W) = 0 \tag{5.2}$$

The constraints force each entry to be recovered from the other entries for the same user and hence encourage the statistical discovery of consistent item aggregation models across all the users. Moreover, with this constrained learning problem, the observed entries of the recommendation matrix Y may not be exactly maintained in the recovered matrix WY. To overcome this problem, we simply combine the original observed entries and the recovered unobserved entries together to form our final recovered item-user recommendation matrix $\hat{Y}$; that is we set $\hat{Y} = Y + (WY) \circ (1 - Y)$, where $\circ$ denotes Hadamard product operator.

Discriminative Prediction from the Side Information

Given the item-based side information expressed as an item feature matrix X, where each feature vector describes the properties of a given product item, we propose to learn a prediction model from the item-feature matrix X to predict the user ratings. Our intuition is that though the item-based side information is common to all the users, different parts of the item properties may correspond to the interests of different users. That is, different users may have different tastes, reflected by the recommendation scores or purchase activities, over same item, which can be related to the different features or feature subsets of the item. We hence assume the recommendation scores/purchase activities of each user over the set of product items can depend on the properties of items, and propose to predict the recommendation scores of each user over a product item from the feature vector of this product item. In particular, we can use a linear regression model, $f_j(x) = x^T q + b$, for the j-th user to predict his recommendation score y for the item described with feature vector x. Given the observed item-feature matrix X and the recovered item-user recommendation matrix $\hat{Y}$, the linear regression model f_j for the j-th user can be trained by minimizing a least squares regression loss $||X\mathbf{q} + b\mathbf{1} - \hat{Y}_{:,j}||^2$ over the model parameters $\mathbf{1}$ and b. For all the t users, we will then train t linear regression models by minimizing the following regularized least squares loss function

$$\underset{W}{\text{minimize}} \quad ||XQ + \mathbf{1}\mathbf{b}^T - \hat{Y}||_F^2 + \beta||Q||_F^2 \tag{5.3}$$

where $Q \in \mathbb{R}^{d \times t}$ and $\mathbf{b} \in \mathbb{R}^t$ are the parameters for the t linear regression models, the regularization term over Q is used to avoid over-fitting. When $\hat{Y}$ is unknown, these linear regression models will contribute to the prediction of the unknown entries of $\hat{Y}$. This prediction component is a novel component, which is fundamentally different from the previous methods in the literature by predicting the recommendation scores directly from the item-based side information.

Integration Model

With the bridge equation $\hat{Y} = Y + (WY) \circ (1 - Y)$, finally we can integrate the two components presented above, the recommendation matrix self-recovery component(i.e., Eq.5.2) and the linear regression prediction component(i.e., Eq.5.3), into the following joint matrix prediction model:

$$\begin{aligned} \underset{W,Q,b}{\text{minimize}} \quad & ||XQ + \mathbf{1}\mathbf{b}^T - Y + (WY) \circ (1 - Y)||_F^2 + \beta||Q||_F^2 + \gamma||Y - WY||_F^2 \\ \text{s.t.} \quad & W \geq 0, diag(W) = 0 \end{aligned} \tag{5.4}$$

where the trade-off parameter γ is introduced to balance the contribution of the two components. This joint model integrates information from both the partially observed item-user recommendation/purchase matrix Y and the item-feature matrix X, and is expected to further improve top-N recommendation systems.

In terms of learning a linear aggregation model to reconstruct the item-user matrix. our proposed method is related to the SLIM method and the cSLIM method developed in the literature. However, SLIM does not exploit side information, while cSLIM exploits the item-based side information with the same aggregation matrix W used for the matrix self-recovery. Our proposed approach is significantly different from these two methods in exploiting the item-based side information with linear regression models and integrating the two components via joint predictions.

5.3.2 Optimization Algorithm

The learning problem we formulated in Eq.5.4 is a joint convex minimization problem over two sets of parameters, the linear regression model parameters $\{Q, \mathbf{b}\}$ and the linear aggregation coefficient matrix W. For a fixed coefficient matrix W, the minimization problem in Eq.5.4 over Q and b becomes a standard linear regression problem, which has the following closed-form solution:

$$\mathbf{b} = \frac{1}{n}(Y + (WY) \circ (1 - Y) - XQ)^T\mathbf{1} \tag{5.5}$$

$$Q = (X^T H X + \beta I)^{-1} X^T H (Y + (WY) \circ (1 - Y)) \tag{5.6}$$

where $H = I - \frac{1}{n}\mathbf{1}\mathbf{1}^T$ is a centering matrix. By pluging these solutions bck into the objective function of Eq.5.4, we can reformulate Eq.5.4 equivalently into the following minimization problem over W:

$$\begin{aligned} \underset{W,Q,b}{\text{minimize}} \quad & ||B(Y + (WY) \circ (1 - Y))||_F^2 + \beta ||A(Y + (WY) \circ (1 - Y))||_F^2 + \gamma ||Y - WY||_F^2 \\ \text{s.t.} \quad & W \geq 0, diag(W) = 0 \end{aligned} \tag{5.7}$$

where $A = (X^T H X + \beta I)^{-1} X^T H$ and $B = H(XA - I)$. This remains to be a linear constrained convex minimization problem. We next develop a projected gradient descent algorithm to solve it.

Let $f(W)$ denote the objective function of Eq.5.7. The projected gradient descent algorithm will iteratively minimize $f(W)$ subjecting to the constraints. In each iteration, given the current W, the gradient of the objective function can be computed as:

$$\nabla f(W) = 2\gamma(WY - Y)Y^T + 2((1 - Y) \circ ((B^T B + \beta A^T A)(Y + (WY) \circ (1 - Y))))Y^T \tag{5.8}$$

With this gradient, it will first take a gradient descent step over W and then project the new W into the feasible set defined by the constraints. The overall algorithm is described in Algorithm 3: where the step size parameter η for the gradient descent is

Algorithm 3 Projected Gradient Descent Algorithm

Input: X, Y, parameters $\beta > 0$, $\gamma > 0$.
Initialize: W as zeros
while not converged **do**
1. find an optimal step size $\tau^\star \in [0, 1]$ with line search
2. gradient descent: $W = W - \tau^\star \nabla f(W)$
3. project onto feasible set: $W = max(W, 0), diag(W) = 0$

end while

determined using a standard backtracking line search:

$$\tau^\star = \underset{0 \leq \tau \leq 1}{\text{argmin}} \quad f(W - \tau \nabla f(W)) \tag{5.9}$$

5.4 Experiments

We conducted experiments on a few real world datasets. In this section, we will first describe the experimental setup and then present experimental results and discussions.

5.4.1 Experimental Setup

We used the real world Amazon user rating and product review data for five categories of products, **Beauty**, **Office**, **Sports& Outdoors**, **Health** and **Gourmet Foods**, to conduct experiments. For each category, the original dataset we downloaded contains the ratings of the users over the products and the product reviews. Following [50, 51], we converted the rating values to 1s and produced implicit feedback matrices to use. The initial implicit feedback matrices are extremely sparse. We selected the top ten thousand items based on the purchasing times for each item, and then selected the top four thousands of users to use. We further preprocessed the datasets by selecting users with at least 3 purchases and filtering the items without any purchases. Finally we obtained five item-user matrices.

For each item-user purchase (feedback) matrix produced, we used the product reviews as the item-based side information. The reviews are given in plain text and we preprocessed each item review in the following way. We first extracted unigram features from the review articles, and then removed the stopping words and selected the top 5000 frequent unigram features as the item features. Finally each product item is represented as a bag-of-word feature vector in terms of these 5000 unigram features. We used the term-frequency feature values as the item-feature data.

The statistic summary of the produced datasets is presented in Table 5.1. For each dataset, we present the number of items (#items), the number of users (#users), the number of transactions (#trns), the average number of transactions for each item (#rsize), the average number of transactions for each user (#csize) and the density value (#trns/(#users×# items)).

Table 5.1: Statistic summary for the experimental datasets.

dataset	#items	#users	#trns	rsize	csize	density
Beauty	2275	1965	11051	4.86	5.62	0.247%
Office	2107	1888	8307	3.94	4.40	0.209%
Sports	1789	1324	7947	4.44	6.000	0.336%
Gourmet Foods	3343	2813	15264	5.43	4.57	0.162%
Health	2899	2518	14530	5.77	5.01	0.199%

Comparison Methods

We compared our proposed method with the following methods for top-N recommendation systems:

(1) Pure Singular Value Decomposition (**PureSVD**) [17], This model-based collaborative filtering method performs singular value decomposition directly on the item-user purchase matrix, and reconstructs the matrix with the top subsets of singular vectors.

(2) Weighted Regularized Matrix Factorization(**WRMF**) [26]. WRMF extracts the latent factors for the users and items by performing weighted regularized matrix factorization. The reconstructed item-user matrix based on the extracted factors is then used for top-N recommendations.

(3) Sparse Linear Model (**SLIM**) [50]. SLIM learns an item-item aggregation sparse coefficient matrix by minimizing a constrained reconstruction loss. It uses the coefficient matrix to reconstruct the item-user purchase matrix.

(4) Collective Sparse Linear Model (**cSLIM**) [51]. cSLIM incorporates side information into the SLIM model by enforcing the item-feature matrix to share the same aggregation matrix with the item-user matrix.

(5) Inductive Matrix Completion(**IMC**) [49]. IMC is a state-of-art matrix completion method that exploits side information. We applied it to produce top-N recommendation systems.

(6) Side Information SVD(**Side-SVD**) [99]. We introduce Side-SVD in Chapter 4. We already showed that Side-SVD outperforms state-of-the-art models in Chapter 4. In this chapter, we also compare our proposed method with our formal model.

Evaluation Metrics

We evaluated our proposed method and the comparison methods using 5-fold Leave-One-Out-Cross-Validation (LOOCV). For each fold, the dataset is divided into a training set and a testing set: We randomly chose one transaction for each user and placed it to the test set, and the rest is used as training set. The training set is used to perform training. Then a ranked list of top N items are generated from the un-observed items for each user according to their scores in the reconstructed item-user matrix. The test results are produced by comparing the selected top-N items for each user to his observed test set item. If there is a match between the test set item and the top N items, it is counted as one hit. The default N value used in our experiments is 10. We used two measurements to evaluate the test results: Hit Rate(HR) and the Average Reciprocal Hit Rate (ARHR) [20], which are defined as

$$HR = \frac{\#\text{hits}}{\#\text{users}}, \quad ARHR = \frac{1}{\#\text{users}} \sum_{i=1}^{\#\text{hits}} \frac{1}{p_i}, \tag{5.10}$$

where #users is the total number of users, #hits is the number of hits of the top-N recommendations across all users, and p_i is the position of the test item in the ranked recommendation list. ARHR is a weighted version of HR, which takes the ranking position of the test item in the top-N recommendation list into account.

5.4.2 Comparison Results

We compared the six comparison methods ($pureSVD, WRMF, SLIM, cSLIM, IMC$ and $Side-SVD$) with the proposed approach on the five Amazon datasets using the evaluation metrics HR and ARHR. The average results and standard deviations for

all the methods are reported in Table 5.2. We can see that within the three methods that do not exploit side information, SLIM outperforms the other two methods, $pureSVD$ and $WRMF$. By using the item-based side information, $cSLIM$ consistently outperforms all the three methods that do not exploit side information across all the datasets, and IMC outperforms the three methods on four out of the five datasets. This suggests that side information is useful, which is consistent with the results reported in [51]. However, the improvements produced by $cSLIM$ and IMC over the other three methods, especially over $SLIM$, are relatively small. Our proposed approach on the other hand consistently outperforms all the five comparison methods across all the datasets. The improvements achieved by our approach are notably large. For example, on $Office$, the proposed approach outperforms $cSLIM$ by 6.4% and 4.0% in terms of HR and ARHR respectively, and outperforms IMC by 6.7% and 3.9% in terms of HR and ARHR respectively. Theses results demonstrate that our proposed approach provides an effective mechanism for exploiting the item-based side information to improve top-N recommendation systems.

From the table we can also see that both $Side-SVD$ and the proposed model outperform other methods. Generally, the proposed model performs better than $Side-SVD$. On $Sports\&Outdoors$, $Side-SVD$ is slightly better than the proposed model. But on $Beauty$, $Office$, $GourmetFoods$ and $Health$, the proposed model performs better than $Side-SVD$ on both evaluation measurements.

5.4.3 Results with Different N Values

To investigate how much the N value affects the performance of each method, we further conducted sets of experiments with different N values to compare the performance of the five methods for top-N recommendations. We tested a set of different N values such as $N \in \{5, 10, 15, 20\}$. For each N value, the same experimental setting as described above is used. The average results and the standard deviations in terms of HR and ARHR are reported in Figure 5.1. From the figures, we can see that by

increasing the value N, the top-N recommendation performance in terms of HR and $ARHR$ in general improves for all the comparison approaches. This is reasonable since larger N values allow using more candidates to match each test item. Nevertheless, our proposed approach consistently outperforms all the other four comparison methods across all the N values on $Beauty$, $Office$, $Sports\&Outdoors$, $GourmetFoods$ and $Health$. Our proposed method is only slightly worse than $Side-SVD$ on $Sports\&Outdoors$ dataset. This further validated the effectiveness of the proposed approach for top-N recommendations.

5.4.4 Component-wise Study

The proposed approach integrates two components, the self-recovery component and the linear regression prediction component, together to jointly perform item-user matrix reconstruction. How does each component contribute to the final top-N recommendations? Which component is more important? To answer these questions, we conducted another set of experiments to compare the proposed approach to its two individual components, each of which uses only one type of information, side information or the purchase matrix. The linear prediction component based on side information can be obtained by simply $\gamma = 0$ in the proposed objective function Eq.5.4 to drop the self-recovery component. The self-recovery component on item-user purchase matrix can be obtained by dropping the linear regression models from Eq.5.4. This component can be viewed as a variant of the $SLIM$ method without the regularization terms on W. We compared the two components with the proposed joint model on all the five datasets. The comparison performance are showed in Figure 5.2 and the exact results are reported Table 5.3. We can see that the performance of the prediction component with side information outperforms the self-recovery component with only purchase matrix with large margins across all the datasets. By comparing the results in both Table 5.2 and Table 5.3, we can see that the linear prediction component even consistently outperforms the most effective comparison methods, $cSLIM$

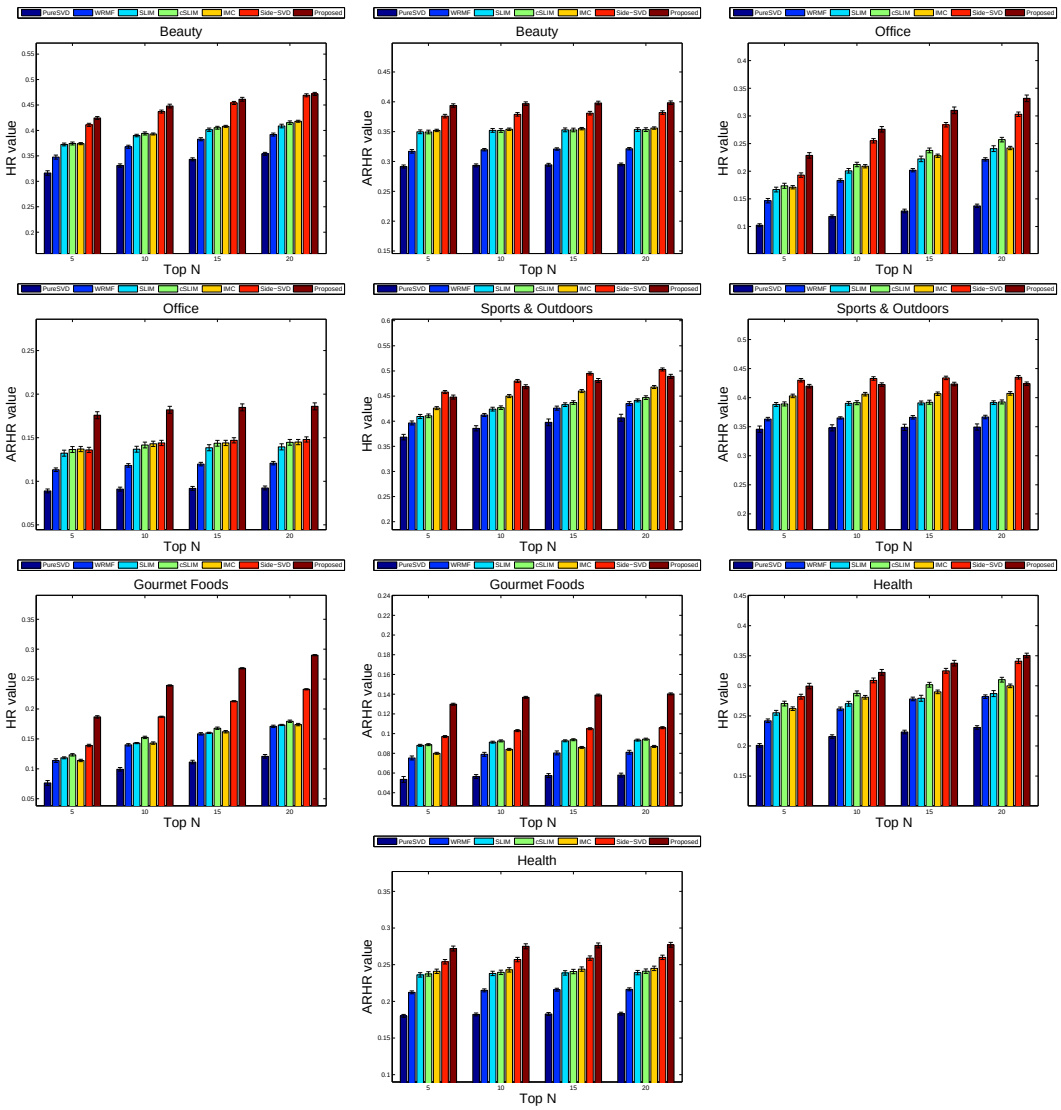

FIGURE 5.1: Comparison results of all the methods with respect to different N values ($N \in \{5, 10, 15, 20\}$) for top-N recommendation systems on $Office, Beauty, Sports\&Outdoor, GourmetFoods$ and $Health$.

and IMC, across all the datasets. This suggests that our novel prediction component that uses linear regression models to predict the item-user recommendation scores from the item-based side information is very effective. Nevertheless, our proposed approach that integrates the two components together consistently outperforms each individual component, which suggests the two components contain complementary information and our proposed model can effectively capture such information to improve top-N recommendation performance.

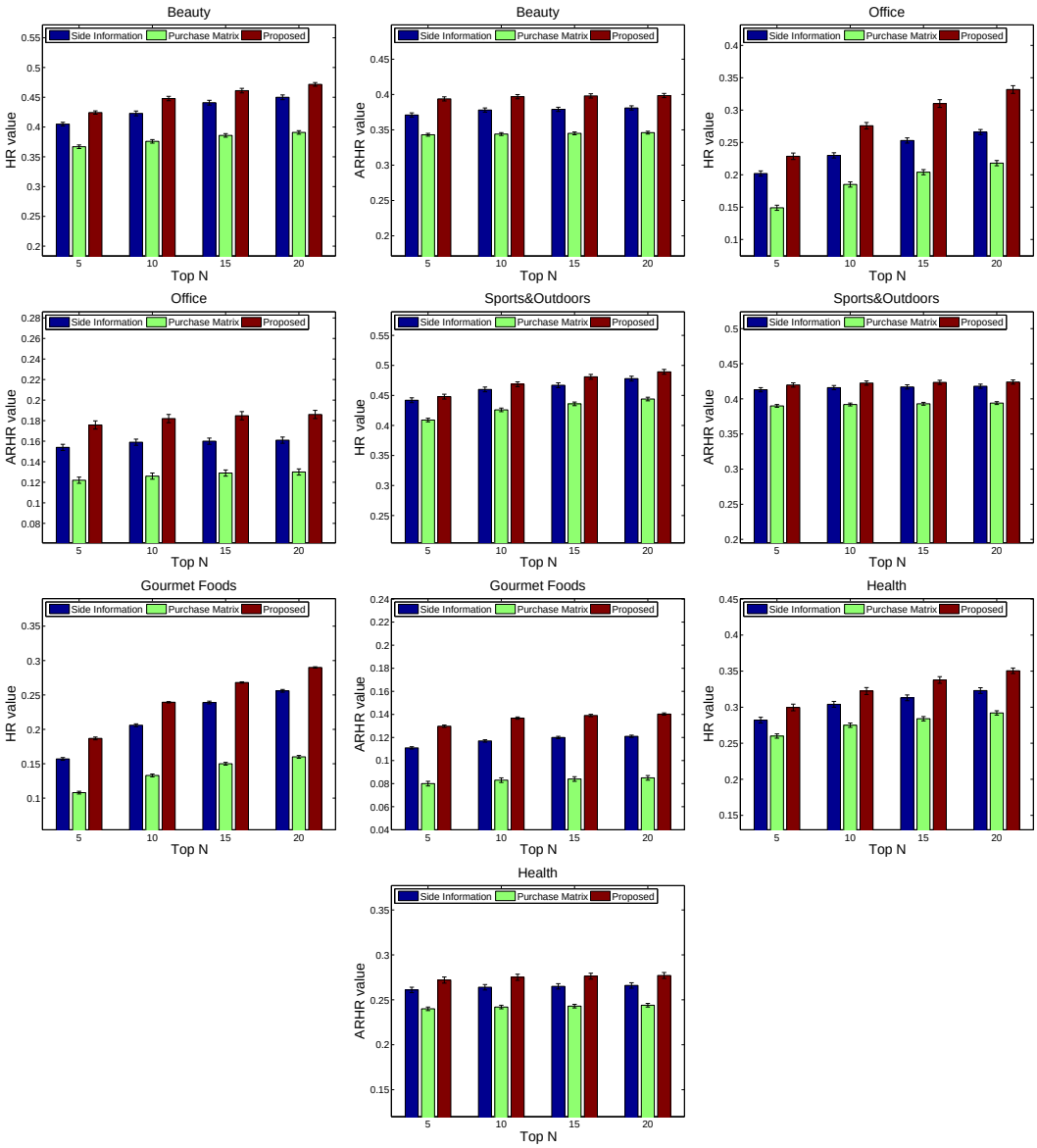

FIGURE 5.2: Evaluation Results of only using item side information, only using purchase matrix and the proposed method with respect to different values of N in top-N (N=5, 10, 15, 20) recommendation systems on Office, Beauty, Sports&Outdoor, Gourmet Foods and Health.

5.4.5 Parameters Sensitivity Study

Our proposed approach(Eq 5.4)has 2 parameters:

β and γ need to be selected. Parameter β is the linear regression regularization parameter and γ controls the weight of item-user purchase matrix self-recovery. The larger $\gamma$is, the more purchase matrix contributes. In this section we analysis the sensitivity for them. The experimental settings are the same as before. When we test the sensitivity of one parameter, we fix another by using the selected value in Table 5.2. We tested different β values from $\{1, 10, 100, 200, 1000\}$ while is selected from $\{1e-3, 1e-2, 0.05, 1e-1, 1, 10\}$. We present the average test results over five runs on all the 5 datasets. Experimental Results are showed in Figure 5.3 for β and Figure 5.4 for γ. From the figures we can see that for all the datasets, HR and ARHR precision get the best results when γ is around 0.05 and 0.1. This means item side information and purchase matrix get to a balance. But when γ is bigger than 1, the accuracy will be worse than the results when $\gamma$is 0. For parameter β, we an see that HR and ARHR are relatively stable. Results will increase when β increases until

the value gets up to 200. When β is bigger than 200, the recommendation quality decreases.

5.5 Conclusion

In this paper, we proposed a novel joint discriminative prediction model for personalized top-N recommendations. The proposed model integrates information from the standard item-user purchase matrix with a linear aggregation matrix and from the auxiliary item-feature matrix with linear regression models to predict the unobserved recommendation entries. We formulated this method as a joint convex optimization problem and solved it using a projected gradient descent algorithm. We conducted experiments on five real world Amazon datasets, and the proposed approach outperforms a number of top-N recommendation methods developed in the literature.

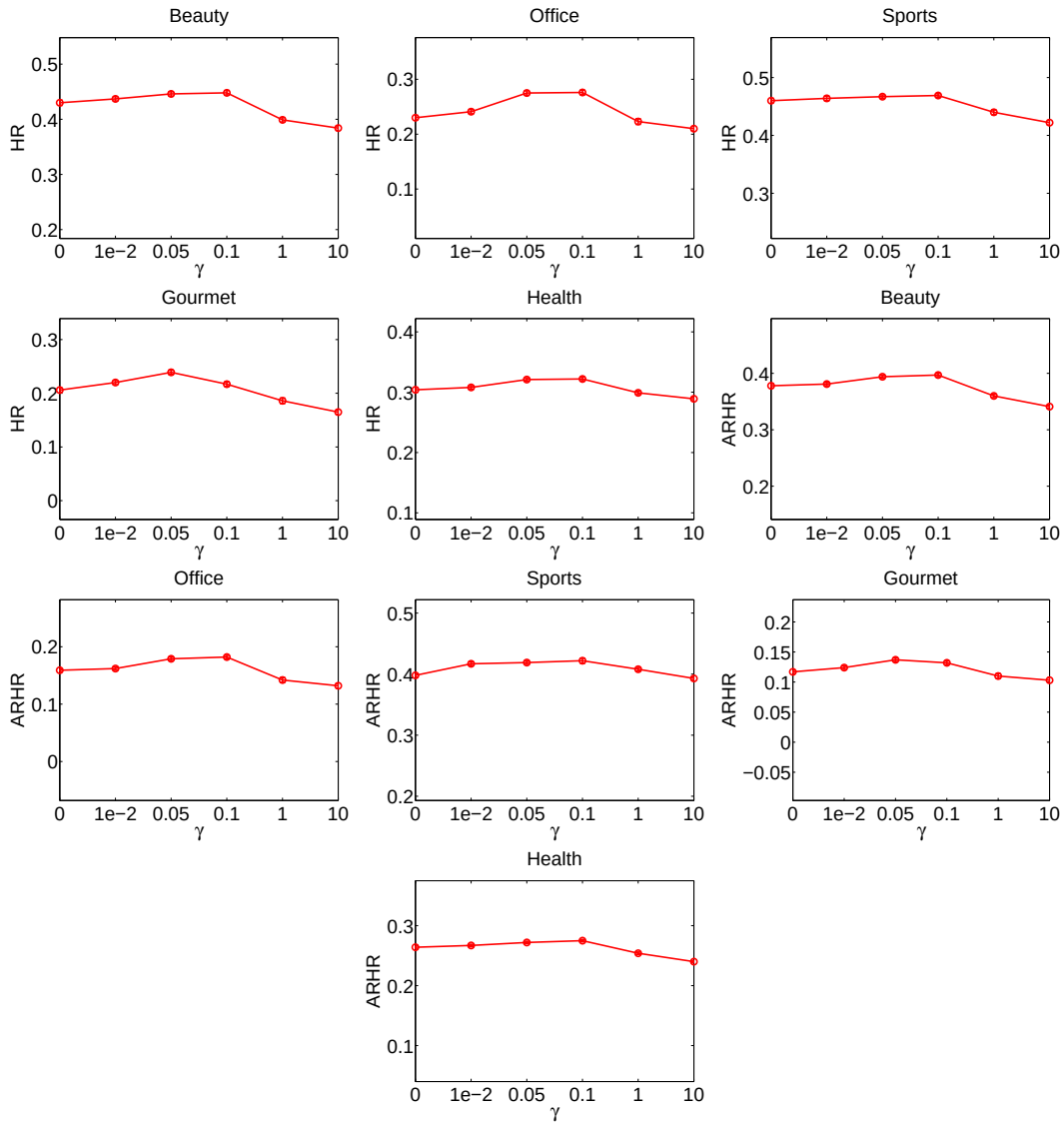

FIGURE 5.3: Sensitivity Analysis of HR and ARHR scores for parameter γ on the five datasets.

Moreover, the experimental results also demonstrated the efficacy of the novel linear regression prediction component of the proposed model, which suggests it is effective to exploit the item-based side information in a discriminative way and encourages researchers to explore prediction models for building top-N recommendation systems.

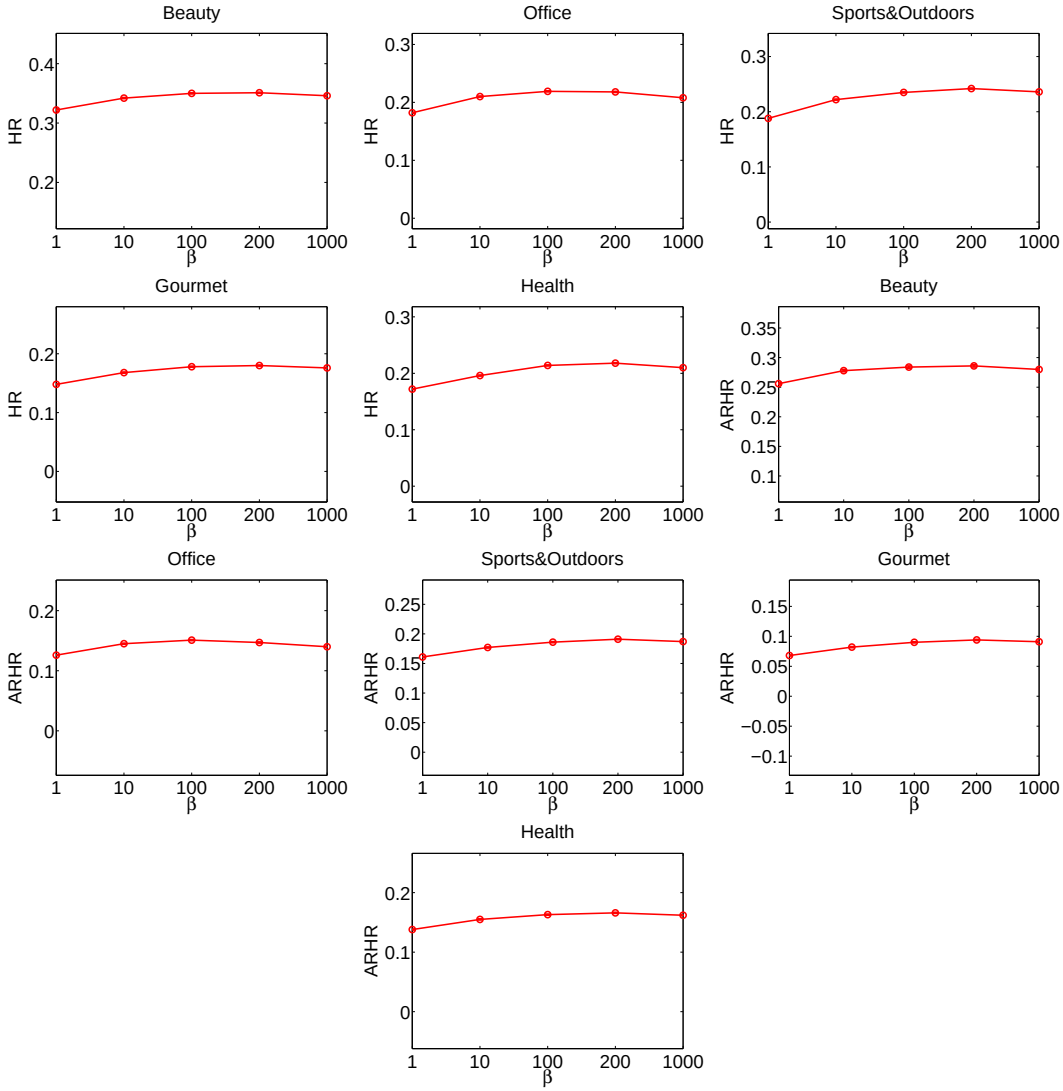

FIGURE 5.4: Sensitivity Analysis of HR and ARHR scores for parameter β on the five datasets.

Table 5.2: Comparison of Performance of Top-N Recommendation Algorithms

method	**Beauty**				
	params			HR(%)	ARHR(%)
PureSVD	100	-	-	33.1 ± 0.3	29.3 ± 0.2
WRMF	10	50	-	36.8 ± 0.3	31.9 ± 0.2
SLIM	10	0.01	-	39.0 ± 0.2	35.2 ± 0.003
cSLIM	10	0.01	1e-6	39.4 ± 0.3	35.2 ± 0.3
IMC	200	1	-	39.3 ± 0.2	35.4 ± 0.2
Side-SVD	100	0.05	1000	43.7 ± 0.3	37.9 ± 0.3
Proposed	200	0.1	-	$\mathbf{44.8 \pm 0.4}$	$\mathbf{39.7 \pm 0.3}$
method	**Office**				
	params			HR(%)	ARHR(%)
PureSVD	100	-	-	11.8 ± 0.3	9.1 ± 0.2
WRMF	10	50	-	18.3 ± 0.3	11.8 ± 0.2
SLIM	10	0.01	-	20.1 ± 0.4	13.7 ± 0.3
cSLIM	1e-6	10	0.01	21.2 ± 0.4	14.2 ± 0.3
IMC	200	0.001	-	20.9 ± 0.3	14.3 ± 0.2
Side-SVD	200	0.01	1000	25.6 ± 0.4	14.4 ± 0.3
Proposed	200	0.1	-	$\mathbf{27.6 \pm 0.5}$	$\mathbf{18.2 \pm 0.4}$
method	**Sports&Outdoors**				
	params			HR(%)	ARHR(%)
PureSVD	200	-	-	38.6 ± 0.5	34.8 ± 0.5
WRMF	10	50	-	41.2 ± 0.3	36.5 ± 0.4
SLIM	1	0.01	-	42.4 ± 0.4	39.0 ± 0.3
cSLIM	1e-7	1	0.001	42.7 ± 0.4	39.1 ± 0.3
IMC	200	1	-	45.0 ± 0.4	40.6 ± 0.3
Side-SVD	200	0.1	1000	$\mathbf{48.0 \pm 0.3}$	$\mathbf{43.3 \pm 0.3}$
Proposed	200	0.05	-	46.9 ± 0.4	42.3 ± 0.4
method	**Gourmet Foods**				
	params			HR(%)	ARHR(%)
PureSVD	100	-	-	9.9 ± 0.3	5.6 ± 0.2
WRMF	10	50	-	14.0 ± 0.2	7.9 ± 0.2
SLIM	10	0.01	-	14.3 ± 0.1	9.1 ± 0.1
cSLIM	1e-6	10	0.01	15.3 ± 0.2	9.3 ± 0.1
IMC	200	0.001	-	14.3 ± 0.2	8.4 ± 0.1
Side-SVD	200	0.05	1000	18.7 ± 0.2	10.3 ± 0.1
Proposed	200	0.05	-	$\mathbf{23.9 \pm 0.1}$	$\mathbf{13.7 \pm 0.1}$
method	**Health**				
	params			HR(%)	ARHR(%)
PureSVD	100	-	-	21.6 ± 0.3	18.2 ± 0.3
WRMF	10	50	-	26.2 ± 0.3	21.5 ± 0.3
SLIM	10	0.01	-	27.0 ± 0.4	23.8 ± 0.3
cSLIM	1e-6	1	0.01	28.7 ± 0.4	24.0 ± 0.3
IMC	200	0.001	-	28.1 ± 0.2	24.3 ± 0.2
Side-SVD	200	0.05	1000	30.9 ± 0.3	25.7 ± 0.2
Proposed	200	0.1	-	$\mathbf{32.2 \pm 0.4}$	$\mathbf{27.5 \pm 0.3}$

The params columns contain the parameter setting for each approach. $PureSVD$ has one parameter f, indicating the number of latent factors. $WRMF$ has two parameters, regularization trade-off parameter λ and the latent factor dimension f. $SLIM$ has two parameters, thel_2 and l_1 norm regularization parameters β and λ. Beyond β and λ, cSLIM has another parameter, the side information weight parameter α. The IMC has two parameters, hidden dimensison f and regularization parameter λ. $Side-SVD$ has three paramters, regularization parameterβ, trade-off parameter γ and rank parameter λ. The proposed method has two trade-off parameters, β and γ.

Table 5.3: Comparison of the individual components of the proposed approach and the integrated model in terms of top-N recommendation performance.

method	**Beauty**	
	HR(%)	ARHR(%)
Linear Prediction	42.3 ± 0.3	37.8 ± 0.3
Self-Recovery	37.6 ± 0.3	34.4 ± 0.2
Joint Model	$\mathbf{44.8 \pm 0.4}$	$\mathbf{39.7 \pm 0.3}$
method	**Office**	
	HR(%)	ARHR(%)
Linear Prediction	23.0 ± 0.4	15.9 ± 0.3
Self-Recovery	18.5 ± 0.3	12.6 ± 0.3
Joint Model	$\mathbf{27.6 \pm 0.5}$	$\mathbf{18.2 \pm 0.4}$
method	**Sports&Outdoors**	
	HR(%)	ARHR(%)
Linear Prediction	46.0 ± 0.4	41.6 ± 0.3
Self-Recovery	42.6 ± 0.3	39.0 ± 0.2
Joint Model	$\mathbf{46.9 \pm 0.4}$	$\mathbf{42.3 \pm 0.4}$
method	**Gourmet Foods**	
	HR(%)	ARHR(%)
Linear Prediction	20.6 ± 0.2	11.7 ± 0.2
Self-Recovery	13.3 ± 0.2	8.3 ± 0.2
Joint Model	$\mathbf{23.9 \pm 0.1}$	$\mathbf{13.7 \pm 0.1}$
method	**Health**	
	HR(%)	ARHR(%)
Linear Prediction	30.4 ± 0.4	26.4 ± 0.3
Self-Recovery	27.5 ± 0.3	24.2 ± 0.2
Joint Model	$\mathbf{32.2 \pm 0.4}$	$\mathbf{27.5 \pm 0.3}$

CHAPTER 6

CONCLUSION

There are different kinds of challenges when designing top-N recommender systems on implicit feedbacks. We try to learn and solve two of these challenges: designing a better loss function and using side information in a better way. In particular, we propose three novel top-N Recommender Systems with Implicit Feedbacks.

6.1 Loss function Design

The first approach uses a heterogeneous loss to improved recommendation predictions. In this approach, a combined loss function is introduced. We combine point-wise consistency and pair-wise ranking loss functions together. We try to capture these two structures in the recovery feedback matrix. In order to optimize our loss function, projected stochastic gradient descent method(SGD) is used. We conduct experiments to compare the proposed method with state-of-the-art methods and it outperforms all the other models. In addition, we compare the proposed method with model only uses point-wise consistency and model only uses pair-wise ranking consistency. The experimental results proved that the heterogeneous loss enhances both of the loss function and performs better than them. In addition, when we use the projected gradient descent method to learn the model, we find that by using only a limited

number of triples we can reach very good performance.

6.2 Side Information incorporation

User/item side information can represent users and items and these information can also enhance the performance. In the thesis, we also explore the method to incorporate side information into the recommender systems.

6.2.1 Closed Form Solution with Rank Constraint

We first use item-user feedback matrix and item side information feature matrix together to learn a jointly model. Besides, we also set a low-rank constraint to our jointly model. We derived an analytical closed-form solution for it. The items will be recommended based on the learned recovery matrix. We use Amazon real-world dataset to conduct experiments. The experiment results show that our proposed model can use the side information efficiently to improve the performance of top-N recommender systems. The proposed method also outperforms other methods which also use item side information.

6.2.2 Convex Minimization Solution by using Gradient Descent

In the third model, similar to the former model, we also use item side information. This time we do not have low-rank constraint. But the learned item-item aggregation matrix should be non-negative and in order to avoid trivial solution, the diagonal entries should be zero. We use projected gradient descent method to optimize the objective function. We conducted experiments on Amazon datasets, and the proposed approach outperforms top-N recommendation methods in the literature. Moreover, the experimental results also demonstrated the efficacy of the linear regression prediction component of the proposed model. It encourage others to exploit similar way to learn recommender systems by using side information.

6.3 Future Work

In this section we give two directions of the future work for top-N recommender systems.

6.3.1 Large-Scale recommender systems

Recommender systems are mainly used for applications which have huge amount items and users. In real-world recommender system applications, user size and item size are very large. For example, Netflix and Amazon has million users and items. In real-world applications, recommender systems should be designed to deal with such big data problem. In [67, 61], people try to design large-scale recommender systems. In [67], the authors try to split users into clusters and neighborhood algorithm is applied in every cluster. In [61], Rendle et.al proposed an on-line update algorithm which is proper for huge datasets.

Generally, algorithms should be designed suitable for large-scale problems and the models should have low time complexity or can be learned by parallel computing.

6.3.2 Deep Learning for recommender systems

Deep learning is a very hot topic in the field of machine learning. People also try to use deep neural networks to learn recommender systems. In [64, 81, 70], the authors use deep neural networks to build collaborative filtering models. In [52, 16], deep neural networks are used to learn new representations for music and videos and the learned representations are used in recommender systems. New deep neural network based representations of items still need to be explored. New deep structures for collaborative filtering also need to be designed.

BIBLIOGRAPHY

[1] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, 2008.

[2] H. J. Ahn. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Inf. Sci.*, 178(1), Jan. 2008.

[3] F. Aiolli. Convex auc optimization for top-n recommendation with implicit feedback. In *Proceedings of the 8th ACM Conference on Recommender Systems*, 2014.

[4] Y. Bao, H. Fang, and J. Zhang. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[5] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, Aug. 2013.

[6] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of the KDD Cup Workshop 2007*, 2007.

[7] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research (JMLR)*, 3:993–1022, 2003.

[8] M. Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and Its Applications*, 415(1):20–30, 2006.

[9] K. Chen, T. Chen, G. Zheng, O. Jin, E. Yao, and Y. Yu. Collaborative personalized tweet recommendation. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2012.

[10] L. Chen, G. Chen, and F. Wang. Recommender systems based on user reviews: The state of the art. *User Modeling and User-Adapted Interaction*, 25(2), June 2015.

[11] L. Chen and W. Pan. Cofiset: Collaborative filtering via learning pairwise preferences over item-sets. In *Proceedings of the SIAM International Conference on Data Mining (SDM 2013)*, 2013.

[12] M. Chen, Z. E. Xu, K. Q. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

[13] Z. Chen, M. Chen, K. Q. Weinberger, and W. Zhang. Marginalized denoising for link prediction and multi-label learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[14] Y. Cheng, L. Yin, and Y. Yu. Lorslim: Low rank sparse linear methods for top-n recommendations. In *ICDM*, 2014.

[15] E. Christakopoulou. Moving beyond linearity and independence in top-n recommender systems. In *Proceedings of the 8th ACM Conference on Recommender Systems*, 2014.

[16] P. Covington, J. Adams, and E. Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, 2016.

[17] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys)*, 2010.

[18] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*, 2007.

[19] L. de Campos, J. Fernández-Luna, J. Huete, and M. Rueda-Morales. Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks. *International Journal of Approximate Reasoning*, 51(7):785–799, 2010.

[20] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[21] S. Friedland and A. Torokhti. Generalized rank-constrained matrix approximation. *SIAM Journal on Matrix Analysis and Applications*, 29:656–659, 2007.

[22] H. Gao, J. Tang, X. Hu, and H. Liu. Content-aware point of interest recommendation on location-based social networks. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[23] C. A. Gomez-Uribe and N. Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4), Dec. 2015.

[24] A. Gunawardana and C. Meek. A unified approach to building hybrid recommender systems. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, 2009.

[25] T. Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval (SIGIR)*, 2003.

[26] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 2008.

[27] Z. Huang, H. Chen, and D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1), Jan. 2004.

[28] S. Kabbur, X. Ning, and G. Karypis. FISM: factored item similarity models for top-n recommender systems. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2013.

[29] P. Knees and M. Schedl. A survey of music similarity and recommendation from music context data. *ACM Trans. Multimedia Comput. Commun. Appl.*, 10(1), Dec. 2013.

[30] P. Knees and M. Schedl. Music retrieval and recommendation: A tutorial overview. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015.

[31] Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2008.

[32] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*. 2012.

[34] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521, 2015.

[35] H. Li, Y. Ge, R. Hong, and H. Zhu. Point-of-interest recommendations: Learning potential check-ins from friends. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[36] L. Li and T. Li. News recommendation via hypergraph learning: Encapsulation of user behavior and news content. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, 2013.

[37] L. Li, D. Wang, T. Li, D. Knox, and B. Padmanabhan. Scene: A scalable two-stage personalized news recommendation system. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2011.

[38] S. Li, J. Kawale, and Y. Fu. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015.

[39] D. Lian, C. Zhao, X. Xie, G. Sun, E. Chen, and Y. Rui. Geomf: Joint geographical modeling and matrix factorization for point-of-interest recommendation. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014.

[40] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1), Jan. 2003.

[41] B. Liu, Y. Fu, Z. Yao, and H. Xiong. Learning geographical preferences for point-of-interest recommendation. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013.

[42] J. Liu, P. Dolan, and E. R. Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th International Conference on Intelligent User Interfaces*, 2010.

[43] X. Liu and K. Aberer. Towards a dynamic top-n recommendation framework. In *Proceedings of the 8th ACM Conference on Recommender Systems*, 2014.

[44] 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*, 2011.

[45] J. McAuley and J. Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, 2013.

[46] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*, 2002.

[47] A. K. Menon and C. Elkan. Link prediction via matrix factorization. In *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part II*, 2011.

[48] R. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the ACM Conference on Digital Libraries*, 2000.

[49] N. Natarajan and I. S. Dhillon. Inductive matrix completion for predicting gene-disease associations. *Bioinformatics*, 30(12), 2014.

[50] X. Ning and G. Karypis. SLIM: sparse linear methods for top-n recommender systems. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 2011.

[51] X. Ning and G. Karypis. Sparse linear methods with side information for top-n recommendations. In *Proceedings of the ACM Conference on Recommender Systems*, 2012.

[52] A. v. d. Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, 2013.

[53] V. Ostuni, T. Di Noia, E. Di Sciascio, and R. Mirizzi. Top-n recommendations from implicit feedback leveraging linked open data. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, 2013.

[54] R. Pan, Y. Zhou, B. Cao, N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 2008.

[55] S. Park and D. Pennock. Applying collaborative filtering techniques to movie search for better ranking and browsing. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2007.

[56] S. Park and D. Pennock. Preference completion: Large-scale collaborative ranking from pairwise comparisons. In *roceedings of the 32th International Conference on Machine Learning(ICML2015)*, 2015.

[57] S.-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In *Proceedings of the Third ACM Conference on Recommender Systems*, 2009.

[58] M. Pazzani and D. Billsus. The adaptive web. chapter Content-based Recommendation Systems, pages 325–341. Springer-Verlag, 2007.

[59] 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.

[60] 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*, 2009.

[61] S. Rendle and L. Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, 2008.

[62] J. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2005.

[63] F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. In *Recommender Systems Handbook*, pages 1–35. 2011.

[64] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2007.

[65] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2Nd ACM Conference on Electronic Commerce*, 2000.

[66] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the International Conference on World Wide Web (WWW)*, 2001.

[67] B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering, 2002.

[68] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web*, pages 291–324. Springer-Verlag, 2007.

[69] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002.

[70] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, 2015.

[71] 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*, 2013.

[72] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver. Tfmap: Optimizing map for top-n context-aware recommendation. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2012.

[73] 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 ACM Conference on Recommender Systems (RecSys)*, 2012.

[74] D. Shin, S. Cetintas, K.-C. Lee, and I. S. Dhillon. Tumblr blog recommendation with boosted inductive matrix completion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015.

[75] V. Sindhwani, S. Bucak, J. Hu, and A. Mojsilovic. One-class matrix completion with low-density factorizations. In *Proceedings of the IEEE International Conference on Data Mining*, 2010.

[76] D. Sondermann. Best approximate solutions to matrix equations under rank restrictions. *Statistical Papers*, 27:57–66, 1986.

[77] H. Steck. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010.

[78] D. Stern, R. Herbrich, and T. Graepel. Matchbox: Large scale online bayesian recommendations. In *Proceedings of the International Conference on World Wide Web (WWW)*, 2009.

[79] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009.

[80] A. Theodoridis, C. Kotropoulos, and Y. Panagakis. Music recommendation using hypergraphs and group sparsity. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pages 56–60, 2013.

[81] T. T. Truyen, D. Q. Phung, and S. Venkatesh. Ordinal boltzmann machines for collaborative filtering. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2009.

[82] K. Verstrepen and B. Goethals. Unifying nearest neighbors collaborative filtering. In *Proceedings of the 8th ACM Conference on Recommender Systems*, 2014.

[83] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[84] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011.

[85] 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*, 2015.

[86] M. Weimer, A. Karatzoglou, Q. Le, and A. Smola. COFI$^{\text{RANK}}$ - maximum margin matrix factorization for collaborative ranking. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[87] Y. Xu, W. Lam, and T. Lin. Collaborative filtering incorporating review text and co-clusters of hidden user communities and item groups. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, 2014.

[88] R. Yan, M. Lapata, and X. Li. Tweet recommendation with graph co-ranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, 2012.

[89] D. Yang, T. Chen, W. Zhang, Q. Lu, and Y. Yu. Local implicit feedback mining for music recommendation. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, 2012.

[90] X. Yang, H. Steck, Y. Guo, and Y. Liu. On top-k recommendation using social networks. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, 2012.

[91] M. Ye, X. Liu, and W.-C. Lee. Exploring social influence for recommendation: A generative model approach. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2012.

[92] X. Yu, X. Ren, Y. Sun, B. Sturt, U. Khandelwal, Q. Gu, B. Norick, and J. Han. Recommendation in heterogeneous information networks with implicit user feedback. In *Proceedings of the 7th ACM Conference on Recommender Systems*, 2013.

[93] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann. Time-aware point-of-interest recommendation. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2013.

[94] H. Yun, P. Raman, and S. V. N. Vishwanathan. Ranking via robust binary classification. In *Advances in Neural Information Processing Systems*, 2014.

[95] H. Yun, H.-F. Yu, C.-J. Hsieh, S. Vishwanathan, and I. S.Dhillon. Nomad: Non-locking, stochastic multi-machine algorithm for asynchronous and decentralized matrix completion. In *International Conference on Very Large Data Bases*, 2014.

[96] J. Zhang, C. Wang, P. S. Yu, and J. Wang. Learning latent friendship propagation networks with interest awareness for link prediction. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, 2013.

[97] F. Zhao and Y. Guo. Improving top-n recommendation with heterogeneous loss. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016.

[98] F. Zhao and Y. Guo. Learning discriminative recommendation systems with side information. In *Technical Report*, 2017.

[99] F. Zhao, M. Xiao, and Y. Guo. Predictive collaborative filtering with side information. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016.

[100] Y. Zheng, B. Mobasher, and R. Burke. Cslim: Contextual slim recommendation algorithms. In *Proceedings of the 8th ACM Conference on Recommender Systems*, 2014.

[101] Y. Zheng, B. Mobasher, and R. Burke. Deviation-based contextual slim recommenders. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, 2014.

[102] K. Zhou, S.-H. Yang, and H. Zha. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2011.

[103] M. Zinkevich, M. Weimer, A. J. Smola, and L. Li. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems*, 2011.