MATRIX COMPLETION WITH SIDE INFORMATION FOR EFFECTIVE RECOMMENDATION

By

Iman Barjasteh

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering - Doctor of Philosophy

2016

ABSTRACT

MATRIX COMPLETION WITH SIDE INFORMATION FOR EFFECTIVE RECOMMENDATION

By

Iman Barjasteh

The massive number of online choices, e-commerce items, and related data available on the web makes it profoundly challenging for users to extract insightful information that can help them decide on what to select among a vast multitude of choices. In recent years, recommender systems have played an important role in reducing the overwhelming impact of such information overload. In particular, a specific form of recommender system, which is known as collaborative filtering, is the most popular approach to building these systems, and it has been successfully employed in many applications. More importantly, the matrix completion paradigm provides an appealing solution to the collaborative filtering problem in recommendation systems. However, collaborative filtering based approaches perform poorly for sparse data and specifically for the so-called cold start users.

Recently, there has been an upsurge interest in utilizing other rich sources of side information about items/users to compensate for the insufficiency of rating information. Such information is of more importance to be aggregated when a single view of the data is sparse or even incomplete. Due to the advent and popularity of online social networks and e-commerce websites, many different types of side information are available that can be taken into account in addition to traditional rating matrices in order to improve the recommendation.

The overarching goal of this thesis is to propose a novel and general algorithmic framework based on matrix factorization that simultaneously exploits the similarity information among users and items to alleviate the data sparsity issue and specifically the cold-start problem. We extend matrix factorization and propose a model that takes into account the side information as well as the rating matrix. Therefore, by modeling different types of side information, such as social or trust/distrust relationships between users and meta-data about items, as a constraint similarity/dissimilarity graph, we propose an effective recommendation framework that is able to boost the recommendation accuracy and overcome the challenges in existing recommendation systems such as cold-start users/items and data sparsity problems. The proposed modeling framework is capable of performing both rating and link prediction.

Based on the proposed framework, a key objective of this thesis is to develop novel algorithms and derive theoretical guarantees for their performance. The algorithms we developed so far have been experimentally evaluated and compared against existing state-of-the-art methods on real life datasets (such as MovieLens, NIPS, Epinions and etc.). Our experimental results show that our proposed modeling framework and related algorithms achieve substantial quality gains when compared to with existing methods. Our experimental results also illustrate how our framework and algorithms can overcome the shortcomings of other state-of-the-art recommendation techniques.

To my parents, Leila & Karim

ACKNOWLEDGMENTS

First and foremost, I must offer my most heartfelt thank you to my advisor Dr. Radha, for his guidance, encouragement, and inspiring supervision throughout the course of this research work. His motivation to reach higher taught me how I should not have any limits for my dreams. He taught me how I should earn my PhD rather than getting it and his approach made me earn my PhD rather than getting it. It?s hard to express how thankful I am for his unwavering support over the last years.

I would like to take on this opportunity to thank my thesis committee members Dr. Esfahanian, Dr. Deb and Dr. Biswas, who have accommodated my timing constraints despite their busy schedules, and provided me with precious feedback for the presentation of the results, in both written and oral form.

This work would not be possible without the wonderful guidance from two friends, Dr. Rana Forsati and Dr. Mehrdad Mahdavi, who helped me a lot throughout my PhD. I greatly enjoyed working with them and massive thanks to them for sharing their interest and expertise.

To my friends, you all have stood by me both personally and professionally throughout my journey through graduate school. I would not have gotten through it without you all, nor would my time at Michigan State been as much fun. Thank you all.

Last but definitely not least, I want to express my deepest gratitude to my beloved parents, Leila and Karim, and my dearest siblings. Thank you for encouraging intellectual curiosity throughout my entire life. That along with your love and support were instrumental in finishing this very challenging endeavour.

PREFACE

"We are leaving the era of search and entering one of discovery!"

– CNN Money, the race to create a smart Google

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Due to the popularity and exponential growth of e-commerce websites (e.g. Amazon, eBay) and online streaming websites (e.g. Netflix, Hulu), a compelling demand has been created for efficient recommender systems to guide users toward items of their interests (e.g. products, books, movies). Recommender systems seek to predict the rating that a user would give to an item and further try to suggest items that are most appealing to the users. Recently, recommender systems have received a considerable amount of attention and have been the main focus of many research studies [2].

Content-based filtering (CB) and collaborative filtering (CF) are well-known examples of recommendation approaches. As demonstrated by its performance in the KDD Cup [29] and Netflix competition [13], the most successful recommendation technique used is collaborative filtering. This technique exploits the users' opinions (e.g., movie ratings) and/or purchasing (e.g., watching, reading) history in order to extract a set of interesting items for each user. In factorization based CF methods, both users and items are mapped into a latent feature space based on observed ratings that are later used to make predictions.

In spark contrast to CF, in collaborating ranking (CR) models [58, 22, 119, 25], where the goal is to rank the unrated items in the order of relevance to the user, the popular ranking measures such as as discounted cumulative gain (DCG), normalized discounted cumulative gain (NDCG), and average precision (AP) [54] are often employed to collaboratively learn a ranking model for the latent features.

Recent studies have demonstrated that CR models lead to significantly higher ranking accuracy over their traditional CF counterparts that optimize rating prediction. This is important considering the fact that what we really care in recommendation is not the actual values of ratings, but the order of items to be recommended to a specific user. Therefore, the error measures such as RMSE are often hopelessly insufficient, as their place equal emphasis on all the ratings. Among ranking models, the methods that mainly concentrate on the *top of the list* have received a considerable amount of attention, due to the higher probability of examining the top portion of the list of recommendations by users. Therefore, the introduction of ranking metrics such as push norm or infinite norm [99, 5, 22, 59], sparked a widespread interest in CR models and has been proven to be more effective in practice [112, 22].

On the other hand, the hardness of gathering the complete structure of networked data such as social networks, biological networks, and publication networks is one of the challenging problems that most of network analysis applications are currently facing. However, there are currently very few techniques for working with incomplete network data. Therefore, it is tempting to observe a partial sample of a large network, and based on that sample, infer what the rest of the network looks like, a problem known as the *network completion* or *survey sampling*. Specifically, the sample is chosen by randomly selecting a fixed number of vertices, and for each we are allowed to observe all or a uniformly sampled subset of edges it is incident with. Study of the problem of recovering the topology of an undirected network by observing only a partially observed random subsample of it with enough edges also received a considerable amount of attention and have been the main focus of many research studies [55].

This general problem arises in a number of different settings in information retrieval, social network analysis, and computational biology [7, 86]. For instance, in online social networks such as Facebook, Myspace, and Twitter, given the large adoption of these networks, inferring the full network form a small sample of links between users or finding the implicit social networking structure between them is a challenging problem [86].

Despite significant improvements in recommendation systems and network completion approaches, and in particular factorization based methods, these systems suffer from a few inherent limitations and weaknesses such as *data sparsity* and *cold-start* problems. Specifically, in many real world applications, the rating data or the observed networks are very sparse (i.e., most users do not rate most items typically resulting in a very sparse rating matrix) or for a subset of users or items the rating data is entirely missing (knows as cold-start user and cold-start item problem, respectively [103]). To address the difficulties associated the latter issues, there has been an active line of research during the past decade and a variety of techniques have been proposed [87, 100, 109, 121, 64, 114, 122, 77, 92, 120, 62].

The studies in the literature have approached the cold-start problem from many different angles, but they commonly exploit the auxiliary information about the users and items in addition to the rating data that are usually available (see e.g. [106] for a more recent survey). By leveraging multiple sources of information one can potentially bridge the gap between existing items (or users) and new (cold start) items (or users) to mitigate the cold-start problem.

The main motivation behind these techniques stems from the observation that other sources of data can be used to reveal more information about the underlying patterns between users and items and thus complement the rating data. For instance, knowing the underlying social connections (friends, family, etc.) between users can give us a better understanding of the sources of influence on a user's decision. Subsequently, the availability of auxiliary information such as users' profile information [2], social context (trust and distrust relations) of users [35], information embedded in the review text [64], and features of items [37] provide tangible benefits to the recommender. Hence, an intriguing research question, which is the main focus of this thesis is:

How can side information about users and items be effectively incorporated in factorization methods to overcome the cold-start problem?

1.1 Objectives and Proposed Framework

In this section, two major category of algorithms will be highlighted.

1.1.1 Collaborative Filtering Framework

Although there are many different collaborative filtering algorithms in the literature to augment matrix factorization with side information, such as shared subspace learning [102] and kernelized matrix factorization [122], the dominant paradigm in existing methods is to perform

(a) *completion* of partially observed matrix (e.g. rating matrix)

(b) *transduction* of knowledge from observed entries (e.g. existing ratings) to cold-start entries (e.g. items/users) simultaneously.

While these methods are able to generate good results in practice, they have notable drawbacks:

1. these methods propagate the completion and transduction errors repetitively and in an uncontrolled way

2. many of state-of-art algorithms are usually application based, e.g. see [114, 63], and do not offer a general framework for alleviating cold-start problems.

The overarching goal of this thesis is to answer the above question by proposing an efficient matrix factorization approach using similarity information. In fact, not only we propose a general framework for dealing with cold-start problems, but also we study a somewhat more general problem where both cold-start users and cold-start items are present simultaneously and address these two challenges simultaneously. In particular, by considering the drawbacks of the existing methods, we propose a two-stage algorithm that *decouples* the completion and transduction stages. First, we exclude the cold-start items and users and complete the rating matrix. Next, we transduct the knowledge to cold-start users/items using the recovered sub-matrix in addition to the available side information about the users and items. Hence, there is no error propagation of completion and transduction. Interestingly, beyond just dealing with cold-start problem, the proposed algorithm also provides an effective way to exploit the side information about users (or items) to mitigate the data sparsity and compensate for the lack of rating data.

1.1.2 Collaborative Ranking Method I

Although CR models for recommender systems has been studied extensively and some progress has been made, however, the state of affairs remains unsettled: the issue of handling *cold-start* items in ranking models and coping with *not missing at random* assumption of ratings are elusive open issues. First, in many real world applications, the rating data are very sparse (e.g., the density of the data is around 1% for many publicly available datasets) or for a subset of users or items the rating data is entirely missing (knows as cold-start user and cold-start item problem, respectively) [103]. Second, collaborative filtering and ranking models rely on the critical assumption that the missing ratings are sampled uniformly at random. However, in many real applications of recommender systems, this assumption is not believed to hold, as invariably some users are more active than others and some items are rated by many people while others are rarely rated [111]. These issues have been investigated in factorization based methods, nonetheless, it is not straightforward to adapt them to CR models and are left open [22]. Motivated by these challenges, we ask the following fundamental question in the context of collaborative ranking models:

> *Is it possible to effectively learn a collaborative ranking model in the presence of cold-start items/users that is robust to the sampling of observed ratings?*

In this thesis, we give an affirmative answer to the above question. In particular, we introduce a *semi-supervised collaborative ranking* model, dubbed $\texttt{S}^2\texttt{COR}$, by leveraging side information about *both observed and missing ratings* in collaboratively learning the ranking model. In the learned model, unrated items are conservatively pushed after the relevant and before the irrelevant items in the ranked list of items for each individual user. This crucial difference greatly boosts the performance and limits the bias caused by learning only from sparse non-random observed ratings. We also introduce a graph regularization method to exploit the side information about users to overcome the cold-start users problem. In summary, the key features of $\texttt{S}^2\texttt{COR}$ are:

- Inspired by recent developments in ranking at top [99, 5, **?**], the proposed model is a collaborative ranking model that primarily focuses on the top of the recommendation list for each user. Moreover, in stark contrast to pairwise ranking models which have quadratic dependency on the number of items, the proposed ranking model has a linear dependency on the number of items, making it suitable for large-scale recommendation.

- It leverages side information about items with both observed and missing ratings while collaboratively learning the ranking model, which enables it to effectively incorporate the available side information in knowledge transduction.
- By incorporating the unrated items in ranking, it limits the bias caused by learning solely based on the observed ratings and consequently deals with the not missing at random issue of ratings.
- It is also able to leverage the similarity information between users based on a graph regularization method to make high quality recommendations for users with few ratings or cold-start users without an rating information.

1.1.3 Collaborative Ranking Method II

Recently, online networks where two opposite kind of relationships can occur have become common. For instance, the `Epinions` [44], an e-commerce site for reviewing and rating products, allows users to evaluate others based on the quality of their reviews, and make trust and distrust relations with them. Similar patterns can be found in online communities such as `Slashdot` in which millions of users post news and comment daily and are capable of tagging other users as friends/foes or fans/freaks. Additionally, users on `Wikipedia` can vote for or against the nomination of others to adminship [16]. When more users issuing distrust statements, the more important it will become to exploit this new information source in recommender systems.

It is acknowledged that along with the trust relationships, also distrust can play an important role in boosting the accuracy of recommendations [117, 116, 35]. Recently, some attempts have been made to explicitly incorporate the distrust relations in recommendation process [44, 70, 116]. This demonstrated that the recommender systems can benefit from the proper incorporation of distrust relations in social networks. However, despite these positive results, there are some unique challenges involved in distrust-enhanced recommender systems. In particular, it has proven challenging to model distrust propagation in a manner which is both logically consistent and psychologically plausible. Furthermore, the naive modeling of distrust as negative trust raises a number of challenges- both algorithmic and philosophical. Finally, it is an open challenge how to best incorporate trust and distrust relations in model-based methods, e.g., matrix factorization, simultaneously. In memory based recommender systems, the simultaneous exploitation of trust and distrust has been investigated in [118, 115].

An attempt to simultaneously exploit trust and distrust relations in factorization based method has been made very recently in [35]. In particular, a ranking model was proposed to rank the latent features of users, from the perspective of individual users, that respects their social context. Despite the encouraging improvements that has been brought by simultaneous exploitation of trust and distrust relations, the proposed algorithm suffers from two issues. First, as the number of constraint triplets imposed from social regularization of latent features can increase cubically in the number of users in the social network, it is computationally difficult to make their idea scalable to large social graphs. Second, the algorithm proposed in [35] only considers the trusted and distrusted friends of each user in order to regularize the latent features. Thus, it ignores the neutral friends – the users who has no relation to the user. As a result, the neutral friends might appear before the trusted friends in the ranked list. Therefore neutral friends might become more influential than trusted friends and impact the recommendation negatively.

In this thesis we describe and analyze a fairly general and flexible method to learn and infer from rating data, along with trust and distrust relationships between users. The main building block of our proposed algorithm, dubbed `PushTrust`, is an efficient algorithm to rank the latent features of users by leveraging their social context. In ranking of latent features, we wish to make sure that the top portion of the list includes the trusted friends and the distrusted friends are pushed to the bottom of list. Also, we would like the neutral friends appear after trusted and before distrusted friends in the list. Compared to the method proposed in [35], the key features of the `PushTrust` algorithm are: (i) its quadratic time complexity in the number of users, and (ii) its ability to take the neutral friends of users into the consideration in regularizing the latent features of users. Thorough experiments on the Epinions dataset demonstrate the merits of the proposed algorithm in boosting the quality of recommendations and dealing with data sparsity and cold-start users problems.

1.2 Organization of the Thesis

The remainder of the thesis is organized as follows. Chapter 2 lays out the foundation for the rest of the thesis. In particular, we provide a survey of some of the background materials and state-of-the-art algorithms.

In part **??** we talk about the theory of our proposed algorithms and their analysis. In chapter 3 we propose our first decoupled matrix factorization based algorithm (`RECT`) and its analysis. In chapter 4 we talk about our semi-supervised collaborative ranking algorithm and its analysis ($\texttt{S}^2\texttt{COR}$). In chapter 5, we propose a new collaborative ranking algorithm that utilized the trust/distrust relations between users.

In part **??** we talk about three different major real life applications of our proposed algorithm. Chapter 6 focuses on the first application of `RECT`, which the cold-start recommendation. Chapter 7 is also a cold-start application of algorithm $\texttt{S}^2\texttt{COR}$, which provides a ranking of items for users. Chapter 8 is a cold-start application of algorithm `PushTrust`. Chapter 9 is another application of `RECT` for network completion problem.

Part **??** focuses on the conclusions (chapter 10) of this thesis and the future works (chapter **??**).

Chapter 2

Preliminaries and Related Works

2.1 Preliminaries

In this chapter we formally define the recommendation problem and specifically talk about the matrix completion based problems different classes of existing methods. Then we discuss the approaches for evaluating recommender systems. It is worth mentioning that we explain the problem as a recommendation problem but the proposed framework is a general framework for matrix completion problem.

In recommendation systems we assume that there is a set of n users $\mathcal{U} = \{u_1, \ldots, u_n\}$ and a set of m items $\mathcal{I} = \{i_1, \ldots, i_m\}$ where each user u_i expresses opinions about a subset of items. In this thesis, we assume opinions are expressed through an explicit numeric rating (e.g., scale from one to five), but other rating methods such as hyperlink clicks are possible as well. We are mainly interested in recommending a set of items for an active user such that the user has not rated these items before. The rating information is summarized in an $n \times m$ matrix $\mathbf{R} \in \mathbb{R}^{n \times m}, 1 \leq i \leq n, 1 \leq j \leq m$ where the rows correspond to the users and the columns correspond to the items and (p, q)-th entry is the rate given by user u_p to the item i_q. We note that the rating matrix is partially observed and it is sparse in most cases.

Two general tasks can be defined for a recommender, one is rating prediction and the other one is top-N recommendations. In the following we formally define these two problems:

Rating Prediction

Given a user $u \in \mathcal{U}$ and an item $i \in \mathcal{I}$, for which $r_{u,i}$ is unknown, compute the predicted rating of u on item i, $r_{u,i}$ using the rating matrix $\mathbf{R}$ and the social network $\mathbf{S}$.

Top-N Prediction

Given a user $u \in \mathcal{U}$ and the rating matrix $\mathbf{R}$, recommend the top N most related items to user u that has not rated yet.

The main focus of this thesis is on both rating prediction and top-N recommendation. TheUsing a rating prediction system, one can infer a top-N recommender by simply ranking all items ccording to their predicted rating. This approach is obviously not efficient but effective. There are more sophisticated ways to perform top-N recommendation that we discuss later in this thesis.

An efficient and effective approach for recommender systems is to factorize the user-item rating matrix $\mathbf{R}$ by a multiplicative of k-rank matrices $\mathbf{R} \approx \mathbf{U}\mathbf{V}^\top$, where $\mathbf{U} \in \mathbb{R}^{n \times k}$ and $\mathbf{V} \in \mathbb{R}^{m \times k}$ utilize the factorized user-specific and item-specific matrices, respectively, to make further missing data prediction. There are two basic formulations to solve this problem: these are optimization approaches (see e.g., [97, 65, 71, 58]) and probabilistic approaches [80]. Let $\Omega_\mathbf{R}$ be the set of observed ratings in the user-item matrix $\mathbf{R} \in \mathbb{R}^{n \times m}$, i.e., $\Omega_\mathbf{R} = \{(i,j) \in [n] \times [m] : \mathbf{R}_{ij} \text{ has been observed}\}$. In optimization based matrix factorization, the goal is to learn the latent matrices $\mathbf{U}$ and $\mathbf{V}$ by solving the following optimization problem:

2.2 Rich Side Information of Users and Items

As mentioned in the Introduction, we look beyond the U-I matrix to include two types of additional information that is considered useful for improving the recommendations: rich side information about users and items, and information about the situation in which users interact (e.g., rate, click, or purchase) with items. The main focus of this thesis is incorporating the rich side information available to better complete the rating matrix.

The range of sources of side information on users and items stretching beyond the U-I matrix is quite broad and varied. One of the most common side information sources is attribute information. User attributes may include information such as the user?s gender, age, and hobbies. Item attributes reflect properties of the item, such as category or content. However, two sources of information that have recently increased in importance in the recommender system research are social networks and user-contributed information. In the remainder of this subsection, we discuss these information sources in more detail.

2.2.1 Social Networks

The emergence of social networks has impacted a wide range of research disciplines in the past years, and recommender systems are no exception. Specifically to the recommender system area, social networks introduce information in the form of user-user relationships, which may be particularly useful for improving the quality of recommendation. In general, the social relationships between users can be either directed or undirected. Social trust and distrust relationships are among the most studied of directed social relationships. The trust/distrust relationship can usually be described as an asymmetric user-user graph/matrix, which indicates whether one user trusts/distrusts another. Another important directed social relationship is follow, exemplified by the follow relationship used by Twitter. The follow relationship is similar to the trust relationship in that it reflects the appreciation of one user (the follower) for another (the followee). In the case of Twitter, the follower receives the followee?s microblog posts. The canonical example of an undirected social relationship is friendship, as used in Facebook. Friendship can be represented as a symmetric user-user graph/matrix, which encodes whether two users are friends of each other. It is also possible to extract more complex relationships, such as tie strength and similarity, between users in a social network by analyzing the link structure and the common patterns of user behavior. Recommender system algorithms that attempt to leverage social relationships, both directed and undirected, apply the assumption that users who stand in a positive relationship with each other may also share similar interests, as will be discussed later.

2.2.2 User-Contributed Information

User-contributed information has become widely available in most recommender systems, and its volume has grown steadily since the introduction of Web 2.0 technology. Strictly speaking, the user ratings contained in the U-I matrix can be considered as one type of user-contributed information as well. Here, we introduce four types of user-contributed information that go beyond the U-I matrix: tags, geotags, multimedia content, and free-text reviews and comments, which are increasingly used in recommender systems [106]:

- **Tags:** Tags are short textual labels that users assign to items. Tags differ from conventional category labels in that users can assign them freely?that is, they are not constrained by a preset list. Users tag items for different reasons. Some tags describe item properties, and others express how users feel about an item. Tags are recognized as an information source that can be highly beneficial for improving the performance of recommender systems. In addition to exploiting tags for recommending items, personalized tag recommendation has also become an active research topic, which, however, falls outside the scope of this survey.

- **Geotags:** Since GPS positioning has become a standard functionality of mobile digital devices (e.g., cell phones or digital cameras), location information becomes abundant in social media sites, such as photo and video sharing sites and microblogging sites. The location information of an item in those sites is usually in the form of geotags?that is, explicit latitude and longitude coordinates. As reflected in its name, geotag can be regarded as a special class of tags that are particularly used for geographical positions. In a photo sharing site, geotags of a photo may indicate that the uploader of the photo has been to that location (or nearby). Similarly, the geotags of a user?s tweets may be used to trace the location of the user. Due to the availability of large quantities of geotags, remarkable progress has been achieved in both the research on exploiting location information for improving recommendation and on facilitating location/travel recommendation.

- **Multimedia Content:** Social media sites, such as Flickr and YouTube, have facilitated their users for uploading and sharing multimedia content (e.g., images and videos). The user-contributed multimedia content serves as another type of side information for the online users. For example, the categories of the photos in a user's album may reflect what kinds of items she likes to see. The particular type of videos that a user usually posts may indicate her interest in a particular item. Such information can be exploited for more elaborately modeling the user interests and thus contributing to content recommendation [81, 11, 6].

- **Reviews and Comments:** Last but not least, moving beyond tags and geotags, free text reviews and comments that are published by users online are another important source of community contributions. They are valuable not only because of their semantics but also because of the sentiment dimension. For this reason, it is not surprising that reviews and comments have been exploited as a type of side information for improving recommender systems.

2.3 Related Works

Before delving into the detailed description and analysis of the proposed framework, we would like to draw connections to, and put our work in context of, some of the more recent work. The different approaches can be roughly divided into the following categories.

2.3.1 Matrix Completion and Cold-start Recommendation

Naive methods. The first category includes *naive* algorithms that try to recommend items to users based on their popularity [87], or based on a random selection [66]. Naive methods treat all cold-start users/items in a same way and assume that all users/items contribute the same to recommendations. This has the effect of tremendously reducing the accuracy due to a lack of any filtering.

Warm-start methods. For cold-start scenarios, since there is no historical data available for either users or items, warm-start methods either ask users to rate a set of items or import their preferences from another source of auxiliary information in order to expand the user profile [66, 121, 24]. In particular, these methods explicitly ask a new user to rate k representative items in order to regulate the taste of new user for dealing with cold-star user problems. Similarly, a new item can forced to be rated by k representative users in cold-star item scenarios.

Liu et al. [66] proposed a representative-based matrix factorization model which is able to adjust itself with new users/items by learning their parameters. Sun et al. [113] proposed an algorithm for rapid profiling of new users, which is guided by a decision tree that focuses on increasing the recommendation accuracy while minimizing the user interaction efforts. Zhou et al. [121] proposed a new user preference elicitation strategy to learn the profile of users and items. More recently, Contardo et al. [24] proposed a representative-learning formalism for cold-start user recommendation, which is based on an inductive method whose principle is to code ratings on items in the latent representation space. The drawback of these approaches is that the cold-start users will be forced to rate number of representative items, which increases the user interactions.

Feature combination. As mentioned earlier, in recent years there has been an upsurge of interest in utilizing other rich sources of side information about items and users along with the rating matrix to increase the accuracy of the recommendation and dealing with cold-start challenges [106]. Therefore, *feature combination* approaches, as the third category of cold-start recommendations, became quite appealing. These methods employ and combine features related to users (e.g. profile) or items (e.g. metadata) to increase the accuracy while minimizing the user interactions. Feature combination methods can be used as a single model regardless of the types available information.

The recent advances in matrix factorization methods suggest *subspace sharing* or *matrix co-factorization* can effectively incorporate side information [68, 43, 48, 47], where several matrices (rating and side information matrices) are simultaneously decomposed, sharing some factor matrices. The kernelized matrix factorization approach studied by Zhou et al. [122], incorporates the auxiliary information into the matrix factorization to assess the similarity of latent features using the available similarity matrices. Saveski et al.'s [102] matrix factorization method, in a common low-dimensional space, collectively decomposes the content and the collaborative matrices. We also note that several recent studies extend the maximum margin matrix factorization, which was developed for collaborative filtering [110], to incorporate side information [85, 1]. These studies aim to overcome the data sparsity problem by reducing the number of sampled entries and is different from our work.

An alternative way for feature combination is to map the available auxiliary features to the latent features of the factorization model. Elbadrawy et al. [30] proposed an approach that learns a mapping function to transform the feature vectors of items into their latent space. Ganter et al. [37] also proposed a matrix factorization model that maps the features to the latent features of the model. Boltzmann machines [46] and aspect models [103] are other factor models that utilize side information for cold-start recommendation. Another family of feature combination methods includes those that rely on explicit features of items/users to compute the similarity between items/users by extracting different key features such as textual similarity or semantic similarity [64, 114].

Boltzmann machines are another types of latent factor models, which model the joint distribution of a set of binary variables through their pairwise interactions [46]. Gunawardana et al. [46] built a recommender system that combines the individual user information with their pairwise similarity across the users and learns weights for features, which is the reflection of how well each user's action can be predicted. Aspect model, as another family of latent factor models, has also been employed to tackle cold-start problems. Aspect model is a latent class variable framework that is constructed for contingency table smoothing. Schein et al. [103] employed aspect model, to propose a probabilistic model combing content and collaborative data.

Another family of feature combination methods includes those that that rely on explicit features of items/users. These approaches compute the similarity between items/users by extracting different key features such as textual similarity or semantic similarity. Ling et al. [64] introduced an unified model by combining the ratings with the textual features of the reviews using a topic modeling approach. Trevisiol et al. [114] studied the news browsing behavior of users and they built a BrowsGraph and employed the structural and temporal properties of them to propose a news recommender for users to deal with cold-start problem[1].

Model combination. Finally, model combination methods combine the outputs of different recommeders by various strategies such as: voting [90], weighting the output score of recommenders [23], switching between recommenders [14], or filtering and re-ranking the outputs [17]. The drawback of these approaches is that they require building different and separate recommender systems to each source of used information and combining their outputs [46]. Park et al. [87] cast the cold-start recommendation as a regression problem and applied a combination of all information of users/items to incorporate the side information.

Transduction with matrix completion. In matrix completion the objective is to fill out the missing entries of a matrix based on the observed ones. Early work on matrix completion, also referred to as maximum margin matrix factorization [110] was developed for collaborative filtering. Theoretical studies show that, it is sufficient to perfectly recover a matrix [122, 77, 92].

We also note that several recent studies extend the maximum margin matrix factorization which was developed for collaborative filtering [110] to incorporate side information [32, 85]. We also note that matrix completion with infinite dimensional side information was exploited in [1].

Finally, we note that although various hybrid methods such as factorization machines [94],

[1]Predictions and modeling of networks are very popular in different fields of studies such as Biology [26] or Civil Engineering [27].

content-boosted collaborative filtering [76], probabilistic models [91], pairwise kernel methods [12], and filterbots-based methods [88] have been developed to blend collaborative filtering with side information, they are specifically designed to address the data sparsity problem and fail to cope with cold-start users or items problem which is the main focus of this thesis. Various methods for blending pure collaborative filtering with content-based filtering have been developed. For example, these include content-boosted collaborative filtering [76], probabilistic models [91], pairwise kernel methods [12], and filterbots-based method [88]. A method based on the predictive discrete latent factor models was proposed [4], which makes use of the additional information in the form of pre-specified covariates.

2.3.2 Network Completion and Cold-start Recommendation

Network completion and inference. There are a few learning algorithms for network completion problem where the common theme is to make structural assumption about the underlying network and devise efficient methods to construct it. In [55] the network completion problem in which an incomplete network including unobserved nodes and edges is given is addressed. It is assumed that the underlying network follows the Kronecker graph model. The Expectation Maximization (EM) framework has been used to infer the unobserved part of the network. A new mathematical and computational method which can identify both missing and spurious interactions in complex networks by using the stochastic block models to capture the structural features in the network is presented in [45]. This method was also applied to a protein interaction network of yeast. In [49] a sampling method to derive confidence intervals from sample networks is proposed. Also, the problem of link prediction [61] relates to our work in which the aim is to predict the future edges of a network. Since we do not observe any links for unsampled nodes in network completion, we cannot utilize the local link structures or simple link prediction algorithms for those nodes. Therefore, the statistical models are not feasible or do not perform well for network completion problem.

Transduction with matrix completion. In matrix completion the objective is to fill out the missing entries of a matrix based on the observed ones. Early work on matrix completion, also referred to as maximum margin matrix factorization [110] was developed for collaborative filtering. Theoretical studies show that, it is sufficient to perfectly recover a matrix [122, 77, 92]. Several recent studies involve matrix recovery with side information are based on graphical models by assuming special distribution of latent factors; these algorithms, as well as [32, 85], consider side information in matrix factorization. Finally, we note that matrix completion with infinite dimensional side information was exploited in [1].

Chapter 3

Decoupled Completion and Transduction Algorithm

3.1 Matrix Completion

Although the matrix completion paradigm provides an appealing solution to the collaborative filtering problem in recommendation systems, some major issues, such as data sparsity and cold-start problems, still remain open. In particular, when the rating data for a subset of users or items is entirely missing, commonly known as the *cold-start* problem, the standard matrix completion methods are inapplicable due the non-uniform sampling of available ratings. In recent years there has been considerable interest in dealing with cold-start users or items that are principally based on the idea of exploiting other sources of information to compensate for this lack of rating data. In this chapter, we propose a novel and general algorithmic framework based on matrix factorization that simultaneously exploits the similarity information among users and items to alleviate the cold-start problem. In contrast to existing methods, our proposed algorithm *decouples* the following two aspects of the cold-start problem to effectively exploit the side information: (i) the completion of a rating sub-matrix, which is generated by excluding cold-start users/items from the original rating matrix; and (ii) the transduction of knowledge from existing ratings to cold-start items/users using side information. This crucial difference significantly boosts the performance when appropriate side information is incorporated. The recovery error of the proposed algorithm is analyzed theoretically and, to the best of our knowledge, this is the first algorithm that addresses the cold-start problem with provable guarantees on performance. We conduct thorough experiments on real datasets that complement our theoretical results. These experiments demonstrate the effectiveness of the proposed algorithm in handling the cold-start users/items problem and mitigating data sparsity issues.

3.1.1 The Setting

In this section we establish the notations which are used throughout the chapter. Our general convention throughout this chapter (and thesis) is to use lower case letters such as u for scalars and bold face lower case letters such as $\mathbf{u}$ for vectors. The set of non-negative real numbers is denoted by $\mathbb{R}_+$. We use $[n]$ to denote a set on integers $\{1, 2, \ldots, n\}$. We use bold face upper case letters such as $\mathbf{M}$ to denote matrices. The transpose of a vector and a matrix denoted by $\mathbf{m}^\top$ and $\mathbf{M}^\top$, respectively. The Frobenius and spectral norms of a matrix $\mathbf{M} \in \mathbb{R}^{n\times m}$ is denoted by $\|\mathbf{M}\|_\mathrm{F}$, i.e, $\|\mathbf{M}\|_\mathrm{F} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}|\mathbf{M}_{ij}|^2}$ and $\|\mathbf{M}\|_2$, respectively. The nuclear norm of a matrix is denoted by $\|\mathbf{M}\|_* = \mathrm{trace}(\sqrt{\mathbf{M}^\top\mathbf{M}})$. We use $(\mathbf{M})^\dagger$ to denote the Moore-Penrose pseudo inverse of matrix $\mathbf{M}$. The dot product between two vectors $\mathbf{m}$ and $\mathbf{n}$ is denoted by $\mathbf{m}^\top\mathbf{n}$.

In collaborative filtering we assume that there is a set of n users $\mathcal{U} = \{u_1, \ldots, u_n\}$ and a set of m items $\mathcal{I} = \{i_1, \ldots, i_m\}$ where each user u_i expresses opinions about a subset of items. In this chapter, we assume opinions are expressed through an explicit numeric rating (e.g., scale from one to five), but other rating methods such as hyperlink clicks are possible as well. We are mainly interested in recommending a set of items for an active user such that the user has not rated these items before. The rating information is summarized in an

$n \times m$ matrix $\mathbf{R} \in \mathbb{R}^{n \times m}, 1 \leq i \leq n, 1 \leq j \leq m$ where the rows correspond to the users and the columns correspond to the items and (p, q)-th entry is the rate given by user u_p to the item i_q. We note that the rating matrix is partially observed and it is sparse in most cases.

An efficient and effective approach for recommender systems is to factorize the user-item rating matrix $\mathbf{R}$ by a multiplicative of k-rank matrices $\mathbf{R} \approx \mathbf{U}\mathbf{V}^\top$, where $\mathbf{U} \in \mathbb{R}^{n \times k}$ and $\mathbf{V} \in \mathbb{R}^{m \times k}$ utilize the factorized user-specific and item-specific matrices, respectively, to make further missing data prediction. There are two basic formulations to solve this problem: these are optimization approaches (see e.g., [97, 65, 71, 58]) and probabilistic approaches [80]. Let $\Omega_{\mathbf{R}}$ be the set of observed ratings in the user-item matrix $\mathbf{R} \in \mathbb{R}^{n \times m}$, i.e., $\Omega_{\mathbf{R}} = \{(i, j) \in [n] \times [m] : \mathbf{R}_{ij} \;\; \text{has been observed}\}$. In optimization based matrix factorization, the goal is to learn the latent matrices $\mathbf{U}$ and $\mathbf{V}$ by solving the following optimization problem:

$$\begin{aligned} \mathcal{L}(\mathbf{U}, \mathbf{V}) = \min_{\mathbf{U},\mathbf{V}} \frac{1}{2} \sum_{(i,j)\in\Omega_{\mathbf{R}}} \left(\mathbf{R}_{i,j} - \mathbf{u}_i^\top \mathbf{v}_j\right)^2 \\ + \lambda_{\mathbf{U}} \|\mathbf{U}\|_{\mathrm{F}}^2 + \lambda_{\mathbf{V}} \|\mathbf{V}\|_{\mathrm{F}}^2 \end{aligned} \tag{3.1}$$

The optimization problem in Eq. (3.1) has three main terms. The first term aims to minimize the inconsistency between the observed entries and their corresponding values obtained by the factorized matrices. The last two terms regularize the latent matrices for users and items, respectively. The parameters $\lambda_{\mathbf{U}}$ and $\lambda_{\mathbf{V}}$ are regularization parameters that are introduced to control the regularization of latent matrices $\mathbf{U}$ and $\mathbf{V}$, respectively. After learning the latent features for users and items, the prediction of each missing entry can be computed by the inner product of latent vectors of the corresponding row and the corresponding column.

For new users or items in the system, since there is no observed rating data in $\mathbf{R}$, the corresponding rows and columns in the rating matrix are fully unobserved which results in an inability to draw inferences to either recommend existing items to new users or new items to existing users. In particular, it is not feasible for standard matrix factorization methods to fill in a row or a column that is entirely missing in the original rating matrix. In this chapter, we assume that there is a sub-matrix $\mathbf{M} \in \mathbb{R}^{p\times q}, 1 \leq p \leq n, 1 \leq q \leq m$ which includes enough rating data to be fully recovered via standard methods such as matrix factorization or matrix completion. We call the rest of items and users for which the rating data is entirely missing and are not present in $\mathbf{M}$ as cold-start. To make recommendations to cold-start users/items we assume that besides the observed entries in the matrix $\mathbf{M}$, there exist two auxiliary similarity matrices $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{B} \in \mathbb{R}^{m\times m}$ that capture the pairwise similarity between users and items, respectively. The similarity matrices can be computed from the available information such as users' profile or social context or items' features. The main focus of this chapter is on exploiting available side information to improve the accuracy of recommendations and resolve cold-start item and user problems.

Although the main focus of this chapter is dealing with cold-start problems, our general framework, will also apply to other problems involving matrix completion with side information such as network completion [73], where a small sample of a network is observed and side information about the nodes are available (i.e., $\mathbf{R} \in \{0, 1, ?\}^{n\times n}$ is the partially observed adjacency matrix where '?' stands for a unobserved link and $\mathbf{A} = \mathbf{B} \in \mathbb{R}_{+}^{n\times n}$ is the pairwise similarity matrix between the nodes) and the goal is to infer the unobserved part of the network.

3.1.2 Subspace sharing and error propagation

Before delving into the algorithm, we discuss the matrix co-factorization method used to exploit the similarity information. This has been proven to be very effective for handling cold-start problems [102] and motivates our own algorithm.

A straightforward approach to exploit and transfer knowledge from similarity matrices to the rating data is to cast the problem as a shared subspace learning framework (a.k.a matrix co-factorization) based on a joint matrix factorization to jointly learn a common subset of basis vectors for the rating matrix **R** and the similarity matrices **A** and **B** for users and items as formulated in the following optimization problem:

$$\begin{aligned}\min_{\substack{\mathbf{U}\in\mathbb{R}^{n\times r},\mathbf{V}\in\mathbb{R}^{m\times r},\\ \mathbf{W}\in\mathbb{R}^{n\times r},\mathbf{Z}\in\mathbb{R}^{m\times r}}} \; & \frac{1}{2}\|\mathbf{R}-\mathbf{U}\mathbf{V}^\top\|_{\mathrm{F}}^2 + \lambda\left(\|\mathbf{U}\|_{\mathrm{F}}^2 + \|\mathbf{V}\|_{\mathrm{F}}^2\right) \\ & + \frac{1}{2}\|\mathbf{A}-\mathbf{U}\mathbf{W}^\top\|_{\mathrm{F}}^2 + \frac{1}{2}\|\mathbf{B}-\mathbf{Z}\mathbf{V}^\top\|_{\mathrm{F}}^2 \\ & + \lambda\left(\|\mathbf{W}\|_{\mathrm{F}}^2 + \|\mathbf{Z}\|_{\mathrm{F}}^2\right)\end{aligned} \tag{3.2}$$

with λ as the regularization parameter for the norms of the solution matrices and common latent space representation is achieved by using the same matrices **W** and **Z**.

The main issue with this approach is that the *completion* of the unobserved entries in rating matrix **R** and *transduction* of knowledge from these entries to cold-start users/items via similarity matrices is carried out simultaneously. Therefore, the completion and transduction errors are propagated repetitively and uncontrollably. The issue with error propagation becomes even worse due to the non-convexity of optimization problems in Eq. (3.8)– jointly in parameters **U** and **V**.

In an effort to alleviate this difficulty, we propose an alternative approach that diverges from these algorithms and transfers information from similarity matrices to a rating matrix via the fully recoverable sub-matrix $\mathbf{M}$. In particular, the proposed algorithm **decouples** the *completion* from *transduction* and constitutes of two stages: (i) completion of the sub-matrix $\mathbf{M}$ which can be done perfectly with zero completion error with a very high probability, and (ii) transduction of rating data from the recovered sub-matrix to cold-start items and users using similarity matrices. This crucial difference greatly boosts the performance of the proposed algorithm when appropriate side information is incorporated.

3.1.3 A decoupled solution

With our assumption on the correlation of rating and similarity matrices in place, we can now describe our algorithm. To this end, we construct an orthogonal matrix $\mathbf{U}_{\mathbf{A}} = [\mathbf{u}_1^{\mathbf{A}}, \cdots, \mathbf{u}_s^{\mathbf{A}}] \in \mathbb{R}^{n \times s}$ whose column space subsumes the row space of the rating matrix. We also construct another orthogonal matrix $\mathbf{U}_{\mathbf{B}} = [\mathbf{u}_1^{\mathbf{B}}, \cdots, \mathbf{u}_s^{\mathbf{B}}] \in \mathbb{R}^{m \times s}$ whose column space subsumes the column space of the rating matrix. To construct subspaces $\mathbf{U}_{\mathbf{A}}$ and $\mathbf{U}_{\mathbf{B}}$ we use the first s eigen-vectors corresponding to the s largest eigen-values of the provided similarity matrices $\mathbf{A}$ and $\mathbf{B}$, respectively.

We note that the extent to which the extracted subspaces $\mathbf{U}_{\mathbf{A}}$ and $\mathbf{U}_{\mathbf{B}}$ from similarity matrices subsume the corresponding row and column spaces of the rating matrix, depends on the richness of the similarity information. To formalize this, first, from the low-rank assumption of the rating matrix $\mathbf{R}$, it follows that it can be decomposed as $\mathbf{R} = \sum_{i=1}^{r} \mathbf{u}_i \mathbf{v}_i^\top$ where r is the rank of the matrix. Now we note that the i-th latent features vector $\mathbf{u}_i$ can be decomposed in a unique way into two parts, parallel and orthogonal: $\mathbf{u}_i = \mathbf{u}_i^{\|} + \mathbf{u}_i^{\perp}$, where $\mathbf{u}_i^{\|}$ is the part that is spanned by the subspace $\mathbf{U}_{\mathbf{A}}$ extracted from the similarity information and $\mathbf{u}_i^{\perp}$ is the part orthogonal to $\mathbf{U}_{\mathbf{A}}$.

In a similar way, for the latent features vector of j-th item, i.e., $\mathbf{v}_j$, we have its decomposition as $\mathbf{v}_i = \mathbf{v}_i^{\|} + \mathbf{v}_i^{\perp}$, where $\mathbf{v}_i^{\|}$ is the part that is spanned by the subspace $\mathbf{U}_{\mathbf{B}}$ extracted from the similarity information about users and $\mathbf{v}_i^{\perp}$ is the part orthogonal to $\mathbf{U}_{\mathbf{B}}$. We note that the orthogonal components of left and singular vectors $\mathbf{u}_i^{\perp}$ and $\mathbf{v}_i^{\perp}$ capture the extent to which the similarity matrices do not provide information about rating data and can not be recovered using these auxiliary information.

To build intuition for the algorithm that we propose, we first relate the rating matrix to the similarity matrices. Having decomposed the latent features as above into two parallel and orthogonal components, we can rewrite the rating matrix $\mathbf{R}$ as:

$$\begin{aligned}
\mathbf{R} &= \sum_{i=1}^{r} \mathbf{u}_i \mathbf{v}_i^{\top} = \sum_{i=1}^{r} (\mathbf{u}_i^{\|} + \mathbf{u}_i^{\perp})(\mathbf{v}_i^{\|} + \mathbf{v}_i^{\perp})^{\top} \\
&= \sum_{i=1}^{r} \mathbf{u}_i^{\|} \mathbf{v}_i^{\|\top} + \sum_{i=1}^{r} \mathbf{u}_i^{\|} \mathbf{v}_i^{\perp\top} + \sum_{i=1}^{r} \mathbf{u}_i^{\perp} \mathbf{v}_i^{\|\top} + \sum_{i=1}^{r} \mathbf{u}_i^{\perp} \mathbf{v}_i^{\perp\top} \qquad (3.3) \\
&= \mathbf{R}_* + \mathbf{R}_{\mathrm{L}} + \mathbf{R}_{\mathrm{R}} + \mathbf{R}_{\mathrm{E}},
\end{aligned}$$

where $\mathbf{R}_*$ is the part of the rating matrix that is fully spanned by the subspaces $\mathbf{U}_{\mathbf{A}}$ and $\mathbf{U}_{\mathbf{B}}$, the matrix $\mathbf{R}_{\mathrm{L}}$ is the part where only the left singular vectors are spanned by $\mathbf{U}_{\mathbf{A}}$ and the right singular vectors are orthogonal to the subspace spanned by $\mathbf{U}_{\mathbf{B}}$, and the matrix $\mathbf{R}_{\mathrm{R}}$ is the part that the left singular vectors are orthogonal to the subspace spanned by $\mathbf{U}_{\mathbf{A}}$ and the right singular vectors are spanned by $\mathbf{U}_{\mathbf{B}}$. Finally, the matrix $\mathbf{R}_{\mathrm{E}}$ is the error matrix where both left and singular vectors are orthogonal to the subspaces spanned by $\mathbf{U}_{\mathbf{A}}$ and $\mathbf{U}_{\mathbf{B}}$, respectively, which does not benefit form the side information at all. In particular, the error matrix $\mathbf{R}_{\mathrm{E}}$ can not be recovered from the side information as the extracted subspaces provide no information about the orthogonal parts $\mathbf{u}_i^{\perp}$ and $\mathbf{v}_i^{\perp}$ of the singular vectors. Therefore, the error contributed by this matrix into the estimation error of final recovered rating matrix

Algorithm 1 `RECT`

```
1: Input:
   ❶ $\mathbf{R} \in \mathbb{R}^{n\times m}, r$: the partially observed rating matrix and its rank
   ❷ $\mathbf{A} \in \mathbb{R}^{n\times n}$: the users' similarity matrix
   ❸ $\mathbf{B} \in \mathbb{R}^{m\times m}$: the items' similarity matrix
2: Extract the maximal recoverable rating sub-matrix $\mathbf{M} \in \mathbb{R}^{p\times q}$ (according to Theorem ??)
3: Complete the sub-matrix $\mathbf{M}$ to get $\widehat{\mathbf{M}}$        (according to equation 3.9)
4: Decompose $\widehat{\mathbf{M}}$ as $\widehat{\mathbf{M}} = \sum_{i=1}^{r} \hat{\mathbf{u}}_i \hat{\mathbf{v}}_i^\top$
5: Extract subspaces $\mathbf{U}_\mathbf{A}$ and $\mathbf{U}_\mathbf{B}$ by spectral clustering from similarity matrices $\mathbf{A}$ and $\mathbf{B}$, respectively
6: Compute $\widehat{\mathbf{a}}_i = \left(\widehat{\mathbf{U}}_\mathbf{A}^\top \widehat{\mathbf{U}}_\mathbf{A}\right)^\dagger \widehat{\mathbf{U}}_\mathbf{A}^\top \hat{\mathbf{u}}_i, i = 1, 2, \cdots, r$
7: Compute $\widehat{\mathbf{b}}_i = \left(\widehat{\mathbf{U}}_\mathbf{B}^\top \widehat{\mathbf{U}}_\mathbf{B}\right)^\dagger \widehat{\mathbf{U}}_\mathbf{B}^\top \hat{\mathbf{v}}_i, i = 1, 2, \cdots, r$
8: Compute $\widehat{\mathbf{R}} = \mathbf{U}_\mathbf{A} \left(\sum_{i=1}^{r} \widehat{\mathbf{a}}_i \widehat{\mathbf{b}}_i^\top\right) \mathbf{U}_\mathbf{B}^\top$
9: Output: $\widehat{\mathbf{R}}$
```

is unavoidable.

In the following subsections, we first devise an effective method to recover the rating matrix $\mathbf{R}$ from the sub-matrix $\mathbf{M}$ and subspaces $\mathbf{U}_\mathbf{A}$ and $\mathbf{U}_\mathbf{B}$, and then provide theoretical guarantees on the estimation error in terms of the magnitude of the error matrix $\|\mathbf{R}_\mathrm{E}\|_\mathrm{F}$.

The completion stage. The first step in Algorithm 2, after extracting the sub-matrix $\mathbf{M}$, is to complete $\mathbf{M}$ to get the fully recovered matrix $\widehat{\mathbf{M}}$. To do so, we use the matrix factorization formulation in Eq. (3.1) which has achieved great success and popularity among the existing matrix completion techniques [110, 80, 104]. In particular, we solve the following convex optimization algorithm [18] to fully recover the submatrix:

$$\widehat{\mathbf{M}} = \arg \min_{\mathbf{X}\in\mathbb{R}^{p\times q}} \quad \|\mathbf{X}\|_* \qquad \text{s.t.} \quad \mathbf{X}_{ij} = \mathbf{M}_{ij},\ \forall\ (i,j) \in \Omega_\mathbf{M} \tag{3.4}$$

where $\Omega_{\mathbf{M}} \subseteq \Omega$ is the set of observed ratings in $\mathbf{M}$.

We note that based on matrix completion theory, it is guaranteed that the low rank matrix $\mathbf{M}$ can be perfectly recovered provided that the number of observed entries is sufficient.

The transduction stage. We now turn to recovering the matrix $\mathbf{R} = \sum_{i=1}^{r} \mathbf{u}_i \mathbf{v}_i^\top$ from the submatrix $\widehat{\mathbf{M}}$ and the subspaces $\mathbf{U}_{\mathbf{A}}$ and $\mathbf{U}_{\mathbf{B}}$ extracted from the similarity matrices $\mathbf{A}$ and $\mathbf{B}$ about users and items, respectively. The detailed steps of the proposed completion algorithm are shown in Algorithm 2.

In the next step, the rating information in the recovered matrix $\widehat{\mathbf{M}}$ is transduced to the cold-start users and items. To motivate the transduction step, let us focus on the $\mathbf{R}_*$ matrix as defined in Eq. (3.3). Since $\mathbf{u}_i^{\|}$ and $\mathbf{v}_i^{\|}$ are fully spanned by the subspaces $\mathbf{U}_{\mathbf{A}}$ and $\mathbf{U}_{\mathbf{B}}$ following our construction above, we can write them as:

$$\begin{aligned} \mathbf{u}_i^{\|} &= \mathbf{U}_{\mathbf{A}} \mathbf{a}_i, i = 1, 2, \cdots, r \\ \mathbf{v}_i^{\|} &= \mathbf{U}_{\mathbf{B}} \mathbf{b}_i, i = 1, 2, \cdots, r, \end{aligned} \tag{3.5}$$

where $\mathbf{a}_i \in \mathbb{R}^s$ and $\mathbf{b}_i \in \mathbb{R}^s, i = 1, 2, \cdots, r$ are the orthogonal projection of the singular vectors onto the corresponding subspaces. By substituting the equations in Eq. (3.5) into the decomposition of $\mathbf{R}_*$ we get:

$$\begin{aligned} \mathbf{R}_* = \sum_{i=1}^{r} \mathbf{u}_i^{\|} \mathbf{v}_i^{\|\top} &= \sum_{i=1}^{r} \mathbf{U}_{\mathbf{A}} \mathbf{a}_i \mathbf{b}_i^\top \mathbf{U}_{\mathbf{B}}^\top \\ &= \mathbf{U}_{\mathbf{A}} \left(\sum_{i=1}^{r} \mathbf{a}_i \mathbf{b}_i^\top \right) \mathbf{U}_{\mathbf{B}}^\top \end{aligned} \tag{3.6}$$

From the above derivation, we observe that the key for the recovery of the matrix $\mathbf{R}_*$ is to estimate the vectors $\mathbf{a}_i, \mathbf{b}_i, i = 1, 2, \cdots, r$. Next we show how the recovered rating submatrix $\widehat{\mathbf{M}}$, along with the subspaces extracted from the similarity matrices, can be utilized to estimate these vectors under some mild conditions on the number of cold-start users and items. To this end, first consider the decomposition of the recovered matrix as $\widehat{\mathbf{M}} = \sum_{i=1}^{r} \hat{\mathbf{u}}_i \hat{\mathbf{v}}_i^\top$. The estimation of vectors $\mathbf{a}_i, \mathbf{b}_i, i = 1, 2, \cdots, r$ in Eq. (3.6) and equivalently the matrix $\mathbf{R}_*$ is as follows. First, let $\widehat{\mathbf{U}}_{\mathbf{A}} \in \mathbf{R}^{p \times s}$ be a random submatrix of $\mathbf{U}_{\mathbf{A}}$ where the sampled rows correspond to the subset of rows in the matrix $\widehat{\mathbf{M}}$. Similarly we construct a submatrix of $\mathbf{U}_{\mathbf{B}}$ denoted by $\widehat{\mathbf{U}}_{\mathbf{B}} \in \mathbf{R}^{q \times s}$ by sampling the rows of $\mathbf{U}_{\mathbf{B}}$ corresponding to the columns in $\widehat{\mathbf{M}}$. An estimation of $\mathbf{a}_i, \mathbf{b}_i, i \in [r]$ vectors is obtained by orthogonal projection of left and right singular vectors of $\widehat{\mathbf{M}}$ onto the sampled subspaces $\widehat{\mathbf{U}}_{\mathbf{A}}$ and $\widehat{\mathbf{U}}_{\mathbf{B}}$ by solving following optimization problems:

$$\begin{aligned} \hat{\mathbf{a}}_i &= \arg\min_{\mathbf{a} \in \mathbb{R}^s} \left\| \hat{\mathbf{u}}_i - \widehat{\mathbf{U}}_{\mathbf{A}} \mathbf{a} \right\|_2^2, \\ \widehat{\mathbf{b}}_i &= \arg\min_{\mathbf{b} \in \mathbb{R}^s} \left\| \hat{\mathbf{v}}_i - \widehat{\mathbf{U}}_{\mathbf{B}} \mathbf{a} \right\|_2^2. \end{aligned} \tag{3.7}$$

Then, we estimate $\mathbf{R}_*$ by:

$$\begin{aligned} \widehat{\mathbf{R}}_* &= \mathbf{U}_{\mathbf{A}} \left(\sum_{i=1}^{r} \hat{\mathbf{a}}_i \widehat{\mathbf{b}}_i^\top \right) \mathbf{U}_{\mathbf{B}}^\top \\ &= \mathbf{U}_{\mathbf{A}} \left(\widehat{\mathbf{U}}_{\mathbf{A}}^\top \widehat{\mathbf{U}}_{\mathbf{A}} \right)^\dagger \widehat{\mathbf{U}}_{\mathbf{A}}^\top \left(\sum_{i=1}^{r} \hat{\mathbf{u}}_i \hat{\mathbf{v}}_i^\top \right) \widehat{\mathbf{U}}_{\mathbf{B}} \left(\widehat{\mathbf{U}}_{\mathbf{B}}^\top \widehat{\mathbf{U}}_{\mathbf{B}} \right)^\dagger \mathbf{U}_{\mathbf{B}}, \end{aligned}$$

where in the last equality we used the fact that $\left(\widehat{\mathbf{U}}_{\mathbf{A}}^\top \widehat{\mathbf{U}}_{\mathbf{A}} \right)^\dagger \widehat{\mathbf{U}}_{\mathbf{A}}^\top \hat{\mathbf{u}}_i$ and $\left(\widehat{\mathbf{U}}_{\mathbf{B}}^\top \widehat{\mathbf{U}}_{\mathbf{B}} \right)^\dagger \widehat{\mathbf{U}}_{\mathbf{B}}^\top \hat{\mathbf{v}}_i$ are the optimal solutions to the ordinary least squares regression problems in Eq. (3.7). Here $(\cdot)^\dagger$ denotes the Moore-Penrose pseudo inverse of a matrix. The final estimated rating matrix $\widehat{\mathbf{R}}$ is simply set to be $\widehat{\mathbf{R}} = \widehat{\mathbf{R}}_*$.

Remark 3.1.1. *The main computational cost in implementing the Algorithm 2 is the spec-*

tral clustering of similarity matrices, orthogonal projections onto subspaces $\widehat{\mathbf{U}}_{\mathbf{A}}$ *and* $\widehat{\mathbf{U}}_{\mathbf{B}}$*, and computing the singular value decomposition of* $\widehat{\mathbf{M}}$*. We note that although the original rating matrix* $\mathbf{R}$ *might be large, the recoverable sub-matrix is significantly small and can be decomposed efficiently.*

3.2 Network Completion

This section investigates the network completion problem, where it is assumed that only a small sample of a network (e.g., a complete or partially observed subgraph of a social graph) is observed and we would like to infer the unobserved part of the network. In this chapter, we assume that besides the observed subgraph, side information about the nodes such as the pairwise similarity between them is also provided. In contrast to the original network completion problem where the standard methods such as matrix completion is inapplicable due the non-uniform sampling of observed links, we show that by effectively exploiting the side information, it is possible to accurately predict the unobserved links. In contrast to existing matrix completion methods with side information such as shared subspace learning and matrix completion with transduction, the proposed algorithm decouples the completion from transduction to effectively exploit the similarity information. This crucial difference greatly boosts the performance when appropriate similarity information is used. The recovery error of the proposed algorithm is theoretically analyzed based on the richness of the similarity information and the size of the observed submatrix. To the best of our knowledge, this is the first algorithm that addresses the network completion with similarity of nodes with provable guarantees. Experiments on synthetic and real networks from Facebook and Google+ show that the proposed two-stage method is able to accurately reconstruct the network and outperforms other methods.

3.2.1 The Setting

Before proceeding to the proposed algorithm, in this section we establish the notation used throughout the section and formally describe our problem setting.

Notation. Scalars are denoted by lower case letters, vectors by bold face lower case letter such as $\mathbf{a}$, and matrices by bold face upper letter such as $\mathbf{M}$. We use $[n]$ to denote the set of integers $\{1, 2, \cdots, n\}$. The dot product between two vectors $\mathbf{a}$ and $\mathbf{b}$ is denoted by $\mathbf{a}^\top \mathbf{b}$. The parallel and perpendicular projections of a vector $\mathbf{a}$ to a subspace are denoted by $\mathbf{a}^{\|}$ and $\mathbf{a}^{\perp}$, respectively. The Frobenius, spectral, and trace norms of a matrix $\mathbf{M}$ are denoted by $\|\mathbf{M}\|_{\mathrm{F}}$, $\|\mathbf{M}\|_2$, and $\|\mathbf{M}\|_*$, respectively. The transpose of a vector and a matrix are denoted by $\mathbf{a}^\top$ and $\mathbf{M}^\top$, respectively. We use $(\mathbf{A})^\dagger$ to denote the Moore-Penrose pseudo inverse of the matrix $\mathbf{A}$.

To formalize the setting, we assume there is a true undirected unweighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ on $n = |\mathcal{V}|$ distinguishable vertices with adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$. In this section we assume that only a partially observed submatrix $\mathbf{O} \in \{0, 1, ?\}^{m \times m}, 1 \leq m \leq n$ induced by a sample sub-graph $\mathcal{G}' = (\mathcal{V}, \mathcal{E}'), \mathcal{E}' \subset \mathcal{E}$ of original graph is given. Here "0" represents a known absent edge, "1" denotes a known present edge, and "?" indicates an unobserved edge. In contrast to the classical link prediction problem where the assumption is that random entries of $\mathbf{A}$ are missing, in network completion problem [55], the information about a subset of nodes is entirely missing, i.e., the corresponding row of these nodes is completely missing. However, we assume that missing edges in $\mathbf{O}$ are sampled uniformly at random and matrix completion methods are capable of completing this matrix.

We assume that besides the observed entries, side information about nodes in the social graph is also available. In particular, it is assumed that the pairwise similarity between nodes is obtained from features of nodes and it is captured in a similarity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$. The overarching goal is to show that side information can potentially benefit the process of network completion. In particular, we aim at predicting the unobserved edges to complete the adjacency matrix $\mathbf{A}$ based on the partially observed submatirx $\mathbf{O}$ and the similarity matrix $\mathbf{S}$.

3.2.2 The Proposed Algorithm

We now turn to describing our algorithm and the assumptions underlying it. We assume that the structure of the network and the similarity information are correlated and to some extent, which will be formalized later, share the same latent information; that is, the row vectors in $\mathbf{A}$ share an underlying subspace spanned by the leading eigen-vectors in the similarity matrices $\mathbf{S}$. This assumption follows the fact that the similarity of nodes provides auxiliary information about the links that exist between them; otherwise there would not be any hope to benefit from these information in completing the matrix.

Before delving into the algorithm, we discuss two alternative methods and note a few of their deficiencies for our setting. A naive approach to solve the network completion problem is to apply the standard tools from matrix completion [19, 18, 93] and in particular transductive matrix completion [42] by incorporating $\mathbf{S}$ as side information [1]. However, as mentioned earlier, in contrast to the classical matrix completion or missing-link inference problem where the assumption is that random entries of $\mathbf{A}$ are missing, in our case complete rows of $\mathbf{A}$ are missing, which makes the standard matrix completion algorithms inapplicable.

[1]We note that for many real world social networks the underlying adjacency matrix is low-rank (e.g., see [21]).

Another straightforward approach to exploit and transfer knowledge from similarity matrix to predict the structure of network is to cast the problem as a shared subspace learning framework based on a joint matrix factorization to jointly learn a common subset of basis vectors for the adjacency matrix $\mathbf{A}$ and the corresponding similarity matrix $\mathbf{S}$. In particular, the goal would be to factorize the matrices $\mathbf{A}$ and $\mathbf{S}$ into three subspaces: one is shared between the adjacency and the similarity matrices, and two are specific to the matrices as formulated in the following optimization problem:

$$\begin{aligned}\min_{\mathbf{U},\mathbf{V},\mathbf{W}} \quad & \frac{1}{2}\|\mathbf{A}-\mathbf{U}\mathbf{V}^\top\|_{\mathrm{F}}^2+\frac{1}{2}\|\mathbf{S}-\mathbf{U}\mathbf{W}^\top\|_{\mathrm{F}}^2 \\ & +\lambda\left(\|\mathbf{U}\|_{\mathrm{F}}^2+\|\mathbf{V}\|_{\mathrm{F}}^2+\|\mathbf{W}\|_{\mathrm{F}}^2\right)\end{aligned} \tag{3.8}$$

with λ as the regularization parameter for the norms of the solution matrices.

The main issue with subspace sharing is that the *completion* of the unobserved entries of adjacency matrix $\mathbf{A}$ from sampled observed in $\mathbf{O}$ and *transduction* of knowledge from these entries to fully unobserved nodes via similarity matrix $\mathbf{S}$, is carried out simultaneously. Therefore, the completion and transduction errors are propagated repetitively and in an uncontrolled way that hinders the effectiveness of similarity information.

In effort to alleviate these difficulties, we propose an alternative approach that diverges from theres algorithms and transfers information from similarity matrices to adjacency matrix via the fully recoverable submatrix $\mathbf{O}$. In particular, the proposed algorithm *decouples* the completion from transduction and constitutes of two stages: (i) completion of the partially observed submatrix $\mathbf{O}$ which can be done perfectly with zero completion error, and (ii) transduction of links from the recovered submatrix to unobserved nodes using similarity matrix. This crucial difference provides an effective way to exploit the similarity informa-

Algorithm 2 Network Completion with Side Information [34]

```
1: Input:
     • $n$: the number of nodes in netwrok $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
     • $\mathbf{O}$: the adjacency matrix of subgraph with $m$ nodes
     • $\mathbf{S}$: the partially observed pairwise similarity matrix
     • $s \geq \text{rank}(\mathbf{A})$: number of eigenvectors in subspace
                                                                  [Extraction]
2: Extract $\mathbf{U}_s$ from $\mathbf{S}$ by spectral clustering
3: Complete the submatrix $\mathbf{O}$ by solving the convex optimization problem in (3.9) to get
   $\widehat{\mathbf{O}}$                                          [Completion]
4: Sample $m$ rows of $\mathbf{U}_s \in \mathbb{R}^{n\times s}$ uniformly at random to create matrix $\widehat{\mathbf{U}}_s \in \mathbb{R}^{m\times s}$
5: Set $\widehat{\Lambda} = \left(\widehat{\mathbf{U}}_s^\top \widehat{\mathbf{U}}_s\right)^\dagger \widehat{\mathbf{U}}_s^\top \widehat{\mathbf{O}} \widehat{\mathbf{U}}_s \left(\widehat{\mathbf{U}}_s^\top \widehat{\mathbf{U}}_s\right)^\dagger$
6: Output: $\widehat{\mathbf{A}} = \mathbf{U}_s \widehat{\Lambda} \mathbf{U}_s^\top$             [Transduction]
```

tion and greatly boosts the performance of the proposed algorithm when appropriate side information is incorporated.

With our assumption on the correlation of adjacency matrix and node similarities in place, we can now describe our algorithm. The detailed steps of the proposed algorithms are shown in Algorithm 2. In what follows, we will discuss each stage in greater detail.

Subspace extraction. The first step in Algorithm 2 is to extract a representative subsapce from the similarity matrix $\mathbf{S}$. To this end, we extract an orthogonal matrix $\mathbf{U}_s \in \mathbf{R}^{n\times s}$ from similarity matrix whose column space subsumes the column space of adjacency matrix, where s is chosen to be larger the rank of adjacency matrix, i.e., $s \geq \text{rank}(\mathbf{A})$. To construct subspaces $\mathbf{U}_s$ we use the first s eigen-vectors corresponding to the s largest eigen-values of the provided similarity matrices $\mathbf{S}$ which can be done by singular value decomposition or spectral clustering [67].

Completion. In the second step, we recover the partially observed submatrix $\mathbf{O}$. Although tools from matrix completion are not applicable to the full adjacency matrix $\mathbf{A}$, but since based on our assumption the observed links in the sampled subgraph are uniformly sampled, we can recover the submatrix $\mathbf{O}$ using tools fom matrix completion. To do so, let Ω be the set of observed links in the induced submatirx $\mathbf{O}$, i.e., $\Omega = \{(i,j) \in [n] \times [n] : \mathbf{O}_{ij}$ has been observed$\}$. Then we solve the following convex optimization algorithm [18] to fully recover the submatrix:

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{X}\|_* \\ \text{s.t.} \quad & \mathbf{X}_{ij} = \mathbf{O}_{ij}, \ \forall \ (i,j) \in \Omega. \end{aligned} \tag{3.9}$$

We use $\widehat{\mathbf{O}}$ to denote the optimal solution of the optimization problem in Eq. (3.9).

Transduction. Having recovered the submatrix and extracted the subspace $\mathbf{U}_s$ from the similarity matrix, we now turn to the transduction step. To do so, we first note that since both the adjacency matrix of the social network $\mathbf{A}$ and the recovered submatrix $\widehat{\mathbf{O}}$ are low rank, we can decompose these matrices as:

$$\mathbf{A} = \sum_{i=1}^{r} \mathbf{a}_i \mathbf{a}_i^\top \quad \text{and} \quad \widehat{\mathbf{O}} = \sum_{i=1}^{r} \widehat{\mathbf{a}}_i \widehat{\mathbf{a}}_i^\top \tag{3.10}$$

where r is the rank of adjacency matrix. To see this, we can consider $\mathbf{a}_i \in \{1,0\}^n$ and $\widehat{\mathbf{a}}_i \in \{1,0\}^m$ as the corresponding memberships assignment to the ith hidden components of the graph. We note that if the similarity matrix is set to be equivalent to the adjacency matrix, then the indicator vectors of connected components are exactly $\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_r$.

To formalize the correlation of similarity information and the adjacency matrix, we assume that both matrices share a common subspace. Therefore each $\mathbf{a}_i, i \in [r]$ can be decomposed in a unique way into two parallel and orthogonal parts as $\mathbf{a}_i = \mathbf{a}_i^{\|} + \mathbf{a}_i^{\perp}$, where $\mathbf{a}_i^{\|}$

belongs to column span of $\mathbf{U}_s$ and $\mathbf{a}_i^{\perp}$ is orthogonal to column span of $\mathbf{U}_s$. Moreover, since $\mathbf{a}_i^{\|}$ belongs to column spam of $\mathbf{U}_s$ we can write

$$\mathbf{a}_i^{\|} = \mathbf{U}_s \mathbf{b}_i, i = 1, 2, \cdots, r, \tag{3.11}$$

for some $\mathbf{b}_i \in \mathbb{R}^s$.

To build intuition for the transduction step, we first relate the adjacency matrix to the similarity matrix. Having decomposed the latent features as above into two parallel and orthogonal components, we can rewrite the adjacency matrix as:

$$\begin{aligned} \mathbf{A} = \sum_{i=1}^{r} \mathbf{a}_i \mathbf{a}_i^{\top} &= \sum_{i=1}^{r} (\mathbf{a}_i^{\|} + \mathbf{a}_i^{\perp})(\mathbf{a}_i^{\|} + \mathbf{a}_i^{\perp})^{\top} \\ &= \sum_{i=1}^{r} \mathbf{a}_i^{\|} \mathbf{a}_i^{\|\top} + \sum_{i=1}^{r} \mathbf{a}_i^{\perp} \mathbf{a}_i^{\perp\top} \\ &= \mathbf{A}_* + \mathbf{A}_{\mathrm{E}}, \end{aligned} \tag{3.12}$$

where $\mathbf{A}_*$ is the part fully captured by the similarity information, and the matrix $\mathbf{A}_{\mathrm{E}}$ is the error matrix whose singular vectors are orthogonal to the subspaces spanned by the similarity subspace and do not benefit form the side information at all. In particular, the error matrix $\mathbf{A}_{\mathrm{E}}$ can not be recovered from the side information as the extracted subspaces provide no information about the orthogonal part $\mathbf{a}_i^{\perp}$ of the singular vectors. Therefore, the error contributed by this matrix into the recovery error of final inferred adjacency matrix is unavoidable.

By combining Eq. (3.11) with Eq. (3.12), the adjacency matrix can be written as:

$$\mathbf{A} = \sum_{i=1}^{r} \mathbf{a}_i^{\|} \mathbf{a}_i^{\|\top} + \mathbf{A}_{\mathrm{E}} = \mathbf{U}_s \left(\sum_{i=1}^{r} \mathbf{b}_i \mathbf{b}_i^{\top} \right) \mathbf{U}_s^{\top} + \mathbf{A}_{\mathrm{E}}. \tag{3.13}$$

From above derivation, we observe that the key to recover the matrix $\mathbf{A}$ is to estimate the vectors $\mathbf{b}_i, i = 1, 2, \cdots, r$. In the following we show how the recovered submatrix $\widehat{\mathbf{O}}$ along with the subspace $\mathbf{U}_s$ extracted from the similarity matrix can be utilized to estimate these vectors perfectly under some mild condition on the number of sampled nodes in the subgraph $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$.

The key idea is that although we can not solve the linear system $\mathbf{a}_i^{\parallel} = \mathbf{U}_s \mathbf{b}_i$ for $\mathbf{b}_i$ to estimate $\mathbf{a}_i$ as $\mathbf{a}_i^{\parallel}$ is not accessible, but we can replace $\mathbf{a}_i$ and $\mathbf{U}_s$ with other accessible vector $\widehat{\mathbf{a}}_i$ and subspace $\widehat{\mathbf{U}}_s$, respectively, and estimate $\mathbf{a}_i^{\parallel}$ perfectly in high probability. In particular let $\widehat{\mathbf{U}}_s$ be a random subspace obtained by $\widehat{\mathbf{U}}_s = \mathbf{\Pi}\mathbf{U}_s$ where $\mathbf{\Pi}$ is a $m \times n$ random matrix distributed as the first m rows of uniformly random permutation matrix of size n where selected rows correspond to the rows in $\mathbf{O}$. To obtain an estimate of $\mathbf{b}_i$ we solve the following optimization problem:

$$\widehat{\mathbf{b}}_i = \min_{\mathbf{b} \in \mathbb{R}^s} \left\| \widehat{\mathbf{a}}_i - \widehat{\mathbf{U}}_s \mathbf{b} \right\|_2^2 = \left(\widehat{\mathbf{U}}_s^\top \widehat{\mathbf{U}}_s \right)^\dagger \widehat{\mathbf{U}}_s^\top \widehat{\mathbf{a}}_i,$$

where $(\cdot)^\dagger$ denotes the Moore-Penrose pseudo inverse of a matrix and in the last equality we used the fact that $\left(\widehat{\mathbf{U}}_s^\top \widehat{\mathbf{U}}_s \right)^\dagger \widehat{\mathbf{U}}_s^\top$ is the optimal solution to the ordinary least squares regression problem above. Having computed an estimate for each $\mathbf{b}_i$, we recover the full matrix by:

$$\widehat{\mathbf{A}} = \mathbf{U}_s \left(\sum_{i=1}^{r} \widehat{\mathbf{b}}_i \widehat{\mathbf{b}}_i^\top \right) \mathbf{U}_s^\top.$$

3.3 Analysis of Recovery Error

In order to see the impact of similarity information on predicting the missing structure of the network, we theoretically analyze the recovery error of the proposed algorithm. In particular, the performance of Algorithm 2 on inferring the full structure of the input graph with adjacency matrix $\mathbf{A}$ is stated in the following theorem [10].

Theorem 3.3.1. *Let $\mathbf{A} \in \{0,1\}^{n\times n}$ be the adjacency matrix of the social graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with coherence parameter μ. Let $\mathbf{O} \in \{0,1,?\}^{m\times m}$ be a partially observed submatrix of $\mathbf{A}$ where the links are uniformly sampled with*

$$m \geq 8\mu r \log\left(\frac{r}{\delta}\right)$$

and number of observed links

$$|\Omega| \geq C\mu^4 m r^2 \log^2 m,$$

where r is the rank of the original matrix $\mathbf{A}$. Let $\widehat{\mathbf{A}}$ be the recovered matrix by Algorithm 2 using similarity matrix $\mathbf{S}$ about nodes. Then with probability $1 - n^{-3}$, it holds:

$$\|\mathbf{A} - \widehat{\mathbf{A}}\|_{\mathrm{F}} \leq \frac{n}{m}\|\mathbf{A}_{\mathrm{E}}\|_{\mathrm{F}}^2 + \left(1 + n\sqrt{\frac{n}{m}}\right)\|\mathbf{A}_{\mathrm{E}}\|_{\mathrm{F}},$$

where $\mathbf{A}_{\mathrm{E}} = \sum_{i=1}^{r} \mathbf{a}_i^{\perp}\mathbf{a}_i^{\perp\top}$ is the orthogonal component of projection of adjacency matrix to the subspace of similarity matrix.

In above inequality, the parameter μ is known as incoherence [18, 93] which is the prevailing assumption in analysis of matrix completion algorithms. This parameter states that the singular vectors of $\mathbf{A}$ should be de-localized, in the sense of having small inner product with the standard basis, to make the full recovery possible which is formally defined below:

Definition 3.3.2. *An $n \times n$ matrix $\mathbf{M}$ with singular value decomposition $\mathbf{M} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$ is μ-incoherent if*

$$\max_{i,j} |\mathbf{U}_{ij}| \leq \frac{\sqrt{\mu}}{\sqrt{n}} \quad \textit{and} \quad \max_{i,j} |\mathbf{V}_{ij}| \leq \frac{\sqrt{\mu}}{\sqrt{n}}. \tag{3.14}$$

Before proving Theorem 3.3.1, let us pause to make some remarks concerning the result given above. First, one can observe that the recovery error is stated in terms of the norm of error matrix $\mathbf{A}_\mathrm{E}$ which captures the extent to which the similarity matrix fails to infer the links between users. Also, the error decreases by reducing the number of unobserved nodes as expected.

The following theorem which follows from [19] shows that under high incoherence and uniform sampling, solving Eq. (3.9) exactly recovers $\mathbf{O}$ with high probability.

Lemma 3.3.3. *Let $\mathbf{M}$ be an $m \times m$ matrix with rank r. In addition, assume $\mathbf{M}$ is μ-incoherent. Then there exists some constant C, such that if $C\mu^4 mr^2 \log^2 m$ and entries are uniformly sampled, then with probability at least $1 - n^{-3}$, $\mathbf{M}$ is the unique optimizer of Eq. (3.9).*

Lemma 3.3.3 indicates, if that the observed subgraph has enough links, i.e., $|\Omega| \geq C\mu^4 mr^2 \log^2 m$, that one can fully recover the subgraph. Therefore, the recovery error of induced submatrix is zero, provided the number of observed examples is sufficiently large and uniformly sampled.

3.3.1 Proof of Theorem 3.3.1

We now prove the result in Theorem 3.3.1.

Proof. (of Theorem 3.3.1) To prove the result, we first rewrite the error as:

$$\begin{aligned}
&\|\mathbf{A} - \widehat{\mathbf{A}}\|_{\mathrm{F}} \\
&= \|\mathbf{A}_* + \mathbf{A}_{\mathrm{E}} - \widehat{\mathbf{A}}\|_{\mathrm{F}} \leq \|\mathbf{A}_* - \widehat{\mathbf{A}}\|_{\mathrm{F}} + \|\mathbf{A}_{\mathrm{E}}\|_{\mathrm{F}} \\
&= \left\|\mathbf{U}_s \left(\sum_{i=1}^{r} \mathbf{b}_i \mathbf{b}_i^\top\right) \mathbf{U}_s^\top - \mathbf{U}_s \left(\sum_{i=1}^{r} \widehat{\mathbf{b}}_i \widehat{\mathbf{b}}_i^\top\right) \mathbf{U}_s^\top\right\|_{\mathrm{F}} + \|\mathbf{A}_{\mathrm{E}}\|_{\mathrm{F}} \\
&= \left\|\mathbf{U}_s \left(\sum_{i=1}^{r} \mathbf{b}_i \mathbf{b}_i^\top - \sum_{i=1}^{r} \widehat{\mathbf{b}}_i \widehat{\mathbf{b}}_i^\top\right) \mathbf{U}_s^\top\right\|_{\mathrm{F}} + \|\mathbf{A}_{\mathrm{E}}\|_{\mathrm{F}} \\
&= \left\|\sum_{i=1}^{r} \mathbf{b}_i \mathbf{b}_i^\top - \sum_{i=1}^{r} \widehat{\mathbf{b}}_i \widehat{\mathbf{b}}_i^\top\right\|_{\mathrm{F}} + \|\mathbf{A}_{\mathrm{E}}\|_{\mathrm{F}}
\end{aligned}$$

For simplicity, we define two matrices $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_r]$ and $\widehat{\mathbf{B}} = [\widehat{\mathbf{b}}_1, \widehat{\mathbf{b}}_2, \cdots, \widehat{\mathbf{b}}_r]$. Then the above inequality can be written as:

$$\|\mathbf{A} - \widehat{\mathbf{A}}\|_{\mathrm{F}} \leq \left\|\mathbf{B}^\top \mathbf{B} - \widehat{\mathbf{B}}^\top \widehat{\mathbf{B}}\right\|_{\mathrm{F}} + \|\mathbf{A}_{\mathrm{E}}\|_{\mathrm{F}}$$

To bound the first term in the R.H.S of above inequality, by a simple algebraic inequality, we write it as:

$$\begin{aligned}
\|\mathbf{B}^\top \mathbf{B} - \widehat{\mathbf{B}}^\top \widehat{\mathbf{B}}\|_{\mathrm{F}} &\leq \|\mathbf{B} - \widehat{\mathbf{B}}\|_{\mathrm{F}}^2 + 2\|(\mathbf{B} - \widehat{\mathbf{B}})\widehat{\mathbf{B}}\|_{\mathrm{F}} \\
&\leq \|\mathbf{B} - \widehat{\mathbf{B}}\|_{\mathrm{F}}^2 + 2\|\mathbf{B} - \widehat{\mathbf{B}}\|_{\mathrm{F}} \|\mathbf{B}\|_{\mathrm{F}}
\end{aligned}$$

Now we turn to bounding $\|\mathbf{B} - \widehat{\mathbf{B}}\|_{\mathrm{F}}$. To this end, we have:

$$\|\mathbf{B} - \widehat{\mathbf{B}}\|_{\mathrm{F}}^2 = \sum_{i=1}^{r} \|\mathbf{b}_i - \widehat{\mathbf{b}}_i\|_2^2 \tag{3.15}$$

First, we show that $\widehat{\mathbf{b}}_i = \mathbf{b}_i + \mathbf{M}\mathbf{a}_i^\perp$, where

$$\mathbf{M} = \left(\mathbf{U}_s^\top \mathbf{\Pi}^\top \mathbf{\Pi} \mathbf{U}_s\right)^\dagger \mathbf{U}_s^\top \mathbf{\Pi}^\top \mathbf{\Pi}.$$

We have,

$$\begin{aligned}
\widehat{\mathbf{b}}_i &= \left(\widehat{\mathbf{U}}_s^\top \widehat{\mathbf{U}}_s\right)^\dagger \widehat{\mathbf{U}}_s^\top \widehat{\mathbf{a}}_i = \left((\mathbf{\Pi}\mathbf{U}_s)^\top \mathbf{\Pi}\mathbf{U}_s\right)^\dagger (\mathbf{\Pi}\mathbf{U}_s)^\top \mathbf{\Pi}\mathbf{a}_i \\
&= \left(\mathbf{U}_s^\top \mathbf{\Pi}^\top \mathbf{\Pi}\mathbf{U}_s\right)^\dagger \mathbf{U}_s^\top \mathbf{\Pi}^\top \mathbf{\Pi}\left(\mathbf{a}_i^\parallel + \mathbf{a}_i^\perp\right) \\
&= \mathbf{b}_i + \left(\mathbf{U}_s^\top \mathbf{\Pi}^\top \mathbf{\Pi}\mathbf{U}_s\right)^\dagger \mathbf{U}_s^\top \mathbf{\Pi}^\top \mathbf{\Pi}\mathbf{a}_i^\perp \\
&= \mathbf{b}_i + \mathbf{M}\mathbf{a}_i^\perp .
\end{aligned}$$

By substituting this inequality in Eq. (3.15) we have,

$$\begin{aligned}
\|\mathbf{B} - \widehat{\mathbf{B}}\|_\mathrm{F}^2 &= \sum_{i=1}^r \|\mathbf{M}\mathbf{a}_i^\perp\|_2^2 \le \sum_{i=1}^r \lambda_{\max}(\mathbf{M})\|\mathbf{a}_i^\perp\|_2^2 \\
&= \lambda_{\max}(\mathbf{M}) \sum_{i=1}^r \|\mathbf{a}_i^\perp\|_2^2 = \lambda_{\max}(\mathbf{M})\|\mathbf{A}_\mathrm{E}\|_\mathrm{F}^2 \\
&= \lambda_{\max}(\widehat{\mathbf{U}}_s^{-1})\|\mathbf{A}_\mathrm{E}\|_\mathrm{F}^2
\end{aligned}$$

$\square$

Putting all the inequalities together and noting that $\|\mathbf{B}\|_\mathrm{F} \le n$, yields

$$\begin{aligned}
&\|\mathbf{A} - \widehat{\mathbf{A}}\|_\mathrm{F} \le \|\mathbf{B} - \widehat{\mathbf{B}}\|_\mathrm{F}^2 + 2\|\mathbf{B} - \widehat{\mathbf{B}}\|_\mathrm{F}\|\mathbf{B}\|_\mathrm{F} + \|\mathbf{A}_\mathrm{E}\|_\mathrm{F} \\
&\le \lambda_{\max}(\widehat{\mathbf{U}}_s^{-1})\|\mathbf{A}_\mathrm{E}\|_\mathrm{F}^2 \\
&\quad + 2n\sqrt{\lambda_{\max}(\widehat{\mathbf{U}}_s^{-1})\|\mathbf{A}_\mathrm{E}\|_\mathrm{F}^2} + \|\mathbf{A}_\mathrm{E}\|_\mathrm{F}
\end{aligned} \tag{3.16}$$

With the bounding of $\lambda_{\max}(\widehat{\mathbf{U}}_s^{-1})$ which relies on the following result [40].

Lemma 3.3.4. *Let $\mathbf{U}$ be an n by k matrix with orthonormal columns. Take μ to be the coherence of $\mathbf{U}$. Select $\epsilon \in (0,1)$ and a nonzero failure probability δ. Let $\mathbf{\Pi}$ be a random matrix distributed as the first p columns of a uniformly random permutation matrix of size n, where*

$$p \ge \frac{2\mu}{(1-\epsilon)^2} k \log\left(\frac{k}{\delta}\right)$$

Then with probability exceeding $1-\delta$*, the matrix* $\mathbf{U}^\top\mathbf{\Pi}$ *has full row rank and satisfies*

$$\left\|\left(\mathbf{U}^\top\mathbf{\Pi}\right)^\dagger\right\|_2^2 \leq \frac{n}{\epsilon p}.$$

Applying Lemma 3.3.4 to the inequality in Eq. (3.17) yields:

$$\|\mathbf{A}-\tilde{\mathbf{A}}\|_{\mathrm{F}} \leq \left(\frac{n}{\epsilon m} + 2n\sqrt{\frac{n}{\epsilon m}} + 1\right)\|\mathbf{A}_{\mathrm{E}}\|_{\mathrm{F}}^2 \tag{3.17}$$

By setting $\epsilon = \frac{1}{2}$ in Lemma 3.3.4 we get the desired bound stated in the Theorem 3.3.1.

Chapter 4

Semi-supervised Collaborative Ranking Algorithm

Existing collaborative ranking based recommender systems tend to perform best when there is enough observed ratings for each user and the observation is made completely at random. Under this setting recommender systems can properly suggest a list of recommendations according to the user interests. However, when the observed ratings are extremely sparse (e.g. in the case of cold-start users where no rating data is available), and are not sampled uniformly at random, existing ranking methods fail to effectively leverage side information to transduct the knowledge from existing ratings to unobserved ones. We propose a **semi-supervised collaborative ranking** model, dubbed $\texttt{S}^2\texttt{COR}$ [8], to improve the quality of cold-start recommendation. $\texttt{S}^2\texttt{COR}$ mitigates the sparsity issue by leveraging side information about *both observed and missing* ratings by collaboratively learning the ranking model. This enables it to deal with the case of missing data not at random, but to also effectively incorporate the available side information in transduction. We experimentally evaluated our proposed algorithm on a number of challenging real-world datasets and compared against state-of-the-art models for cold-start recommendation. We report significantly higher quality recommendations with our algorithm compared to the state-of-the-art.

4.1 The Setting

In this section we establish the notation used throughout the thesis and formally describe our problem setting.

Scalars are denoted by lower case letters and vectors by bold face lower case letters such as $\mathbf{u}$. We use bold face upper case letters such as $\mathbf{M}$ to denote matrices. The Frobenius norm of a matrix $\mathbf{M} \in \mathbb{R}^{n\times m}$ is denoted by $\|\mathbf{M}\|_{\mathrm{F}}$, i.e, $\|\mathbf{M}\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} |M_{ij}|^2}$ and its (i,j)th entry is denoted by $A_{i,j}$. The trace norm of a matrix is denoted by $\|\mathbf{M}\|_*$ which is defined as the sum of its singular values. The transpose of a vector and a matrix denoted by $\mathbf{u}^\top$ and $\mathbf{U}^\top$, respectively. We use $[n]$ to denote the set on integers $\{1, 2, \cdots, n\}$. The set of non-negative real numbers is denoted by $\mathbb{R}_+$. The indicator function is denoted by $\mathbb{I}[\cdot]$. For a vector $\mathbf{u} \in \mathbb{R}^p$ we use $\|\mathbf{u}\|_1 = \sum_{i=1}^{p} |\mathbf{u}_i|$, $\|\mathbf{u}\|_2 = \left(\sum_{i=1}^{p} |\mathbf{u}_i|^2\right)^{1/2}$, and $\|\mathbf{u}\|_\infty = \max_{1\leq i\leq p} \mathbf{u}_i$ to denote its ℓ_1, ℓ_2, and ℓ_∞ norms, respectively. The dot product between two vectors $\mathbf{u}$ and $\mathbf{u'}$ is denoted by either $\langle \mathbf{u}, \mathbf{u'}\rangle$ or $\mathbf{u}^\top\mathbf{u'}$.

In collaborative filtering we assume that there is a set of n users $\mathcal{U} = \{u_1, \cdots, u_n\}$ and a set of m items $\mathcal{I} = \{i_1, \cdots, i_m\}$ where each user u_i expresses opinions about a set of items. The rating information is summarized in an $n \times m$ matrix $\mathbf{R} \in \{-1, +1, ?\}^{n\times m}, 1 \leq i \leq n, 1 \leq j \leq m$ where the rows correspond to the users and the columns correspond to the items and (p, q)th entry is the rate given by user u_p to the item i_q. We note that the rating matrix is partially observed and it is sparse in most cases. We are mainly interested in recommending a set of items for an active user such that the user has not rated these items before.

4.2 Transductive Collaborating Ranking

We now turn our attention to the main thrust of the thesis where we present our transductive collaborative ranking algorithm with accuracy at top by exploiting the features of unrated data. We begin with the basic formulation and then extend it to incorporate the unrated items. The pseudo-code of the resulting learning algorithm is provided in Algorithm 3.

4.2.1 A basic formulation

We consider a ranking problem, where, given a set of users $\mathcal{U}$ and known user feedback on a set of items $\mathcal{I}$, the goal is to generate rankings of unobserved items, adapted to each of the users' preferences. Here we consider the bipartite setting in which items are either relevant (positive) or irrelevant (negative). Many ranking methods have been developed for bipartite ranking, and most of them are essentially based on pairwise ranking. These algorithms reduce the ranking problem into a binary classification problem by treating each relevant/irrelevant instance pair as a single object to be classified [**?**].

As mentioned above, most research has concentrated on the rating prediction problem in CF where the aim is to accurately predict the ratings for the unrated items for each user. However, most applications that use CF typically aim to recommend only a small ranked set of items to each user. Thus rather than concentrating on rating prediction we instead approach this problem from the ranking viewpoint where the goal is to rank the unrated items in the order of relevance to the user. Moreover, it is desirable to concentrate aggressively on top portion of the ranked list to include mostly relevant items and push irrelevant items down from the top. Specifically, we propose an algorithm that maximizes the number of relevant items which are pushed to the absolute top of the list by utilizing

Algorithm 3 S^{2}COR

```
1: input: $\lambda \in \mathbb{R}_+$: the regularization parameter, and $\{\eta_t\}_{t\geq 1}$: the sequence of scalar step sizes
2: Initialize $\mathbf{W}_0 \in \mathbb{R}^{n\times d}$
3: Choose an appropriate step size
4: for $t = 1, \ldots, T$ do
5:     Compute the sub-gradient of $\mathbf{G}_t \in \partial\mathcal{L}(\mathbf{W}_t)$ using Eq. (4.11)
6:     $[\mathbf{U}_t, \mathbf{\Sigma}_t, \mathbf{V}_t] \leftarrow \mathtt{SVD}(\mathbf{W}_{t-1} - \frac{1}{\eta_{t-1}}\mathbf{G}_t))$
7:     $\mathbf{W}_t \leftarrow \mathbf{U}_t \left[\mathbf{\Sigma} - \frac{\lambda}{\eta_{t-1}}\mathbf{I}\right]_+ \mathbf{V}_t^\top$
8: end for
9: output:
```

the P-Norm Push ranking measure which is specially designed for this purpose [99] .

For simplicity of exposition, let us first consider the ranking model for a single user u. Let $\mathcal{X}^+ = \{\mathbf{x}_1^+, \cdots, \mathbf{x}_{n_+}^+\}$ and $\mathcal{X}^- = \{\mathbf{x}_1^-, \cdots, \mathbf{x}_{n_-}^-\}$ be the set of feature vectors of n_+ relevant and n_- irrelevant items to user u, respectively. We consider linear ranking functions where each item features vector $\mathbf{x} \in \mathbb{R}^d$ is mapped to a score $\mathbf{w}^\top\mathbf{x}$. The goal is to find parameters $\mathbf{w}$ for each user such that the ranking function best captures past feedback from the user. The goal of ranking is to maximize the number of relevant items ranked above the highest-ranking irrelevant item. We cast this idea for each user u individually into the following optimization problem:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n^+} \sum_{i=1}^{n^+} \mathbb{I}\left[\langle \mathbf{w}, \mathbf{x}_i^+\rangle \leq \max_{1\leq j\leq n^-} \langle \mathbf{w}, \mathbf{x}_j^-\rangle\right] \tag{4.1}$$

where $\mathbb{I}[\cdot]$ is the indicator function which returns 1 when the input is true and 0 otherwise, n^+ and n^- are the the number of relevant and irrelevant items to user u, respectively.

Let us now derive the general form of our objective. We hypothesize that most users base their decisions about items based on a number of latent features about the items. In order to uncover these latent feature dimensions, we impose a low-rank constraint on the set of parameters for all users. To this end, let $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_n]^\top \in \mathbb{R}^{n \times d}$ denote the matrix of all parameter vectors for n users. Let $\mathcal{I}_i^+ \subseteq \{1, 2, \ldots, m\}$ and $\mathcal{I}_i^- \subseteq \{1, 2, \ldots, m\}$ be the set of relevant and irrelevant items of ith user, respectively. The overall objective for all users is formulated as follows:

$$\begin{aligned}\mathcal{F}(\mathbf{W}) = {} & \lambda \|\mathbf{W}\|_* \\ & + \sum_{i=1}^{n} \left(\frac{1}{|\mathcal{I}_i^+|} \sum_{j \in \mathcal{I}_i^+} \mathbb{I} \left[\langle \mathbf{w}_i, \mathbf{x}_j \rangle \leq \max_{k \in \mathcal{I}_i^-} \langle \mathbf{w}_i, \mathbf{x}_k \rangle \right] \right),\end{aligned} \tag{4.2}$$

where $\|\cdot\|_*$ is the trace norm (also known as nuclear norm) which is the sum of the singular values of the input matrix.

The objective in Eq. (4.2) is composed of two terms. The first term is the regularization term and is introduced to capture the factor model intuition discussed above. The premise behind a factor model is that there is only a small number of factors influencing the preferences, and that a user's preference vector is determined by how each factor applies to that user. Therefore, the parameter vectors of all users must lie in a low-dimensional subspace. Trace-norm regularization is a widely-used and successful approach for collaborative filtering and matrix completion. The trace-norm regularization is well-known to be a convex surrogate to the matrix rank, and has repeatedly shown good performance in practice [110, 20]. The second term is introduced to push the relevant items of each user to the top of the list when ranked based on the parameter vector of the user and features of items.

The above optimization problem is intractable due to the non-convex indicator function. To design practical learning algorithms, we replace the indicator function in (4.2) with its convex surrogate. To this end, define the convex loss function $\ell : \mathbb{R} \mapsto \mathbb{R}_+$ as $\ell(x) =$

$[1-x]_+$. This is the widely used hinge loss in SVM classification (see e.g., [15]) [1]. This loss function reflects the amount by which the constraints are not satisfied. By replacing the non-convex indicator function with this convex surrogate leads to the following tractable convex optimization problem:

$$\begin{aligned}\mathcal{F}(\mathbf{W}) = &\ \lambda\|\mathbf{W}\|_* \\ &+ \sum_{i=1}^{n}\left(\frac{1}{|\mathcal{I}_i^+|}\sum_{j\in\mathcal{I}_i^+}\ell\left(\langle\mathbf{w}_i,\mathbf{x}_j\rangle - \|\mathbf{X}_i^-\mathbf{w}_i\|_\infty\right)\right)\end{aligned} \tag{4.3}$$

where $\mathbf{X}_i^- = [\mathbf{x}_1,\ldots,\mathbf{x}_{n_i^-}]^\top$ is the matrix of features of n_i^- irrelevant items in $\mathcal{I}_i^-$ and $\|\cdot\|_\infty$ is the max norm of a vector.

4.2.2 Semi-supervised collaborative ranking

In this part, we extend the proposed ranking idea to learn both from rated as well as unrated items. The motivation of incorporating unrated items comes from the following key observations. First, we note that commonly there is a small set of rated (either relevant or irrelevant) items for each user and a large number of unrated items. As it can be seen from Eq. (4.2), the unrated items do not play any role in learning the model for each user as the learning is only based on the pair of rated items. When the feature information for items is available, it would be very helpful if one can leverage such unrated items in the learning-to-rank process to effectively leverage the available side information. By leveraging both types of rated and unrated items, we can compensate for the lack of rating data. Second, the

[1] We note that other convex loss functions such as exponential loss $\ell(x) = \exp(-x)$, and logistic loss $\ell(x) = \log(1+\exp(-x))$ also can be used as the surrogates of indicator function, but for the simplicity of derivation we only consider the hinge loss here.

non-randomness in observing the observed ratings creates a bias in learning the model that may degrade the resulting recommendation accuracy. Therefore, finding a precise model to reduce the effect of bias introduced by non-random missing ratings seems essential.

To address these two issues, we extend the basic formulation in Eq. (4.2) to incorporate items with missing ratings in ranking of items for individual users. A conservative solution is to push the items with unknown ratings to the middle of ranked list, i.e., after the relevant and before the irrelevant items. To do so, let $\mathcal{I}_i^{\circ} = \mathcal{I} \setminus \left(\mathcal{I}_i^{+} \cup \mathcal{I}_i^{-}\right)$ denote the set of items unrated for user $i \in \mathcal{U}$. We introduce two extra terms in the objective in Eq. (4.2) to push the unrated items $\mathcal{I}_{\circ}^{i}$ below the relevant items and above the irrelevant items, which yilelds the following objective:

$$\begin{aligned}\mathcal{L}(\mathbf{w}) = \quad & \frac{1}{|\mathcal{I}_i^{+}|} \sum_{i \in \mathcal{I}_i^{+}} \ell\left(\langle \mathbf{w}, \mathbf{x}_i \rangle \leq \max_{j \in \mathcal{I}_i^{-}} \langle \mathbf{w}, \mathbf{x}_j \rangle\right) \\ + \quad & \frac{1}{|\mathcal{I}_i^{+}|} \sum_{i \in \mathcal{I}_i^{+}} \ell\left(\langle \mathbf{w}, \mathbf{x}_i \rangle \leq \max_{j \in \mathcal{I}_i^{\circ}} \langle \mathbf{w}, \mathbf{x}_j \rangle\right) \\ + \quad & \frac{1}{|\mathcal{I}_i^{\circ}|} \sum_{i \in \mathcal{I}_i^{\circ}} \ell\left(\langle \mathbf{w}, \mathbf{x}_i \rangle \leq \max_{j \in \mathcal{I}_i^{-}} \langle \mathbf{w}, \mathbf{x}_j \rangle\right)\end{aligned} \tag{4.4}$$

Equipped with the objective of individual users, we now turn to the final collaborating ranking objective as:

$$
\begin{aligned}
\mathcal{F}(\mathbf{W}) = \lambda\|\mathbf{W}\|_* &\\
+ \sum_{i=1}^{n} &\left(\frac{1}{|\mathcal{I}_i^+|} \sum_{j\in\mathcal{I}_i^+} \ell\left(\langle \mathbf{w}_i, \mathbf{x}_j\rangle - \|\mathbf{X}_i^- \mathbf{w}_i\|_\infty\right)\right) \\
+ \sum_{i=1}^{n} &\left(\frac{1}{|\mathcal{I}_i^+|} \sum_{j\in\mathcal{I}_i^+} \ell\left(\langle \mathbf{w}_i, \mathbf{x}_j\rangle - \|\mathbf{X}_i^\circ \mathbf{w}_i\|_\infty\right)\right) \\
+ \sum_{i=1}^{n} &\left(\frac{1}{|\mathcal{I}_i^\circ|} \sum_{j\in\mathcal{I}_i^\circ} \ell\left(\langle \mathbf{w}_i, \mathbf{x}_j\rangle - \|\mathbf{X}_i^- \mathbf{w}_i\|_\infty\right)\right),
\end{aligned}
\tag{4.5}
$$

where $\mathbf{X}_i^\circ = [\mathbf{x}_1, \ldots, \mathbf{x}_{n_i^\circ}]^\top$ is the matrix of n_i° unrated items in $\mathcal{I}_i^\circ$.

Remark 4.2.1. *We emphasize that beyond the accuracy considerations of push norm at top of the list, the push norm has a clear advantage to the pairwise ranking models from a computational point of view. In particular, the push norm has a linear $O(m)$ dependency on the number of items which is quadratic $O(m^2)$ for pairwise ranking models.*

4.3 The Optimization

We now turn to solving the optimization problem in (4.5). We start by discussing a gradient descent method with shrinkage operator followed by its accelerated version, and then propose a non-convex formulation with alternative minimization for more effective optimization of objective in S^{2}COR.

4.3.1 Gradient descent with shrinkage operator

Due to the presence of trace norm of the parameters matrix, this objective function falls into the general category of composite optimization, which can be solved by stochastic gradient or gradient descent methods. In this part we propose a projected gradient decent method to solve the optimization problem. First we write the objective as:

$$\min_{\mathbf{W}\in\mathbb{R}^{n\times d}} \mathcal{F}(\mathbf{W}) = \lambda\|\mathbf{W}\|_* + \mathcal{L}(\mathbf{W}), \tag{4.6}$$

where $\mathcal{L}(\mathbf{W}) = \sum_{i=1}^{n} \mathcal{L}(\mathbf{w}_i)$.

A simple way to solving the above optimization problem is gradient descent algorithm [84], which needs to evaluate the gradient of objective at each iteration. To deal with the non-smooth trace norm $\|\mathbf{W}\|_*$ in the objective, we first note that the optimization problem in Eq. (4.6) can be reformulated under the framework of proximal regularization or composite gradient mapping [84]. By taking advantage of the composite structure it is possible to retain the same convergence rates of the gradient method for the smooth optimization problems. In particular, the optimization problem in (4.6) can be solved iteratively by:

$$\begin{aligned}\mathbf{W}_t = \arg\min_{\mathbf{W}} \mathcal{L}(\mathbf{W}_{t-1}) + \mathtt{tr}\left((\mathbf{W}-\mathbf{W}_{t-1})^\top \nabla\mathcal{L}(\mathbf{W}_{t-1})\right) \\ + \frac{\eta_t}{2}\|\mathbf{W}-\mathbf{W}_{t-1}\|_{\mathrm{F}}^2 + \lambda\|\mathbf{W}\|_*,\end{aligned} \tag{4.7}$$

where $\{\eta_t\}_{t\geq 1}$ is a sequence of scalar step sizes and $\mathtt{tr}(\cdot)$ is the trace of input matrix.

By ignoring the constant terms, Eq. (4.7) can also be rewritten as:

$$\frac{\eta_t}{2}\left\|\mathbf{W} - \left(\mathbf{W}_{t-1} - \frac{1}{\eta_t}\nabla\mathcal{L}(\mathbf{W}_{t-1})\right)\right\|_{\mathrm{F}}^2 + \lambda\|\mathbf{W}\|_*. \tag{4.8}$$

We use the singular value shrinkage operator introduced in [18] to find the optimal solution to Eq. (4.8). To this end, consider the singular value decomposition (SVD) of a matrix $\mathbf{M} \in \mathbb{R}^{n\times d}$ of rank r as $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*, \quad \mathbf{\Sigma} = \operatorname{diag}(\{\sigma_i\}_{1\le i\le r})$, where $\mathbf{U}$ and $\mathbf{V}$ are respectively $n \times r$ and $d \times r$ matrices with orthonormal columns, and the singular values σ_i are positive. For a scalar $\tau \in \mathbb{R}_+$, define the singular value shrinkage operator $\mathscr{P}_\tau$ as:

$$\mathscr{P}_\tau(\mathbf{M}) := \mathbf{U}\mathscr{P}_\tau(\mathbf{\Sigma})\mathbf{V}^*, \quad \mathscr{P}_\tau(\mathbf{\Sigma}) = \operatorname{diag}([\sigma_i - \tau]_+\}), \tag{4.9}$$

where $[x]_+$ is the positive part of x, namely, $[x]_+ = \max(0, x)$. The shrinkage operator basically applies a soft-thresholding rule to the singular values of $\mathbf{M}$, effectively shrinking these towards zero.

Theorem 4.3.1 (Theorem 2.1, [18])**.** *For each $\tau \ge 0$ and $\mathbf{W} \in \mathbb{R}^{n\times d}$, the singular value shrinkage operator* (**??**) *obeys*

$$\mathscr{P}_\tau(\mathbf{W}) = \arg\min_{\mathbf{X}} \left\{ \frac{1}{2}\|\mathbf{X} - \mathbf{W}\|_{\mathrm{F}}^2 + \tau\|\mathbf{X}\|_* \right\}. \tag{4.10}$$

The above theorem shows that the singular value shrinkage operator of a matrix is the solution to Eq. (4.10). Equipped with this result, the optimization problem in Eq. (4.8) can be solved by first computing the SVD of updated $\mathbf{W}_{t-1}$ and then applying soft thresholding on the singular values as:

$$\mathbf{W}_t = \mathscr{P}_{\frac{\lambda}{\eta_{t-1}}} \left(\mathbf{W}_{t-1} - \frac{1}{\eta_{t-1}} \nabla\mathcal{L}(\mathbf{W}_{t-1}) \right)$$

Now we need to evaluate the gradient of $\mathcal{L}(\mathbf{W})$ at $\mathbf{W}_{t-1}$. The convex function $\mathcal{L}(\mathbf{W})$ is not differentiable, so we use its subgradient in updating the solutions which can be computed for ith parameter vector $\mathbf{w}_i$, as follows:

$$\begin{aligned}
\mathbf{g}_i &= \partial \mathcal{L} / \partial \mathbf{w}_i \\
&= \sum_{j \in \mathcal{I}_i^+} \mathbb{I}\left[\langle \mathbf{w}_i, \mathbf{x}_j \rangle - \|\mathbf{X}_i^- \mathbf{w}_i\|_\infty \leq 1\right] \left(\partial \|\mathbf{X}_i^- \mathbf{w}_i\|_\infty - \mathbf{x}_j\right) \\
&+ \sum_{j \in \mathcal{I}_i^+} \mathbb{I}\left[\langle \mathbf{w}_i, \mathbf{x}_j \rangle - \|\mathbf{X}_i^\circ \mathbf{w}_i\|_\infty \leq 1\right] \left(\partial \|\mathbf{X}_i^\circ \mathbf{w}_i\|_\infty - \mathbf{x}_j\right) \\
&+ \sum_{j \in \mathcal{I}_i^\circ} \mathbb{I}\left[\langle \mathbf{w}_i, \mathbf{x}_j \rangle - \|\mathbf{X}_i^- \mathbf{w}_i\|_\infty \leq 1\right] \left(\partial \|\mathbf{X}_i^- \mathbf{w}_i\|_\infty - \mathbf{x}_j\right),
\end{aligned} \tag{4.11}$$

where $\partial \|\mathbf{X}_i^- \mathbf{w}_i\|_\infty$ is the subdifferential of the function $\|\mathbf{X}_i^- \mathbf{w}_i\|_\infty$ at point $\mathbf{w}_i$. Since the subdifferential of the maximum of functions is the convex hull of the union of subdifferentials of the active functions at point $\mathbf{w}$ [83], we have:

$$\begin{aligned}
\partial \|\mathbf{X}_i^- \mathbf{w}_i\|_\infty &= \partial \max_{1 \leq j \in \mathcal{I}_i^-} \langle \mathbf{w}_i, \mathbf{x}_j \rangle \\
&= \operatorname{conv}\left\{\mathbf{x}_j | \langle \mathbf{w}_i, \mathbf{x}_j \rangle = \|\mathbf{X}_i^- \mathbf{w}_i\|_\infty, j \in \mathcal{I}_j^- \right\}.
\end{aligned}$$

Then, $\mathbf{G} = [\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_n]^\top \in \mathbb{R}^{n \times d}$ is a subgradient at $\mathbf{W}$, i.e. $\mathbf{G} \in \partial \mathcal{L}(\mathbf{W})$.

Remark 4.3.2. *We note that here, for the ease of exposition, we only consider the non-differentiable convex hinge loss as the surrogate of non-convex due to its computational virtues. However, by using other smooth convex surrogate losses such as smoothed hinge loss or exponential loss, one can apply the accelerated gradient descent methods [83] to solve the optimization problem which results in significantly faster convergence rate compared to the naive gradient descent (i.e, $O(1/\sqrt{\epsilon})$ convergence rate for accelerated method compared to the $O(1/\epsilon^2)$ rate for gradient descent for non-smooth optimization, where ϵ is the target* *accuracy).*

4.3.2 Efficient optimization by dropping convexity

The main computational cost in each iteration of the $\texttt{S}^2\texttt{COR}$ algorithm lies in computing the SVD decomposition of $\mathbf{W}_k$. An alternative cheaper way to solving the optimization problem in Eq. (4.6) is as follows. For a fixed rank of the target parameter matrix $\mathbf{W}$, say k, one can decompose it as $\mathbf{W} = \mathbf{U}\mathbf{V}$. From the equivalence relation between trace norm and the Frobenius of its components in decomposition,

$$\|\mathbf{W}\|_* = \min_{\substack{\mathbf{U}\in\mathbb{R}^{n\times k},\mathbf{V}\in\mathbb{R}^{m\times k}\\ \mathbf{W}=\mathbf{U}\mathbf{V}^\top}} \frac{1}{2}\left(\|\mathbf{U}\|_\mathrm{F}^2 + \|\mathbf{V}\|_\mathrm{F}^2\right)$$

we can write the objective in terms of these low rank components as:

$$\min_{\mathbf{U}\in\mathbb{R}^{n\times k},\mathbf{V}\in\mathbb{R}^{m\times k}} \frac{\lambda}{2}\left(\|\mathbf{U}\|_\mathrm{F}^2 + \|\mathbf{V}\|_\mathrm{F}^2\right) + \mathcal{L}(\mathbf{U}\mathbf{V}), \tag{4.12}$$

These factored optimization problem does not have the explicit trace norm regularization. However, the new formulation is non-convex and potentially subject to stationary points that are not globally optimal. However, despite its non-convexity, the formulation in Eq. (4.12) is competitive as compared to trace-norm minimization, while scalability is much better. In particular, the objective is not jointly convex in both $\mathbf{U}$ and $\mathbf{V}$ but it is convex in each of them fixing the other one. Therefore, to find a local solution one can stick to the standard

Algorithm 4 $\texttt{S}^2\texttt{COR+}$

```
1: input: $\gamma, \lambda \in \mathbb{R}_+$: the regularization parameters, $\{\eta_t\}_{t\geq 1}$: the sequence of scalar step
   sizes, and $\mathbf{S} \in \mathbb{R}^{n\times n}$: the similarity matrix of usets
2: Initialize $\mathbf{W}_0 \in \mathbb{R}^{n\times d}$
3: Choose an appropriate step size
4: for $t = 1, \ldots, T$ do
5:     Compute the sub-gradient of $\mathbf{G}_t \in \partial\mathcal{L}(\mathbf{W}_t)$ using Eq. (4.11)
6:     Compute $\widehat{\mathbf{G}}_t = \mathbf{G}_t + 2\gamma\mathbf{X}\mathbf{X}^\top\mathbf{W}\mathbf{L}$
7:     $[\mathbf{U}_t, \mathbf{\Sigma}_t, \mathbf{V}_t] \leftarrow \texttt{SVD}(\mathbf{W}_{t-1} - \frac{1}{\eta_{t-1}}\widehat{\mathbf{G}}_t))$
8:     $\mathbf{W}_t \leftarrow \mathbf{U}_t\left[\mathbf{\Sigma} - \frac{\lambda}{\eta_{t-1}}\mathbf{I}\right]_+\mathbf{V}_t^\top$
9: end for
10: output:
```

gradient descent method to find a solution in an iterative manner as follows:

$$\mathbf{U}_{t+1} \leftarrow (1-\lambda\eta_t)\mathbf{U}_t - \eta_t\nabla_{\mathbf{U}}\mathcal{L}|_{\mathbf{U}=\mathbf{U}_t,\mathbf{V}=\mathbf{V}_t},$$

$$\mathbf{V}_{t+1} \leftarrow (1-\lambda\eta_t)\mathbf{V}_t - \eta_t\nabla_{\mathbf{V}}\mathcal{L}|_{\mathbf{U}=\mathbf{U}_t,\mathbf{V}=\mathbf{V}_t}.$$

Remark 4.3.3. *It is remarkable that the large number of users or items may cause computational problems in solving the optimization problem using GD method. The reason is essentially the fact that computing the gradient at each iteration requires to go through all the users and compute the gradient for pair of items. To alleviate this problem one can utilize stochastic gradient method [82] to solve the optimization problem. The main idea is to choose a fixed subset of pairs for gradient computation instead of all pairs at each iteration or a sample a user at random for gradient computation instead of including all users. We note that this strategy generates unbiased estimates of the true gradient and makes each iteration of algorithm computationally more efficient compared to the full gradient counterpart.*

Chapter 5

PushTrust: An Efficient Recommendation Algorithm by Leveraging Trust and Distrust Relations

The significance of social-enhanced recommender systems is increasing, along with its practicality, as online reviews, ratings, friendship links, and follower relationships are increasingly becoming available. In recent years, there has been an upsurge of interest in exploiting social information, such as trust and distrust relations in recommendation algorithms. The goal is to improve the quality of suggestions and mitigate the data sparsity and the cold-start users problems in existing systems. In this paper, we introduce a general *collaborative social ranking* model to rank the latent features of users extracted from rating data based on the social context of users. In contrast to existing social regularization methods, the proposed framework is able to simultaneously leverage trust, distrust, and neutral relations, and has a linear dependency on the social network size. By integrating the ranking based social regularization idea into the matrix factorization algorithm, we propose a novel recommendation algorithm, dubbed `PushTrust`. Our experiments on the Epinions dataset demonstrate that collaboratively ranking the latent features of users by exploiting trust and distrust relations leads to a substantial increase in performance, and to effectively deal with cold-start users problem.

5.1 The Setting

In this section we establish the notation used throughout the paper and formally describe our problem setting.

We adopt the following notation throughout the paper. Scalars are denoted by lower case letters and vectors by bold face lower case letters such as $\mathbf{u}$. We use bold face upper case letters such as $\mathbf{A}$ to denote matrices. The Frobenius norm of a matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$ is denoted by $\|\mathbf{A}\|_{\mathrm{F}}$, i.e, $\|\mathbf{A}\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}|\mathbf{A}_{ij}|^2}$ and its (i, j)th entry is denoted by $\mathbf{A}_{i,j}$. The transpose of a vector and a matrix denoted by $\mathbf{u}^\top$ and $\mathbf{U}^\top$, respectively. We use $[n]$ to denote the set on integers $\{1, 2, \cdots, n\}$. The set of non-negative real numbers is denoted by $\mathbb{R}_+$. The indicator function is denoted by $\mathbb{I}[\cdot]$. For a vector $\mathbf{u} \in \mathbb{R}^p$ we use $\|\mathbf{u}\|_1 = \sum_{i=1}^{p}|\mathbf{u}_i|$, $\|\mathbf{u}\|_2 = \left(\sum_{i=1}^{p}|\mathbf{u}_i|^2\right)^{1/2}$, and $\|\mathbf{u}\|_\infty = \max_{1\leq i\leq p}\mathbf{u}_i$ to denote its ℓ_1, ℓ_2, and ℓ_∞ norms, respectively. The dot product between two vectors $\mathbf{u}$ and $\mathbf{u}'$ is denoted by either $\langle\mathbf{u}, \mathbf{u}'\rangle$ or $\mathbf{u}^\top\mathbf{u}'$. We use $[z]_+ = \max(0, z)$ to denote the positive component of a scalar.

5.1.1 Matrix factorization for recommendation

In collaborative filtering we assume that there is a set of n users $\mathcal{U} = \{u_1, \cdots, u_n\}$ and a set of m items $\mathcal{I} = \{i_1, \cdots, i_m\}$ where each user u_i expresses opinions about a set of items. In this paper, we assume opinions are expressed through an explicit numeric rating (e.g., scale from one to five), but other rating methods such as hyperlink clicks are possible as well. We are mainly interested in recommending a set of items for an active user such that the user has not rated these items before. To this end, we are aimed at learning a model from the existing ratings, i.e., *offline phase*, and then use the learned model to generate recommendations for active users, i.e., *online phase*. The rating information is summarized in an $n \times m$ matrix $\mathbf{R} \in \mathbb{R}^{n \times m}, 1 \leq i \leq n, 1 \leq j \leq m$ where the rows correspond to the users and the columns correspond to the items and (p, q)th entry is the rate given by user u_p to the item i_q. We note that the rating matrix is partially observed and it is sparse in most cases.

An efficient and effective approach to recommender systems is to factorize the user-item rating matrix $\mathbf{R}$ by a multiplicative of k-rank matrices $\mathbf{R} \approx \mathbf{U}\mathbf{V}^\top$, where $\mathbf{U} \in \mathbb{R}^{n \times k}$ and $\mathbf{V} \in \mathbb{R}^{m \times k}$ utilize the factorized user-specific and item-specific matrices, respectively, to make further missing data prediction. There are two basic formulations to solve this problem: these are optimization based (see e.g., [97, 65, 71, 58]) and probabilistic [80]. Let $\Omega_{\mathbf{R}}$ be the set of observed ratings in the user-item matrix $\mathbf{R} \in \mathbb{R}^{n \times m}$, i.e., $\Omega_{\mathbf{R}} = \{(i, j) \in [n] \times [m] : \mathbf{R}_{ij} \text{ has been observed}\}$, where n is the number of users and m is the number of items to be rated. In optimization based matrix factorization, the goal is to learn the latent matrices $\mathbf{U}$ and $\mathbf{V}$ by solving the following optimization problem:

$$\mathcal{F}(\mathbf{U}, \mathbf{V}) = \frac{1}{2} \sum_{(i,j) \in \Omega_{\mathbf{R}}} \left(\mathbf{R}_{i,j} - \mathbf{u}_i^\top \mathbf{v}_j\right)^2 + \lambda_{\mathbf{U}} \|\mathbf{U}\|_F^2 + \lambda_{\mathbf{V}} \|\mathbf{V}\|_F^2 \tag{5.1}$$

The optimization problem in Eq. (5.1) constitutes of three terms: the first term aims to minimize the inconsistency between the observed entries and their corresponding value obtained by the factorized matrices. The last two terms regularize the latent matrices for users and items, respectively. The parameters $\lambda_{\mathbf{U}}$ and $\lambda_{\mathbf{V}}$ are regularization parameters that are introduced to control the regularization of latent matrices $\mathbf{U}$ and $\mathbf{V}$, respectively.

5.1.2 Matrix factorization with social regularization

In this section we briefly review the existing factorization methods that are capable of exploiting social context of users. The general theme in these methods is to regularize the latent features of users based on their social context.

To incorporate the trust relations in the optimization problem formulated in Eq. (5.1), few papers [70, 52, 72, 65] proposed the social regularization method which aims at keeping the latent vector of each user similar to his/her trusted neighbors in the social network. The proposed models force the user feature vectors to be close to those of their neighbors to be able to learn the latent features of users with no or very few ratings [52]. More specifically, the optimization problem becomes as:

$$\begin{aligned}\mathcal{F}(\mathbf{U},\mathbf{V}) = \frac{1}{2}\sum_{(i,j)\in\Omega_{\mathbf{R}}}\left(\mathbf{R}_{ij}-\mathbf{u}_i^\top\mathbf{v}_j\right)^2 + \frac{\lambda_{\mathbf{U}}}{2}\|\mathbf{U}\|_{\mathrm{F}} + \frac{\lambda_{\mathbf{V}}}{2}\|\mathbf{V}\|_{\mathrm{F}} \\ + \frac{\lambda_{\mathbf{S}}}{2}\sum_{i=1}^{n}\left\|\mathbf{u}_i - \frac{1}{|\mathcal{N}^+(i)|}\sum_{j\in\mathcal{N}^+(i)}\mathbf{u}_j\right\|,\end{aligned} \tag{5.2}$$

where $\lambda_{\mathbf{S}}$ is the social regularization parameter and $\mathcal{N}^+(i)\subseteq\mathcal{V}$ is the subset of users who have trust relationships with ith user in the social graph.

In a similar way one can exploit the distrust relations by forcing the latent features of each user to be as far as possible from the average of latent features of his/her distrusted friends formulated by:

$$\begin{aligned}\mathcal{F}(\mathbf{U},\mathbf{V}) = \frac{1}{2}\sum_{(i,j)\in\Omega_{\mathbf{R}}}\left(\mathbf{R}_{ij}-\mathbf{u}_i^\top\mathbf{v}_j\right)^2 + \frac{\lambda_{\mathbf{U}}}{2}\|\mathbf{U}\|_{\mathrm{F}} + \frac{\lambda_{\mathbf{V}}}{2}\|\mathbf{V}\|_{\mathrm{F}} \\ - \frac{\lambda_{\mathbf{S}}}{2}\sum_{i=1}^{n}\left\|\mathbf{u}_i - \frac{1}{|\mathcal{N}^-(i)|}\sum_{j\in\mathcal{N}^-(i)}\mathbf{u}_j\right\|,\end{aligned} \tag{5.3}$$

where $\mathcal{N}^{-}(i) \subseteq \mathcal{V}$ is the subset of users who have distrust relationships with ith user in the social graph.

It is remarkable that while intuition and experimental evidence indicate that trust is somewhat transitive, distrust is certainly not transitive. Therefore, the distrust can not be considered as the negative of trust relations in social network [116]. As a result, simultaneous exploitation of trust and distrust relationships is not as straightforward as exploiting either trust or distrust relations. In a very recent work [35], the authors proposed the first matrix factorization based recommender algorithm that is able to exploit both types of relationships in factorization. The main idea is to learning the latent features for each user u such that users trusted by u in the social network (with positive edges) are close and users which are distrusted by u (with negative edges) are more distant. Learning latent features in this way basically induces a ranking of latent features of each user's neighbors where the trusted friends appear on the top of the list and distrusted friends move to the bottom of the list.

To formalize the ranking idea in learning the latent features, let $\Omega_{\mathbf{S}}$ denote the set of triplets (i, j, k) where ith user trust jth user and distrusts kth user in the social relations, i.e., $\Omega_{\mathbf{S}} = \Big\{(i, j, k) \in [n] \times [n] \times [n] : \mathbf{S}_{ij} = 1 \;\;\&\;\; \mathbf{S}_{ik} = -1\Big\}$. Then the objective of factorization becomes:

$$\begin{aligned}\mathcal{F}(\mathbf{U}, \mathbf{V}) = \frac{1}{2} \sum_{(i,j)\in\Omega_{\mathbf{R}}} \left(\mathbf{R}_{ij} - \mathbf{u}_i^{\top}\mathbf{v}_j\right)^2 + \frac{\lambda_{\mathbf{U}}}{2}\|\mathbf{U}\|_{\mathrm{F}} + \frac{\lambda_{\mathbf{V}}}{2}\|\mathbf{V}\|_{\mathrm{F}} \\ + \frac{\lambda_{\mathbf{S}}}{2} \sum_{(i,j,k)\in\Omega_{\mathbf{S}}} \left[1 - \|\mathbf{u}_i - \mathbf{u}_j\|^2 + \|\mathbf{u}_i - \mathbf{u}_k\|^2\right]_{+},\end{aligned} \tag{5.4}$$

where the third term is introduced to regularize the latent features of users who violate the ranking constraints.

As mentioned earlier, the simultaneous exploitation of trust and distrust has been investigated in memory based recommender systems too [118, 115]. The main rational behind the algorithm proposed in [118] is to employ the distrust information to debug or filter out the users' propagated web of trust. It is also has been realized that the debugging methods must exhibit a moderate behavior in order to be effective. [115] addressed the problem of considering the length of the paths that connect two users for computing trust-distrust between them, according to the concept of *trust decay*. This work also introduced several aggregation strategies for trust scores with variable path lengths.

5.2 The PushTrust Algorithm

In this section we introduce the collaborative social ranking framework for social recommendation which, in contrast to social regularization based methods, is able to incorporate both trust and distrust relationships in the social network along with the partially observed rating matrix.

5.2.1 Collaborative social ranking

Before delving into the mathematical formulation, we first discuss the main idea behind the proposed `PushTrust` algorithm. As discussed earlier, to incorporate the social context of users in the factorization model, the appropriate latent features for users must be found such that each user is brought closer to the users she/he trusts and separated from the users that she/he distrusts and have different interests. We note that simply incorporating this idea in matrix factorization, by naively penalizing the similarity of each user's latent features to his distrusted friends' latent features, fails to reach the desired goal. The main reason is that distrust is not as transitive as trust and can not directly replace trust in trust propagation approaches. Therefore, simultaneously utilizing distrust and trust relations in matrix factorization requires careful consideration (trust is transitive, i.e., if user u trusts user v and v trusts w, there is a good chance that u will trust w, but distrust is certainly not transitive, i.e., if u distrusts v and v distrusts w, then w may be closer to u than v or maybe even farther away). This observation implies that distrust should not be used as a way to reverse deviations. This statement is consistent with the preliminary experimental results in [116] for memory-based CF methods which revealed that taking distrust as the negative of trust relations is not the correct way to incorporate distrust in recommender systems.

Motivated by this negative result, in [35] the ranking of latent features of neighbors of users based on the type of their relations has been shown as a suitable solution to incorporate both trust and distrust relations. The goal was to construct a ranking of latent features of all users based on their similarity such that trusted friends get a higher rank/score than the distrusted friends. In particular, from a ranking perspective, the similarity of latent features induce an ordering over all users where trusted users are pushed to top of the list.

However, the ranking method proposed in [35] suffers from two main issues. First, when the neighbors of a specific user u are considered in the ranking model based on the similarity of their latent features to latent features of u, the neutral users who have no relation to u are ignored. This might cause the neutral friends to appear any place in the ranking (e.g., above the trusted friends or below the distrusted friends). This affects the quality of recommendations significantly. More specifically, this contradicts our basic assumption on the effectiveness of social information, which relies on the fact that people tend to rely more on recommendations from people they trust than recommendations based on anonymous people. Therefore, in sharp contrast to [35] which excludes the users who have no relation to a specific user, we also model these users in learning the latent features by forcing them to appear in the ranked list after the trusted friends yet before the distrusted friends. This

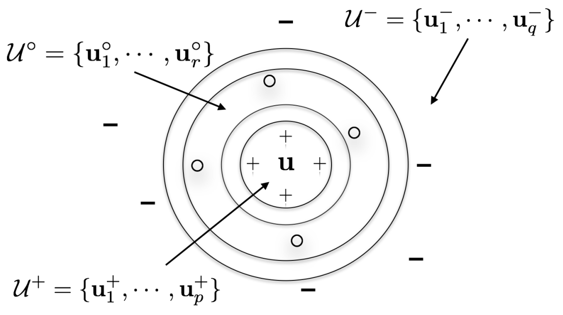
$\mathcal{U}^- = \{\mathbf{u}_1^-, \cdots, \mathbf{u}_q^-\}$
$\mathcal{U}^\circ = \{\mathbf{u}_1^\circ, \cdots, \mathbf{u}_r^\circ\}$
$\mathbf{u}$
$\mathcal{U}^+ = \{\mathbf{u}_1^+, \cdots, \mathbf{u}_p^+\}$

Figure 5.1: The ultimate goal is to extract latent features from the rating matrix $\mathbf{R}$ in a way that respects the social context of users in social network $\mathbf{S}$. In particular, from each user's perspective, say user u, the goal is to find latent features for u's neighbors such that when ranked based on their similarity to the latent features of the user u, i.e., $\mathbf{u} \in \mathbb{R}^k$, the trusted friends $\mathcal{N}^+(u)$ are pushed to the top portion of the list, the distrusted friends $\mathcal{N}^-(u)$ are pushed to the bottom of the list, and the neutral friends $\mathcal{N}^\circ(u)$ appear in the middle of ranked list.

matches our intuition as we expect neutral friends' contribution to the prediction not to be better than the contribution of trusted friends and worse than the contribution distrusted friends.

Moreover, the number of constraints imposed by social regularization in the ranking model proposed in [35] increases cubically with the number of users in the network. This growth limits the applicability of their method to small scale social networks. To resolve this issue, we introduce a novel raking model that only introduces a quadratic number of constraints into the factorization objective, thus making it more favorable for large scale social networks.

The main intuition behind the ranking model utilized in the `PushTrust` algorithm is shown in Figure 5.1. To keep our discussion simple and easy to follow, we consider the social regularization for a single user u with latent features vector $\mathbf{u} \in \mathbb{R}^k$. Let $\mathcal{N}^+(u) = \{v \in [n] \mid \mathbf{S}_{uv} = +1\}$, $\mathcal{N}^-(u) = \{v \in [n] \mid \mathbf{S}_{uv} = -1\}$ be the set of trusted and distrusted neighbors of u in the social network, respectively. Also let $\mathcal{N}^\circ(u) = \mathcal{V} \setminus \Big(\mathcal{N}^+(u) \cup \mathcal{N}^-(u)\Big)$ be the set of users for whom the user u is not socially connected, i.e, $\mathcal{N}^{\circ}(u) = \{v \in [n] \mid \mathbf{S}_{uv} = 0\}$. If we rank the latent features of users from the viewpoint of user u, we wish the trusted friends $\mathcal{N}^{+}(u)$ have the maximum similarity to u and are pushed to the top portion of the ranked list. Similarly, the distrusted friends $\mathcal{N}^{-}(u)$ are expected to have less similarity to $\mathbf{u}$ and appear at the bottom portion of the ranked list. The set of neutral users who are not connected to user u, i.e., $\mathcal{N}^{\circ}(u)$ are potentially either trusted or distorted. As a result, a conservative decision is to place the latent features of neutral users after trusted and before distrusted friends. To accomplish this goal, we introduce a social regularization term for each user $u \in \mathcal{U}$, where the the latent features of the user $\mathbf{u}$ is penalized based on the deviation of ranking of u's neighbors from the optimal ranking discussed above.

5.2.2 A convex formulation for social ranking

To simplify the analysis we rewrite the social regularized matrix factorization problem as:

$$\begin{aligned}\mathcal{F}(\mathbf{U}, \mathbf{V}) = & \frac{1}{2} \sum_{(i,j)\in\Omega_{\mathbf{R}}} \left(\mathbf{R}_{ij} - \mathbf{u}_i^{\top}\mathbf{v}_j\right)^2 + \frac{\lambda_{\mathbf{V}}}{2}\|\mathbf{V}\|_{\mathrm{F}} \\ & + \frac{\lambda_{\mathbf{U}}}{2}\|\mathbf{U}\|_{\mathrm{F}}^2 + \lambda_{\mathbf{S}} \sum_{i=1}^{n} \mathcal{P}(\mathbf{u}_i),\end{aligned}$$

where $\mathcal{P} : \mathbb{R}^k \mapsto \mathbb{R}_+$ is the social regularization of latent features of individual users.

The formalization of discussed intuition on social regularization of latent features of users is as follows. Let $\mathcal{U}^{+} = \{\mathbf{u}_1^{+}, \mathbf{u}_2^{+}, \cdots, \mathbf{u}_p^{+}\}$, $\mathcal{U}^{-} = \{\mathbf{u}_1^{-}, \mathbf{u}_2^{-}, \cdots, \mathbf{u}_q^{-}\}$, and $\mathcal{U}^{\circ} = \{\mathbf{u}_1^{\circ}, \mathbf{u}_2^{\circ}, \cdots, \mathbf{u}_r^{\circ}\}$ be the set of latent features of users in $\mathcal{N}^{+}(u)$, $\mathcal{N}^{-}(u)$, and $\mathcal{N}^{\circ}(u)$, respectively, where $p = |\mathcal{N}^{+}(u)|$, $q = |\mathcal{N}^{-}(u)|$, and $r = |\mathcal{N}^{\circ}(u)|$. The similarity between the latent features of u and their neighbor v in the network is measured by $\langle \mathbf{u}, \mathbf{u}_v \rangle$. A naive implementation of ranking idea is to consider the triplet of latent features which yields a large number of constraints to be satisfied. However, here we rely on the recent methods that assist with rank learning to rank, such as p-norm push, infinite push, and reverse-height push [99, 5], to order the latent features based on their similarity to the latent features of u. In particular, we define the following ranking function that tries to put as many possible trusted friends before the most similar distrusted and neutral friends. As it will be revealed later in this section, this formulation results in a significant decrease in the number of constraints that only linearly depends on the number of users in the social network. Formally, for user u we define the following objective as its social regularization:

$$\begin{aligned} \mathcal{P}(\mathbf{u}) = \quad & \frac{1}{p}\sum_{i=1}^{p} \mathbb{I}\left[\langle \mathbf{u}, \mathbf{u}_i^{+}\rangle \leq \max_{1\leq j\leq q}\langle \mathbf{u}, \mathbf{u}_j^{-}\rangle\right] \\ + \quad & \frac{1}{p}\sum_{i=1}^{p} \mathbb{I}\left[\langle \mathbf{u}, \mathbf{u}_i^{+}\rangle \leq \max_{1\leq j\leq r}\langle \mathbf{u}, \mathbf{u}_j^{\circ}\rangle\right] \\ + \quad & \frac{1}{r}\sum_{i=1}^{r} \mathbb{I}\left[\langle \mathbf{u}, \mathbf{u}_i^{\circ}\rangle \leq \max_{1\leq j\leq q}\langle \mathbf{u}, \mathbf{u}_j^{-}\rangle\right], \end{aligned} \tag{5.5}$$

where $\mathbb{I}[\cdot]$ is the indicator function which returns 1 when the input is true and 0 otherwise.

The social regularization in Eq. (5.5) consists of four terms. The first term is the standard regularization of latent features which is included to simplify the derivation of the objective. The second and third terms aim to put trusted friends at top of the ranked list based on their similarity to the latent features of u. Finally, the last term attempts to put neutral friends before the distrusted friends. Clearly, by minimizing the social regularization for each user; $\min_{\mathbf{u}\in\mathbb{R}^k} \mathcal{P}(\mathbf{u})$, it is guaranteed that the extracted latent features respect the social context of individual users discussed in section 4.1.

The above optimization problem is intractable due to the non-convex indicator function. To design practical learning algorithms, we replace the indicator function in Eq. (5.5) with its convex surrogate. To this end, define the convex loss function $\ell : \mathbb{R} \mapsto \mathbb{R}_{+}$ as $\ell(x) =$

Algorithm 5 `PushTrust Algorithm` [33]

```
1: Input: $\mathbf{R} \in \mathbb{R}^{n \times m}$: the partially observed rating matrix, $\Omega_{\mathbf{R}} \subseteq [n] \times [m]$: the set of
   observed ratings in $\mathbf{R}$, $\mathbf{S} \in \{-1, 0, +1\}^{n \times n}$: the social graph between users
1: $\lambda_{\mathbf{U}}, \lambda_{\mathbf{V}}, \lambda_{\mathbf{S}} \in \mathbb{R}_+$: the regularization parameters
2: for $t = 1, \ldots, T$ do
3:   for $i = 1, \ldots, n$ do
4:     Compute the gradient $\mathbf{g}^t_{\mathbf{u}_i}$ using Eq. (5.8)
5:     Set $\mathbf{u}_i^{t+1} = \mathbf{u}_i^t - \eta_t \mathbf{g}^t_{\mathbf{u}_i}$
6:   end for
7:   for $j = 1, \ldots, m$ do
8:     Compute the gradient $\mathbf{g}^t_{\mathbf{v}_j}$ using Eq. (5.9)
9:     Set $\mathbf{v}_j^{t+1} = \mathbf{v}_j^t - \eta_t \mathbf{g}^t_{\mathbf{v}_j}$
10:  end for
11: end for
12: Output: $\tilde{\mathbf{R}} = \mathbf{U}_T \mathbf{V}_T^\top$, where $\mathbf{U}_T = [\mathbf{u}_1^T, \ldots, \mathbf{u}_n^T]$ and $\mathbf{V}_T = [\mathbf{v}_1^T, \ldots, \mathbf{v}_m^T]$
```

$[1 - x]_+$. This is the widely used hinge loss in SVM classification (see e.g., [15]) [1]. This loss function reflects the amount by which the constraints are not satisfied. By replacing the non-convex indicator function with this convex surrogate leads to the following tractable convex optimization problem:

$$\begin{aligned} \mathcal{P}(\mathbf{u}) = \quad & \frac{1}{p}\sum_{i=1}^{p} \ell\left(\langle \mathbf{u}, \mathbf{u}_i^+ \rangle - \max_{1 \le j \le q} \langle \mathbf{u}, \mathbf{u}_j^- \rangle\right) \\ + \quad & \frac{1}{p}\sum_{i=1}^{p} \ell\left(\langle \mathbf{u}, \mathbf{u}_i^+ \rangle - \max_{1 \le j \le r} \langle \mathbf{u}, \mathbf{u}_j^\circ \rangle\right) \\ + \quad & \frac{1}{r}\sum_{i=1}^{r} \ell\left(\langle \mathbf{u}, \mathbf{u}_i^\circ \rangle - \max_{1 \le j \le q} \langle \mathbf{u}, \mathbf{u}_j^- \rangle\right). \end{aligned} \tag{5.6}$$

Equipped with the social regularization of individual users, we now turn to the final matrix factorization objective. To do so, Let $\mathbf{U}_i^- = [\mathbf{u}_{i,1}^-, \cdots, \mathbf{u}_{i,q}^+]^\top$, and $\mathbf{U}_i^\circ = [\mathbf{u}_{i,1}^\circ, \cdots, \mathbf{u}_{i,r}^\circ]^\top$ denote the matrix of latent features of distrusted and neutral friends of ith user in the social network, respectively. Here $\mathbf{u}_{i,j}^-$ and $\mathbf{u}_{i,j}^\circ$ is the latent features of jth distrusted and neutral

[1]We note that other convex loss functions such as exponential loss $\ell(x) = \exp(-x)$, and logistic loss $\ell(x) = \log(1 + \exp(-x))$ also can be used as the surrogates of indicator function, but for the simplicity of derivation we only consider the hinge loss here.

friends of the ith user, respectively. Then, we have:

$$\begin{aligned}\mathcal{F}(\mathbf{U},\mathbf{V}) = &\frac{1}{2}\sum_{(i,j)\in\Omega_{\mathbf{R}}}\left(\mathbf{R}_{ij}-\mathbf{u}_i^\top\mathbf{v}_j\right)^2+\frac{\lambda_{\mathbf{V}}}{2}\|\mathbf{V}\|_{\mathrm{F}}+\frac{\lambda_{\mathbf{U}}}{2}\|\mathbf{U}\|_{\mathrm{F}}^2\\
&+\lambda_{\mathbf{S}}\sum_{i=1}^{n}\left(\frac{1}{p}\sum_{j\in\mathcal{N}^+(i)}\ell\left(\langle\mathbf{u}_i,\mathbf{u}_{i,j}^+\rangle-\|\mathbf{U}_i^-\mathbf{u}_i\|_\infty\right)\right)\\
&+\lambda_{\mathbf{S}}\sum_{i=1}^{n}\left(\frac{1}{p}\sum_{j\in\mathcal{N}^+(i)}\ell\left(\langle\mathbf{u}_i,\mathbf{u}_{i,j}^+\rangle-\|\mathbf{U}_i^\circ\mathbf{u}_i\|_\infty\right)\right)\\
&+\lambda_{\mathbf{S}}\sum_{i=1}^{n}\left(\frac{1}{r}\sum_{j\in\mathcal{N}^\circ(i)}\ell\left(\langle\mathbf{u}_i,\mathbf{u}_{i,j}^\circ\rangle-\|\mathbf{U}_i^-\mathbf{u}_i\|_\infty\right)\right).\end{aligned}\tag{5.7}$$

5.3 Optimization procedure

We now turn to solving the optimization problem in Eq. (5.7). We note that by excluding the ranking constraints from the formulation in Eq. (5.7), we get the standard matrix factorization objective as formulated in Eq. (5.1) which is non-convex jointly in both $\mathbf{U}$ and $\mathbf{V}$. However, despite its non-convexity, it is widely used in practical collaborative filtering applications as the performance is competitive (or better) when compared to trace-norm minimization, while scalability is much better. For example, as indicated in [58], to address the Netflix problem, Eq. (5.1) has been applied with a fair amount of success to factorize datasets with 100 million ratings. To find a local solution one can stick to the standard gradient descent method to find a solution in an iterative manner.

The detailed steps of the proposed `PushTrust` algorithm in its most basic form are shown in Algorithm 5. The optimization is done by alternating minimization, i.e., updating $\mathbf{U}$ while keeping $\mathbf{V}$ fixed, and then updating $\mathbf{V}$ while keeping $\mathbf{U}$ fixed. The objective function is bi-convex, i.e., convex in one argument while keeping the other fixed. The updates of the latent factors for each user and each item can be done using gradient descent. In particular, a local minimum of the objective function can be found by performing gradient descent in feature vectors $\mathbf{u}_i$ and $\mathbf{v}_j$. To do so, we first compute the (sub)gradient of objective function $\mathcal{F}(\mathbf{U}, \mathbf{V})$ with respect to $\mathbf{u}_i$ and $\mathbf{v}_j$ as:

$$\begin{aligned}\mathbf{g}_{\mathbf{u}_i} = \frac{\partial \mathcal{F}}{\partial \mathbf{u}_i} &= \sum_{j=1}^{m} \mathbf{I}_{ij}(\mathbf{R}_{ij} - \mathbf{u}_i^\top \mathbf{v}_j)\mathbf{v}_j + \lambda_{\mathbf{U}} \mathbf{u}_i \\ &+ \frac{\lambda_{\mathbf{S}}}{p} \sum_{j=1}^{p} \mathbb{I}\left[\langle \mathbf{u}_i, \mathbf{u}_{i,j}^{+}\rangle - \|\mathbf{U}_i^{-}\mathbf{u}_i\|_\infty \leq 1\right] \left(\partial \|\mathbf{U}_i^{-}\mathbf{u}_i\|_\infty - \mathbf{u}_{i,j}^{+}\right) \\ &+ \frac{\lambda_{\mathbf{S}}}{p} \sum_{j=1}^{p} \mathbb{I}\left[\langle \mathbf{u}_i, \mathbf{u}_{i,j}^{+}\rangle - \|\mathbf{U}_i^{\circ}\mathbf{u}_i\|_\infty \leq 1\right] \left(\partial \|\mathbf{U}_i^{\circ}\mathbf{u}_i\|_\infty - \mathbf{u}_{i,j}^{+}\right) \\ &+ \frac{\lambda_{\mathbf{S}}}{r} \sum_{j=1}^{r} \mathbb{I}\left[\langle \mathbf{u}_i, \mathbf{u}_{i,j}^{\circ}\rangle - \|\mathbf{U}_i^{-}\mathbf{u}_i\|_\infty \leq 1\right] \left(\partial \|\mathbf{U}_i^{-}\mathbf{u}_i\|_\infty - \mathbf{u}_{i,j}^{\circ}\right)\end{aligned} \tag{5.8}$$

and

$$\mathbf{g}_{\mathbf{v}_j} = \frac{\partial \mathcal{F}}{\partial \mathbf{v}_j} = \sum_{i=1}^{n} \mathbf{I}_{ij}(\mathbf{R}_{ij} - \mathbf{u}_i^\top \mathbf{v}_j)\mathbf{u}_i + \lambda_{\mathbf{V}} \mathbf{v}_j. \tag{5.9}$$

where $\mathbf{I}_{ij}$ is the indicator function of observed entries in the rating matrix $\mathbf{R}$, i.e., $\mathbf{I}_{ij} = 1$ if $(i, j) \in \Omega_{\mathbf{R}}$ and 0 otherwise, and $\partial \|\mathbf{U}_i^{-}\mathbf{u}_i\|_\infty$ is the subdifferential of the function $\|\mathbf{U}_i^{-}\mathbf{u}_i\|_\infty$ at point $\mathbf{u}_i$. Since the subdifferential of the maximum of functions is the convex hull of the union of subdifferentials of the active functions at point $\mathbf{u}$ [83], we have:

$$\begin{aligned}\partial \|\mathbf{U}_i^{-}\mathbf{u}_i\|_\infty &= \partial \max_{1 \leq j \leq q} \langle \mathbf{u}_i, \mathbf{u}_j^{-}\rangle \\ &= \operatorname{conv}\left\{\mathbf{u}_j^{-} | \langle \mathbf{u}_i, \mathbf{u}_j^{-}\rangle = \|\mathbf{U}_i^{-}\mathbf{u}_i\|_\infty, j \in [q]\right\}.\end{aligned}$$

In a similar way, we can compute the $\partial \|\mathbf{U}_i^{\circ}\mathbf{u}\|_\infty$. Having computed the gradients, we iteratively update the latent features at iteration $t+1$ by:

$$\mathbf{u}_i^{t+1} \leftarrow \mathbf{u}_i^t - \eta_t \mathbf{g}_{\mathbf{u}_i}^t, \quad \mathbf{v}_i^{t+1} \leftarrow \mathbf{v}_i^t - \eta_t \mathbf{g}_{\mathbf{v}_i}^t.$$

Here, η_t is the learning rate, and $\mathbf{g}_{\mathbf{u}_i}^t$ and $\mathbf{g}_{\mathbf{v}_i}^t$ are the subgradients of the objective with respect to $\mathbf{u}_i$ and $\mathbf{v}_i$ at iteration t, respectively. We observe that, due to the non-smooth max operator in the social regularization term, one needs to use to sub-gradient descent methods to optimize the above objective.

Comparing the objective in Eq. (5.7) with objective Eq. (5.4) that is introduced in [35], a few comments are in order. First, here the neutral friends of users are also incorporated in ranking the latent features. This is more conservative than the formulation in Eq. (5.4) which ignores these types of relations. In particular, in Eq. (5.4) the neutral friends may appear any place in the ranking even before the trusted friends. However, in Eq. (5.7) neutral friends are pushed to the middle of ranked list which is more reasonable under real world assumptions. Second, as it will become more clear in the dual formation of objective Eq. (5.7), the number of constraints in Eq. (5.7) increases quadratically $O(n^2)$ with the number of users because of the maximum norm used in formulation, while in Eq. (5.4) the number of constrains is cubic $O(n^3)$ in the number of users due to the pairwise ranking used to rank the neighbors of each user. This provides a significant computational advantage when we tackle social recommendation problems with large number of users.

Remark 5.3.1. *We note that for large-scale networks, the computation of gradient at each step of Algorithm 5 might be computationally expensive and one can use stochastic methods to replace the expensive full gradient computation with cheap stochastic gradient computation. This can be done by sampling a user and only updating his/her latent features.*

Chapter 6

Cold-start Recommendation: Rating Prediction

6.1 Introduction

Due to the popularity and exponential growth of e-commerce websites (e.g. Amazon, eBay) and online streaming websites (e.g. Netflix, Hulu), a compelling demand has been created for efficient recommender systems to guide users toward items of their interests (e.g. products, books, movies). Recommender systems seek to predict the rating that a user would give to an item and further try to suggest items that are most appealing to the users. Recently, recommender systems have received a considerable amount of attention and have been the main focus of many research studies [2].

Content-based filtering (CB) and collaborative filtering (CF) are well-known examples of recommendation approaches. As demonstrated by its performance in the KDD Cup [29] and Netflix competition [13], the most successful recommendation technique used is collaborative filtering. This technique exploits the users' opinions (e.g., movie ratings) and/or purchasing (e.g., watching, reading) history in order to extract a set of interesting items for each user. In factorization based CF methods, both users and items are mapped into a latent feature space based on observed ratings that are later used to make predictions.

Despite significant improvements in recommendation systems, and in particular factorization based methods, these systems suffer from a few inherent limitations and weaknesses such as *data sparsity* and *cold-start* problems. Specifically, in many real world applications, the rating data are very sparse (i.e., most users do not rate most items typically resulting in a very sparse rating matrix) or for a subset of users or items the rating data is entirely missing (knows as cold-start user and cold-start item problem, respectively [103]). To address the difficulties associated the latter issues, there has been an active line of research during the past decade and a variety of techniques have been proposed [87, 100, 109, 121, 64, 114, 122, 77, 92, 120, 62].

The studies in the literature have approached the cold-start problem from many different angles, but they commonly exploit the auxiliary information about the users and items in addition to the rating data that are usually available (see e.g. [106] for a more recent survey). By leveraging multiple sources of information one can potentially bridge the gap between existing items (or users) and new (cold start) items (or users) to mitigate the cold-start problem.

The main motivation behind these techniques stems from the observation that other sources of data can be used to reveal more information about the underlying patterns between users and items and thus complement the rating data. For instance, knowing the underlying social connections (friends, family, etc.) between users can give us a better understanding of the sources of influence on a user's decision. Subsequently, the availability of auxiliary information such as users' profile information [2], social context (trust and distrust relations) of users [35], information embedded in the review text [64], and features of items [37] provide tangible benefits to the recommender. Hence, an intriguing research question, which is the main focus of this thesis is:

How can side information about users and items be effectively incorporated in factorization methods to overcome the cold-start problem?

Although there are many different algorithms in the literature to augment matrix factorization with side information, such as shared subspace learning [102] and kernelized matrix factorization [122], the dominant paradigm in existing methods is to perform (a) the *completion* of rating matrix and (b) the *transduction* of knowledge from existing ratings to cold-start items/users simultaneously. While these methods are able to generate good results in practice, they have notable drawbacks: 1) these methods propagate the completion and transduction errors repetitively and in an uncontrolled way, 2) many of state-of-art algorithms are usually application based, e.g. see [114, 63], and do not offer a general framework for alleviating cold-start problems.

The overarching goal of our work is to answer the above question by proposing an efficient matrix factorization approach using similarity information. In fact, not only we propose a general framework for dealing with cold-start problems, but also we study a somewhat more general problem where both cold-start users and cold-start items are present simultaneously and address these two challenges simultaneously. In particular, by considering the drawbacks of the existing methods, we propose a two-stage algorithm that *decouples* the completion and transduction stages. First, we exclude the cold-start items and users and complete the rating matrix. Next, we transduct the knowledge to cold-start users/items using the recovered sub-matrix in addition to the available side information about the users and items. Hence, there is no error propagation of completion and transduction. Interestingly, beyond just dealing with cold-start problem, the proposed algorithm also provides an effective way to exploit the side information about users (or items) to mitigate the data sparsity and compensate for the lack of rating data.

Unlike most existing methods, we also provide a theoretical performance guarantee on the estimation error of our proposed algorithm. We further complement our theoretical results with comprehensive experiments on a few benchmark datasets to demonstrate the merits and advantages of proposed framework in dealing with cold-start problems. Our results demonstrate the superiority of our proposed framework over several of state-of-art cold-start recommender algorithms.

6.2 Experiments

In this section, we conduct exhaustive experiments to demonstrate the merits and advantages of the proposed algorithm. We conduct our experiments on three well-known datasets: MovieLens (1M and 100K) [1], Epinions [2] and NIPS [3]. Using these, we explore several fundamental questions.

- **Prediction accuracy:** How does the proposed algorithm perform in comparison to the state-of-the-art algorithms with incorporating side information of users/items. Further, to what degree can the available side information help in making more accurate recommendations existing items for existing users?
- **Dealing with cold-start users:** How does exploiting similarity relationships between users affect the performance of recommending existing items to cold-start users?
- **Dealing with cold-start items:** How does exploiting similarity information between items affect the performance of recommending cold-start items to existing users?

[1] `http://www.grouplens.org/node/73`
[2] `http://www.trustlet.org/wiki/Epinions\_dataset`
[3] `http://www.cs.nyu.edu/~roweis/data.html`

- **Dealing with cold-start users and items simultaneously:** How does exploiting similarity information between users and items affect the performance of recommending cold-start items to cold-start users?

6.2.1 Datasets

Since our proposed general framework is applicable solution for both cold-start and network completion problems, we selected real well-known datasets for each each problem. We used MovieLens and Epinions as two samples of cold-start problem and used NIPS dataset as a network completion problem.

MovieLens dataset. We used two of the well known MovieLens datasets, 100K and 1M, which have side information for both users and movies (items) that makes them ideal for our experiment. The statistics of MovieLens datasets are given in Table 6.1.

Epinions dataset. This dataset is obtained from a user-oriented product review website that has a trust network of users. Users can specify whether they trust other users or not. We use this trust network as the side information for the users. The statistics of Epinions are also shown in Table 6.1.

NIPS dataset. We have also applied our algorithm to paper-author and paper-word matrices extracted from the co-author network at the NIPS conference [98]. The contents of the papers are pre-processed such that all words are converted to lower cases and stop-words are removed. We compute the cosine similarity of the vector representation weighted with TD-IDF of papers with the ones of all other papers. The statistics of this dataset are shown in Table 6.1.

6.2.2 Metrics

We adopt the widely used the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) metrics for prediction accuracy [50]. Let $\mathcal{T}$ denote the set of ratings that we want to predict, i.e., $\mathcal{T} = \{(i, j) \in [n] \times [m], \mathbf{R}_{ij}$ needs to be predicted$\}$ and let $\widehat{\mathbf{R}}$ denote the prediction matrix obtained by a recommendation algorithm. Then,

$$\text{MAE} = \frac{\sum_{(i,j)\in\mathcal{T}} |\mathbf{R}_{ij} - \widehat{\mathbf{R}}_{ij}|}{|\mathcal{T}|},$$

The RMSE metric is defined as:

$$\text{RMSE} = \sqrt{\frac{\sum_{(i,j)\in\mathcal{T}} \left(\mathbf{R}_{ij} - \widehat{\mathbf{R}}_{ij}\right)^2}{|\mathcal{T}|}}.$$

Given an item i, let r_i be the relevance score of the item ranked at position i, where $r_i = 1$ if the item is relevant to the i and $r_i = 0$ otherwise. The NDCG measure is a normalization of the Discounted Cumulative Gain (DCG) measure. DCG is a weighted sum of the degree of relevancy of the ranked users. The value of NDCG is between $[0, 1]$ and at position k is defined as:

$$\text{NDCG@}k = Z_k \sum_{i=1}^{k} \frac{2^{r_i} - 1}{\log(i+1)}$$

In all our experiments, we set the value of k to the number of rated items of users for calculating the NDCG.

Table 6.1: Statistics of real datasets used in our experiments.

Statistics	MovieLens 100K	MovieLens 1M	Epinioins	NIPS
Number of users	943	6,040	8,577	2,073
Number of items	1682	3,706	3,769	1,740
Number of ratings	100,000	1,000,209	203,275	3,990
Range of ratings	1 - 5	1 - 5	1 - 5	0-1
Maximum number of ratings by users	737	2314	416	43
Maximum number of ratings for items	583	3428	1236	10
Average number of ratings by users	106.04	165.59	23.70	1.95
Average number of ratings for items	59.45	257.58	53.93	2.29

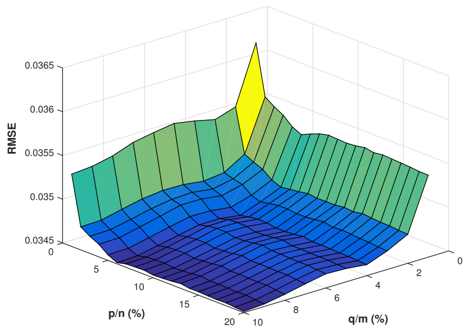

Figure 6.1: RMSE of synthetic dataset for different values of p and q

6.2.3 Experiments on synthetic dataset

In this section we present the results of our proposed algorithm on one synthetic dataset. We generated a synthetic dataset to evaluate our approach before moving forward to real datasets. First we generate two matrices $\mathbf{U} \in [0,1]^{4{,}000\times r}$ and $\mathbf{V} \in [0,1]^{2{,}000\times r}$. Then by using $\mathbf{U}$ and $\mathbf{V}$ we generated a rating matrix $\mathbf{R}^{4{,}000\times 2{,}000} = \mathbf{U}\mathbf{V}^\top$ that includes 4,000 users and 2,000 items. Then we generated a similarity matrix $\mathbf{A}^{4{,}000\times 4{,}000} = \mathbf{U}\mathbf{U}^\top$ for users and a similarity matrix $\mathbf{B}^{2{,}000\times 2{,}000} = \mathbf{V}\mathbf{V}^\top$ for items. Then we added random noise to the all elements of $\mathbf{A}$ and $\mathbf{B}$ where the noise follows a Gaussian distribution $\mathcal{N}(0, 0.5)$. We consider $\mathbf{A}$ and $\mathbf{B}$ as two similarity matrices between users and items, respectably.

In this study we tried different values for p within the range of 100 to 2,000 with step-size

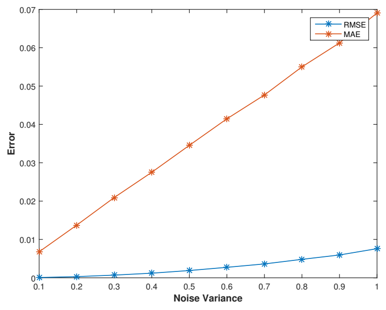

Figure 6.2: RMSE and MAE of synthetic dataset for different values of noise variance

of 100 and different values for q within the range of 100 to 1,000 with same step-size. To generate cold-start users and items in our dataset, we divided the dataset into 4 partitions. We divided users into two groups of p existing users and $n - p$ cold-start users. We also selected q items as existing items and $m - q$ as cold-start items. Hence Partitions 1 and 2 have p users and partitions 3 and 4 have $n - p$ cold-start users; partitions 1 and 3 have q items and partitions 2 and 4 have $m - q$ cold-start items.

The RMSE of test data for different combinations of p and q values is shown in Figure 6.1. It shows that by increasing p and q, the RMSE decreases. Hence, the fewer unobserved elements we have, the lower the error is. Considering the fact that our theoretical upper bound of error decreases by increasing $\frac{p}{m}$ or $\frac{q}{n}$, this observation is completely consistent with our theoretical upper bound.

For our synthetic dataset, we added noise to similarity matrices that follows a zero mean Gaussian distribution with variance of 0.5. To investigate the effects of noise variance, we vary the variance from 0.1 to 1 with step size of 0.1 and calculate the accuracy of our algorithm. As Figure 6.2 shows, by increasing the noise variance, both RMSE and MAE of the results on test data increase; Hence, we can see the stability of `RECT` with respect to noise.

6.2.4 Experiments setup

To better evaluate the effect of utilizing the features of users and items, we studied the following recommendation scenarios.

- **I [existing users/existing items]**: There are many different algorithms for predicting the rating of existing users on those existing items that they have not rated yet. We randomly removed 20% of the available ratings from the rating matrix and used the rest of the ratings for our training to predict the removed values.
- **II [existing users/cold-start items]**: In this case we have a number of cold-start items with no cold-start users. To simulate this scenario, we considered 80% of the items as training items and the remaining 20% as cold-start items. Then we tried to predict the ratings of existing users on the cold-start items.
- **III [cold-start users/existing items]**: Often, in a recommender system a vast majority of the users are new users. To simulate the cold-start user scenario, we divided the users into a disjoint training (80%) set and a test (%20) set. Then we predicted the rating of of cold-start users on the existing items.
- **IV [cold-start users/cold-start items]**: In this final scenario we divided both users and items into two subsets of existing and cold-start users and items, respectively. We aimed to recommend the cold-start items to the cold-start users, which is the most challenging form of cold-start problem.

We also analyzed the running time needed for training of our algorithm and cold-start scenario baseline algorithms to better evaluate the cold-start scenarios. It is also worth mentioning that we tuned the parameter of our algorithm by running it on training sets and reported these values for each result.

6.2.5 The baseline algorithms

To evaluate the performance of our proposed `RECT` algorithm, we considered a variety baseline approaches. The baseline algorithms are chosen from four types of categories: (i) state-of-the-art algorithms for rating predictions, (ii) those that can only deal with cold-start items, (iii) ones that can deal with cold-start users and (iv) algorithms that are capable of dealing with both cold-start items and users. In particular, we considered the following basic algorithms [4]:

- **RS** (Random Strategy) [66]: A simple baseline that selects at random a subset of users or items. The recommendation for cold-start users and items is a challenging case, where RS is one of the baseline methods.
- **U-KNN** (User KNN) [57]: Predicts the rates using the similarity with the K nearest neighbor where users have weights.
- **I-KNN** (Item KNN) [57]: Is a weighted item-based KNN approach for rate prediction.
- **GA** (Global Average): Uses the average of ratings over all ratings.
- **I-A** (Item Average): Uses the average rating of an item for its prediction.
- **U-A** (User Average): Uses the average rating of a user for its prediction.
- **SO** (Slope One) [60]: Pre-computes the average difference between two items that are rated by users. SO is a frequency weighted slope one rating prediction.
- **BPSO** (BiPolar Slope One) [60]: Is a Bi-polar frequency weighted slope one rating prediction.
- **SMF** (Social Matrix Factorization) [51]: Is a matrix factorization algorithm that incorporates the social network for prediction.
- **CC** (CoClustering) [39]: Is a weighted co-clustering algorithm that involves simulta-

[4] Some baseline algorithms used for our experiments are implemented in the MyMedia recommender framework [38]

neous clustering of users and items.

- **LFLLM** (Latent Feature Log Linear Model) [78]: Is a scalable log-linear model that exploits the side information for dyadic prediction.
- **U-I-B** (User Item Baseline) [57]: Is a rating prediction method that uses the average rating value along with a regularized user and item bias.
- **FWMF** (FactorWise Matrix Factorization): Is a matrix factorization based model with a factor-wise learning
- **BMF** (Biased Matrix Factorization) [96]: Is a matrix factorization that learns by stochastic gradient descent with explicit bias for users and items.
- **SVDPP** (SVD++) [56]: Is a matrix factorization that takes into account what users have rated and directly profiles users and items.
- **SSVDPP** (Sigmoid SVD++) [56]: Is a version of SVD++ that variant that uses a sigmoid function.
- **SU-AFM** (Sigmoid User Asymmetric Factor Model) [89]: Is an asymmetric factor model that represents the items in terms of those users that rated them.
- **SI-AFM** (Sigmoid Item Asymmetric Factor Model) [89]: Is an asymmetric factor model that represents users in terms of the items they rated.
- **SCAFM** (Sigmoid Combined Asymmetric Factor Model) [89]: Is an asymmetric factor model that represents items in terms of the users that rated them, and users in terms of the items they rated.
- **CBF** (Content Based Filtering) [102]: This algorithm builds a profile for each user based on the properties of the past user's preferred items.
- **LCE** (Local Collective Embeddings) [102]: Is a matrix factorization method that exploits properties of items and past user preferences while enforcing the manifold struc-

Table 6.2: Synthetic DataSet

Setting	Algorithm	Training (10%)		Training (20%)		Training (30%)		Training (40%)		Training (50%)	
		MAE	RSME	MAE	RSME	MAE	RSME	MAE	RSME	MAE	RSME
Scenario I	RECT	0.0353	0.0020	0.0345	0.0018	0.0345	0.0018	0.0341	0.0018	0.0340	0.0018
Scenario II	RECT	0.0353	0.0020	0.0345	0.0018	0.0345	0.0018	0.0341	0.0018	0.0340	0.0018
Scenario III	RECT	0.0345	0.001	0.0342	0.0019	0.0342	0.0019	0.0343	0.0019	0.0342	0.0019
Scenario IV	RECT	0.0349	0.0019	0.034	0.0018	0.0346	0.0019	0.0343	0.0019	0.0344	0.0018

ture exhibited by the collective embeddings. LCE can be fed with properties of either items or users to deal with cold-start items or users, respectively. We also conduct experiments with a version of LCE without laplacian regularization, which will be referred as LCE-NL [102].

- **KMF** (Kernelized Matrix Factorization) [122]: Is a matrix completion based algorithm, which incorporates external side information of the users or items into the matrix factorization process.
- **RECT**: Our proposed algorithm.

6.2.6 Existing Users/Existing Items

Scenario I is the standard case for recommender systems that all users and items have a history of rating and being rated, respectively. Each user usually rates only a number of items and there are no ratings for other items. The goal here is to predict the ratings for those unrated items. There are many different techniques for predicting the rates but we can divide them into two major categories: *neighbor based approaches* and *laten factor models.* We selected our baseline algorithms from each of these categories for our comparison.

To show the results, we applied RECT on MovieLens 100K, MovieLens 1M and Epinions datasets. GroupLens Research [5] made 5 sets available, which are 80%/20% splits of the

[5] grouplens.org/

MovieLens 100K into training and test data. We also split the MovieLens 1M and Epinions into 80%/20% randomly 5 times to conduct a 5-fold cross-validation and recorded the average values of our metrics.

Table 6.3 shows the average RMSE and MAE of 5-fold cross-validation for all baseline algorithms and `RECT` and the smallest RMSE and MAE values for each column is indicated by boldface. The results suggested that among neighbor-based approaches, I-KNN is the best-performing algorithm for MovieLens 1M and 100K and U-KNN is the second best-performing one. That is because of the fact that the similarity between movies (genre, director, etc) and similarity between taste of users, play an important role in prediction accuracy. I-KNN and U-I-B outperformed other neighbor based methods on Epinions in respect to RMSE measures while BPSO and I-KNN outperformed others in respect to MAE measures. Similarity of items, and having the same pattern of ratings for similar items, on Epinions helped I-KNN's results perform better than other neighbor based methods.

Table 6.3 also shows the comparison between latent factor methods in which `RECT` (SVDPP) algorithm achieved the first (second) best performance of RMSE and MAE for MovieLens 100K and RMSE for MovieLens 1M and achieved the second (first) best performance of MAE for MovieLens 1M. On Epinions, too, `RECT` outperformed other latent factor methods.

Table 6.3 clearly shows that `RECT` achieved the best performance for all datasets among all methods of both categories, latent factor and neighbor based methods (except for MAE on MovieLens 1M), confirming the performance advantage of `RECT` over all baseline algorithms. Hence, our proposed decoupled method by incorporating the auxiliary information, reveals the need for preventing error propagation along with using side information to obtain more accurate predictions.

Table 6.3: Results of scenario I on MovieLens 100K and 1M and Epinions

	Algorithms	**Hyperparameters**	**MovieLens 100K**		**MovieLens 1M**		**Epinions**	
			RMSE	MAE	RMSE	MAE	RMSE	MAE
Neighbor-Based Methods	GA	—	1.1190	0.9399	1.116	0.9327	1.1692	0.8878
	I-A	—	1.0220	0.8159	0.9759	0.7790	1.0695	0.8140
	U-A	—	1.0390	0.8350	1.034	0.8272	1.1276	0.8769
	U-KNN	k=80	0.9355	0.7398	0.8952	0.7030	2.3999	2.2229
	I-KNN	k=80, sh=10,λ_u=25, λ_v=10	0.9241	0.7270	0.8711	0.6830	1.0279	0.6993
	U-I-B	λ_u=5, λ_v=2	0.9419	0.7450	0.9081	0.7190	1.0290	0.8010
	SO	—	0.9397	0.7403	0.9020	0.7120	1.0865	0.7067
	BPSO	—	0.9744	0.7482	0.9390	0.7199	1.0449	0.6813
	CC	C_i=3, C_u=3, T=30	0.9559	0.7526	0.9118	0.7134	1.0573	0.7890
Latent Factor Models	SMF	p=10, λ_u=0.015, λ_v=0.015, λ_b=0.01 λ_s=1, η=0.01, η_b=1, T=30	1.0134	0.7884	1.2284	0.9315	1.1224	0.8436
	FWMF	p=5, T=5, sh=150	0.9212	0.7252	0.8601	0.6730	1.5090	1.0597
	BMF	p=160, λ_b=0.003, η=0.07, T=100 λ_u=0.08,λ_v=0.1	0.9104	0.7194	0.8540	0.6760	1.0240	0.7918
	MF	p=10, T=75 λ_g=0.05, η=0.005	0.9133	0.7245	0.8570	0.6751	1.0908	0.8372
	LFLLM	p=10, λ_b=0.01, λ_u=0.015, λ_v=0.015 η=0.01, T=30, η_b=1	0.9550	0.7617	0.9012	0.7082	1.2891	1.0386
	SI-AFM	η_b=0.7, λ_g=0.015, η=0.001, p=10 λ_b=0.33, T=1	0.9568	0.7628	1.035	0.8488	1.1534	0.8816
	SU-AFM	η_b=0.7,λ_g=0.015 λ_b=0.33, T=1, η=0.001, p=10	0.9569	0.7634	0.9062	0.7189	1.069	0.8398
	SCAFM	—	0.9499	0.7559	0.9121	0.7239	1.0600	0.8312
	SVDPP	η_b=0.07,λ_g=1 λ_b=0.005, p=50, η=0.01, T=50	0.9065	0.7135	0.8510	**0.6680**	1.0550	0.8220
	SSVDPP	η_b=0.7, T=30 p=10,λ_g=0.015, η=0.001, λ_b=0.33	1.185	0.9147	0.9402	0.7352	1.3328	0.9022
	RS	—	1.6960	1.3860	1.7070	1.3940	1.9096	1.5789
	`RECT`	r=10	**0.7002**	**0.6628**	**0.7157**	0.6721	**0.7157**	**0.6796**

6.2.7 Existing Users/Cold-start Items

To simulate cold-start item problems (scenario II), we divided the items into two disjoint training and test subsets. We used 80% of the items as existing items for training and the remaining 20% as cold-start items for testing.

As we mentioned earlier, our general framework can be employed to deal with link prediction challenges as well. In order to kill two birds with one stone, we selected the NIPS dataset to not only simulate the cold-start item scenario, but also to show the results of `RECT` for link prediction challenge. NIPS has rich side information for the items (papers) and shows the relationship (0 or 1) between papers and authors. Since in NIPS the values are either 0 or 1, predicting the authors of new papers can be perceived as a link prediction problem.

We compared our algorithm with four competitive recommendation methods: CBF, KMF, LCE and LCE-NL on NIPS dataset. Table 6.4 shows the average RMSE and MAE of 5-fold cross-validation for these algorithms for cold-start item scenario. The parameter [6] setting of each algorithm is also given in the Table 6.4 for reproducing the experiments.

The results indicate that `RECT` is the best performing algorithm among all baseline algorithms in cold-start item scenario in respect to RMSE, MAE and NDCG. Having the highest NDCG value among all competitive algorithms shows that `RECT` can present the top-ranked items to users better than other algorithms. `RECT` has also the lowest RMSE and MAE value from which we can conclude that predicting the ratings for cold-start items is more accurate. Hence, `RECT` can better suggest the top-ranked cold-start items to users with higher accuracy. Table 6.4 also shows the running time of the algorithms. Here CBF takes the least time to generate its results due to the fact that it only is required to build the user profiles. It is

[6] The details of parameters are explained in [38]

Table 6.4: Results of cold-start scenarios on real datasets.

DS	**Algorithm**	**Hyperparameters**	**Measures**			
			NDCG@k	RMSE	MAE	Time(s)
NIPS	CBF	—	0.3861	0.7943	0.8881	**0.17597**
	LCE	k=500, λ_g=0.5, ε = 0.001, T_m=500, β=0.05	0.4240	0.7692	0.8675	709.869
	LCE-NL	k=500, λ_g=0.5, ε=0.001, T_m=500, β=0	0.4186	0.7532	0.8562	1823.48
	KMF	σ_r=0.4, D=10, η=0.003, γ=0.1	0.1415	0.8804	0.9196	19.5413
	RECT	r=1000	**0.4626**	**0.5111**	**0.6805**	23.8410
Epinions	CBF	—	0.2201	0.6644	0.7741	**4.4800**
	LCE	k=500, λ_g=0.5, ε=0.001, T_m=500, β=0.05	0.2327	0.6713	0.7786	1067.11
	LCE-NL	k=500, λ_g=0.5, ε=0.001, T_m=500, β=0	0.2319	0.6712	0.7785	11969.9
	KMF	σ_r=0.4, D=10, η=0.003, γ=0.1	0.2084	0.8522	0.8882	1196.32
	RECT	r=1063	**0.2343**	**0.6618**	**0.7716**	144.660
MovieLens 100K	RS	—	0.1022	1.1981	0.9782	**0.0131**
	KMF	σ_r=0.4, D=10, η=0.003, γ=0.1	0.2423	0.9823	0.8730	11.540
	RECT	r=100	0.2641	**0.8672**	**0.7230**	4.8729
MovieLens 1M	RS	—	0.0652	1.3820	0.9326	**0.0682**
	KMF	σ_r=0.4, D=10, η=0.003, γ=0.1	0.1834	0.9730	0.8442	63.732
	RECT	r=100	**0.2783**	**0.8524**	**0.7162**	10.578

worth mentioning that KPMF and RECT are comparably fast, but LCE and LCE-NL take much more time due to their convergence conditions.

6.2.8 Cold-start Users/Existing Items

To simulate cold-start user problems (scenario III), we divided the users into two disjoint subsets of training and test. We used 80% of the users for training and the remaining 20% for testing of the cold-start user scenario. To show the relative results of RECT, we compare it with competitive algorithms: CBF, LCE, LCE-NL and KPMF.

Since Epinions has the trust network among the users, which is a useful side information, we chose it to simulate this scenario. Table 6.4 shows the averaged (5-fold cross validation) performance results of the mentioned algorithms and it clearly shows that RECT achieved the best performance of NDCG, RMSE and MAE on Epinions dataset.

Considering the NDCG values, RECT outperforms all other methods while NDCG of LCE and LCE-NL are still close. RECT, LCE and LCE-NL also outperform other methods in respect to RMSE and MAE. In this case, while RECT and LCE and LCE-NL behave similarly, the major difference between RECT and LCE(-NL) is the running time where CBF outperforms RECT and RECT outperforms the rest of methods. Though there are no major differences between the values of these metrics for RECT and others, the consistency in outperforming other competitive methods on both cold-start user and item scenarios confirms the stability and performance advantage of RECT over current state-of-the-art algorithms.

Again, as CBF is only required to build the user profile, it has the shortest running time. Our algorithm has the second fastest running time, but it should be noted that the other algorithms take substantially longer the execute than either RECT or CBF.

6.2.9 Cold-start Users/Cold-start Items

To simulate handling both cold-start users and items (scenario IV), we randomly selected 20% of the users and 20% of the items as cold-start users and items, respectively. In this scenario we tried to predict the ratings of cold-start users on cold-start items. Since there are no historical ratings for either users or items, to show the results of RECT on this challenging scenario, we compared the results of RECT with only RS and KMF. We did not include LCE, LCE-NL and CBF as they are not applicable in this scenario.

Table 6.4 shows the results of applying RS, KMF and RECT algorithms on MovieLens 100K and 1M. On both datasets, RECT outperformed RS and KMF. RS generally performs worse than the other algorithms in cold-start users/items scenarios, which is a common problem in newly launched websites; and confirms that it is necessary to carefully use similarity information of users and items to have a more accurate recommendations. Therefore, RECT is a novel algorithm that can deal with all three types of cold-start problems. We would like to also note that RECT is able to predict an initial guess of an item popularity for cold-start items in online recommender systems accurately.

Chapter 7

Cold-star Recommendation: Ranking Prediction I

7.1 Introduction

Due to the popularity and exponential growth of e-commerce and online streaming websites, a compelling demand has been created for efficient recommender systems to guide users toward items of their interests (e.g. products, books, movies) [2]. In collaborative methods, either *filtering* or *ranking*, by relying on the low-rank assumption on the users' preferences, both users and items are mapped into a latent feature space based on partially observed ratings that are later used to make predictions. In collaborative filtering (CF) methods such as matrix factorization [58], where the aim is to accurately predict the ratings, the latent features are extracted in a way to minimize the prediction error measured in terms of popular performance measures such as root mean square error (RMSE). In spark contrast to CF, in collaborating ranking (CR) models [58, 22, 119, 25], where the goal is to rank the unrated items in the order of relevance to the user, the popular ranking measures such as as discounted cumulative gain (DCG), normalized discounted cumulative gain (NDCG), and average precision (AP) [54] are often employed to collaboratively learn a ranking model for the latent features.

Recent studies have demonstrated that CR models lead to significantly higher ranking accuracy over their traditional CF counterparts that optimize rating prediction. This is important considering the fact that what we really care in recommendation is not the actual values of ratings, but the order of items to be recommended to a specific user. Therefore, the error measures such as RMSE are often hopelessly insufficient, as their place equal emphasis on all the ratings. Among ranking models, the methods that mainly concentrate on the *top of the list* have received a considerable amount of attention, due to the higher probability of examining the top portion of the list of recommendations by users. Therefore, the introduction of ranking metrics such as push norm or infinite norm [99, 5, 22, 59], sparked a widespread interest in CR models and has been proven to be more effective in practice [112, 22].

The general recommendation setting consists of a pool of items and a pool of users, where user feedback such as ratings are available for varying subsets of items. Recommender systems seek to predict the rating that a user would give to an item and further try to suggest items that are most appealing to the users.

Recently, recommender systems have received a considerable amount of attention and have been the main focus of many research studies [2]. As demonstrated by its performance in the KDD Cup [29] and Netflix competition [13], currently the most successful recommendation technique used is collaborative filtering. In factorization based CF methods [58], both users and items are mapped into a latent feature space based on observed ratings that are later used to make predictions.

Although CR models for recommender systems has been studied extensively and some progress has been made, however, the state of affairs remains unsettled: the issue of handling *cold-start* items in ranking models and coping with *not missing at random* assumption of ratings are elusive open issues. First, in many real world applications, the rating data are very sparse (e.g., the density of the data is around 1% for many publicly available datasets) or for a subset of users or items the rating data is entirely missing (knows as cold-start user and cold-start item problem, respectively) [103]. Second, collaborative filtering and ranking models rely on the critical assumption that the missing ratings are sampled uniformly at random. However, in many real applications of recommender systems, this assumption is not believed to hold, as invariably some users are more active than others and some items are rated by many people while others are rarely rated [111]. These issues have been investigated in factorization based methods, nonetheless, it is not straightforward to adapt them to CR models and are left open [22]. Motivated by these challenges, we ask the following fundamental question in the context of collaborative ranking models:

> *Is it possible to effectively learn a collaborative ranking model in the presence of cold-start items/users that is robust to the sampling of observed ratings?*

In this thesis, we give an affirmative answer to the above question. In particular, we introduce a *semi-supervised collaborative ranking* model, dubbed $\texttt{S}^2\texttt{COR}$, by leveraging side information about *both observed and missing ratings* in collaboratively learning the ranking model. In the learned model, unrated items are conservatively pushed after the relevant and before the irrelevant items in the ranked list of items for each individual user. This crucial difference greatly boosts the performance and limits the bias caused by learning only from sparse non-random observed ratings. We also introduce a graph regularization method to exploit the side information about users to overcome the cold-start users problem. In summary, the key features of $\texttt{S}^2\texttt{COR}$ are:

- Inspired by recent developments in ranking at top [99, 5, **?**], the proposed model is a collaborative ranking model that primarily focuses on the top of the recommendation list for each user. Moreover, in stark contrast to pairwise ranking models which have quadratic dependency on the number of items, the proposed ranking model has a linear dependency on the number of items, making it suitable for large-scale recommendation.

- It leverages side information about items with both observed and missing ratings while collaboratively learning the ranking model, which enables it to effectively incorporate the available side information in knowledge transduction.
- By incorporating the unrated items in ranking, it limits the bias caused by learning solely based on the observed ratings and consequently deals with the not missing at random issue of ratings.
- It is also able to leverage the similarity information between users based on a graph regularization method to make high quality recommendations for users with few ratings or cold-start users without an rating information.

To build the intuition on how incorporating missing ratings in $\texttt{S}^2\texttt{COR}$ is beneficial in handling cold-start problem and mitigating data sparsity issue, we note that in many real world applications the available feedback on items is extremely sparse, and therefore the ranking models fail to effectively leverage the available side information in transdcuting the knowledge from existing ratings to unobserved ones. This problem becomes especially eminent in cases where surrogate ranking models such as pairwise models are used due to their computational virtues, where the unobserved ratings do not play any role in learning the model. As a result, by leveraging rich sources of information about all items, one can potentially bridge the gap between existing items and new items to overcome the cold-start problem.

Turning to the non-random sampling issue of observed ratings, we note that the non-randomness is observing the ratings creates a bias in learning the model that negatively impacts the future predictions and may degrade the resulting recommendation accuracy if ignored. Therefore, the nature of missing ratings has to be modeled precisely as to obtain correct results. To reduce the effect of bias, the proposed ranking model takes a conservative approach and pushes the items with unknown ratings to the middle of ranked list, i.e., after the relevant and before the irrelevant items. This is equivalent to assuming a prior about the unknown ratings which is believed to perform well as investigated in [28]. However, unlike [28], the proposed ranking idea is free of deciding an explicit value for missing ratings which makes it more valuable from a practical point of view.

We conduct thorough experiments on real datasets and compare our results with the state-of-the-art models for cold-start recommendation to demonstrate the effectiveness of our proposed algorithm in recommendation at the top of the list and mitigating the data sparsity issue. Our results indicate that our algorithm outperforms other algorithms and provides recommendation with higher quality compared to the state-of-the-art methods.

7.2 Experiments

In this section, we conduct exhaustive experiments to demonstrate the merits and advantages of the proposed algorithm. We conduct our experiments on three well-known datasets MovieLens, Amazon and CiteULike. We investigate how the proposed $\texttt{S}^2\texttt{COR}$ performs in comparison to the state-of-the-art methods. In the following, first we introduce the datasets that we use in our experiments and then the metrics that we employ to evaluate the results, followed by our detailed experimental results on the real datasets.

In the following subsections, we intend to answer these key questions:

- **Ranking versus rating:** How does learning optimization for a ranking based loss function affect the performance of recommending versus the square loss function?
- **Employing missing ratings:** How does employing the missing ratings could help in making more accurate recommendations?
- **Dealing with cold-start items:** How does the proposed algorithm, with incorporating side information of items, perform in comparison to the state-of-the-art algorithms to deal with cold-start items?

7.2.1 Datasets

We use the following well known datasets to evaluate the performance of $\texttt{S}^2\texttt{COR}$:

- **ML-IMDB.** We used ML-IMDB which is a dataset extracted from the IMDB and the MovieLens 1M datasets by mapping the MovieLens and IMDB and collecting the movies that have plots and keywords. The rating values are 10 discrete numbers ranging from 1 to 10 and the rating were made binary by treating all the ratings greater than 5 as $+1$ and below 5 as -1.

- **Amazon.** We used the dataset of best-selling books and their ratings in Amazon. Each book has a one or two paragraphs of textual description, which has been used to have a set of features of the books. Ratings can be integers numbers from 1 to 5. The ratings were also made binary by treating all the ratings greater or equal to 3 as $+1$

Table 7.1: Statistics of real datasets used in our experiments.

Statistics	ML-IMDB	Amazon	CiteULike
Number of users	2,113	13,097	3,272
Number of items	8,645	11,077	21,508
Number of ratings	739,973	175,612	180,622
Number of features	8,744	5,766	6,359
Average number of ratings by users	350.2	13.4	55.2
Average number of ratings for items	85.6	13.4	55.2
Density	4.05%	0.12%	0.13%

and below 3 as -1.

- **CiteULike.** It is an online free service for managing and discovering scholarly references. Users can add those articles that they are interested in to their libraries. Collected articles in a user's library will be considered as relevant items for that user. This dataset does not have explicit irrelevant items and was chosen to illustrate the effect of considering missing data while only having relevant itmes.

For all above datasets, the description about the items were tokenized and after removing the stop words, the rest of the words were stemmed. Then those words that have been appeared in less than 20 items and more that 20% of the items were also removed [105]. At the end, the TF-IDF was applied on the remaining words and the TF-IDF scores represented the features of the items. The statistics of the datasets are given in Table 7.1. As it is shown in Table 7.1, all these three datasets have high dimensional feature space.

7.2.2 Metrics

We adopt the widely used metrics, Discounted Cumulative Gain at n and Recall at n, for assessing the performance of our and baseline algorithms. For each user u, given an item i, let s_k be the relevance score of the item ranked at position k, where $s_k = 1$ if the item is relevant to the user u and $s_k = 0$ otherwise. Now, given the list of top-n item recommendations for user u, Discounted Cumulative Gain at n, is defined as:

$$\mathrm{DCG_u}@n = s_1 + \sum_{k=2}^{n} \frac{s_k}{\log_2(k)}$$

If we divide the $\mathrm{DCG_u}@n$ by its maximum value, we get the $\mathrm{NDCG_u}@n$ value. Given the list of top-n item recommendations for each user u, Recall at n will count the number of relevant items appeared in the recommendation list divided by the size of the list. Recall at n is defined as:

$$\mathrm{REC_u}@n = \frac{|\{\text{relevant items to } u\} \cap \{\text{top-}n \text{ recommended items}\}|}{|\{\text{top-}n \text{ recommended items}\}|}$$

$\mathrm{DCG}@n$, $\mathrm{NDCG_u}@n$ and $\mathrm{REC}@n$ will be computed for each user and then will be averaged over all users.

7.2.3 Methodology

Given the partially observed rating matrix, we transformed the observed ratings of all datasets from a multi-level relevance scale to a two-level scale $(+1, -1)$ while 0 is considered for unobserved ratings. We randomly selected 60% of the observed ratings for training and 20% for validation set and consider the remaining 20% of the ratings as our test set. To better evaluate the results, we performed a 3-fold-cross validation and reported the average value for our results.

7.2.4 Baseline Algorithms

The proposed S^{2}COR algorithm is compared to the following algorithms:

- **Matrix Factorization (MF) [110]:** Is a matrix completion method that factorizes the incomplete observed matrix and completes the matrix using the unveiled latent features.
- **Matrix Factorization with Side Information (KPMF) [122]:** Is a matrix completion based algorithm, which incorporates external side information of the users or items into the matrix factorization process. It imposes a Gaussian Process prior over all rows of the matrix, and the learned model explicitly captures the underlying correlation among the rows.
- **Decoupled Completion and Transduction (DCT) [9]:** Is a matrix factorization based algorithm that decouples the completion and transduction stages and exploits the similarity information among users and items to complete the (rating) matrix.
- **Feature Based Factorized Bilinear Similarity Model (FBS) [105]:** This algorithm uses bilinear model to capture pairwise dependencies between the features. In particular, this model accounts for the interactions between the different item features.
- **Collaborative User-specific Feature-based Similarity Models (CUFSM):** By using the history of ratings for users, it learns personalized user model across the

dataset. This method is one of the best performing collaborative latent factor based model [31].

- **Regression based Latent Factor Model (RLF):**[1] This method incorporates the features of items in factorization process by transforming the features to the latent space using linear regression [3]. If the learning method is Markov Chain Monte Carlo, we name it RLF-MCMC.

- **Cosine Similarity Based Recommender (CSR):** Using the similarity between features of items, the preference score of a user on an item will be estimated.

We would like to mention that as baseline algorithms we only consider state-of-the art methods that are able to exploit the side information about items.

7.2.5 $\texttt{S}^2\texttt{COR}$ vs. rating

Many different algorithms are trying to provide recommendations to users such that the predicted rating values be very close to the actual rates that users would provide. These algorithms try to minimize the error between the predicted values and actual rating values by minimizing Mean Squared Error (MAE) or Root Mean Square Error (RMSE) or etc. Then, due to the fact that users tend to only care about the top of their recommendation list, predicting a ranking of items of interest instead of ratings became the main focus of recent works [22]. In this section we compare the results of $\texttt{S}^2\texttt{COR}$ with those state-of-the-art algorithms that try to predict ratings for unrated items. Among the state-of-the-art algorithms, we chose a diverse set of algorithms, which are Matrix Factorization, Matrix Factorization with Side Information, Decoupled Completion and Transduction and

[1]The implementation of this method is available in LibFM library [95].

Table 7.2: Results of comparing rankings methods versus ratings methods on ML-IMDB. λ is regularization parameter, h is dimension of latent features, T is the number of iterations, η is learning rate and σ is the standard deviation.

Algorithms	**Hyperparameters**	**NDCG@10**
MF	$\eta = 0.003, h = 10, \sigma = 0.4$	0.0947
KPMF	$\eta = 0.003, h = 10, \sigma = 0.4$	0.1005
DCT	$h = 10$	0.0095
RLF-MCMC	$T = 100, \sigma = 0.1$	0.0248
$\texttt{S}^2\texttt{COR}$	$\lambda = 0.55, h = 10, T = 100$	**0.265**

Regression Based Latent Factor Model. Table 7.2 shows the NDCG value of top 10 items of recommendation list. It shows that $\texttt{S}^2\texttt{COR}$ outperformed all other rating prediction based algorithms in terms of NDCG measure. The results confirm the effectiveness of providing the ranks of items rather than their ratings.

7.2.6 Robustness to not missing at random ratings

In this section we compare the effect of incorporating the unobserved ratings in our learning in comparison with excluding them from our learning. Most of the methods in the literature ignore the unobserved ratings and train their model only base on observed ratings. By incorporating the unrated items in ranking, our method can limit the bias caused by learning solely based on the observed ratings and consequently deals with the not missing at random issue of ratings. Table 7.3 shows results of comparing these two scenarios for $\texttt{S}^2\texttt{COR}$ on ML-IMDB. In order to see the difference between these two scenarios, we considered 70% of the ratings for training and 30% for test to have more ground truth for our testing. Table 7.3 shows the NDCG@5, 10,15 and 20 for both scenarios and it shows that incorporating the unobserved ratings causes to improve the accuracy of recommendation list. Hence, the NDCG values for top 5, 10, 15 and 20 items improved when unrated items were included as

Table 7.3: Results of employing missing ratings versus ignoring them on ML-IMDB. $\lambda = 0.6$ is regularization parameter, $h = 10$ is dimension of latent features, $T = 100$ is the number of iterations.

Algorithm: $\texttt{S}^2\texttt{COR}$	**NDCG@5**	**NDCG@10**	**NDCG@15**	**NDCG@20**
Observed ratings	1.1690	2.2218	2.8362	3.2849
Observed + missing ratings	**1.1794**	**2.2405**	**2.8585**	**3.3096**

part of the training process.

7.2.7 Dealing with cold-start items

We now turn to evaluating the effectiveness of $\texttt{S}^2\texttt{COR}$ for cold-start recommendation. To do so, we randomly selected 60% of the items as our training items and 20% for validation set and considered the remaining 20% of the items as our test set. In this scenario, baseline algorithms that are used for comparison are CSR, FBS, CUFSM and RLF. For the experiments, we used ML-IMDB, Amazon and CiteULike datasets. Table 7.4 shows the measurement results of applying mentioned algorithms on these datasets. For each test, the parameters' values producing the best ranking on the validation set were selected to be used and reported. As it can be seen from the results in Table 7.4, the proposed $\texttt{S}^2\texttt{COR}$ algorithm outperformed all other baseline algorithms and provided a recommendations with higher quality in comparison to other methods. We can also see from the results of Table 7.4 that for the ML-IMDB dataset, the improvement in terms of REC@10 is significant compared to other datasets. Since the density of this dataset is much higher than other two datasets, this observation indicates that our method is more effective in utilizing side information compared to other methods. These results demonstrate the effectiveness of $\texttt{S}^2\texttt{COR}$ in comparison with other state-of-the-art algorithms. $\texttt{S}^2\texttt{COR}$ was able to outperform other state-of-the-art algorithms by considering the missing data and focusing on top of the recommendation list

Table 7.4: Results on cold-start items. λ, μ_1 and β are regularization parameters, h is dimension of latent features, l is the number of similarity functions and T is the number of iterations.

	Algorithms	**Hyperparameters**	**DCG**@10	**REC**@10
ML-IMDB	CSR	—	0.1282	0.0525
	RLF	$h = 15$	0.0455	0.0155
	CUFSM	$l = 1, \mu_1 = 0.005$	0.2160	0.0937
	FBS	$\lambda = 0.01, \beta = 0.1, h = 5$	0.2270	0.0964
	S^{2}COR	$\lambda = 0.6, h = 10, T = 200$	**0.2731**	**0.2127**
Amazon	CSR	—	0.0228	0.1205
	RLF	$h = 30$	0.0076	0.0394
	CUFSM	$l = 1, \mu_1 = 0.25$	0.0282	0.1376
	FBS	$\lambda = 0.1, \beta = 1, h = 1$	0.0284	0.1392
	S^{2}COR	$\lambda = 0.6, h = 10, T = 200$	**0.1195**	**0.1683**
CiteULike	CSR	—	0.0684	0.1791
	RLF	$h = 75$	0.0424	0.0874
	CUFSM	$l = 1, \mu_1 = 0.25$	0.0791	0.2017
	FBS	$\lambda = 0.25, \beta = 10, h = 5$	0.0792	0.2026
	S^{2}COR	$\lambda = 0.6, h = 10, T = 200$	**0.0920**	**0.2243**

for cold-start items.

Chapter 8

Cold-star Recommendation: Ranking Prediction II

8.1 Introduction

Recommender systems have become ubiquitous in recent years, and are applied in a variety of applications [53]. Typically, the recommendation system selects a small set from the underlying pool of items (e.g., products, news, movies, books) to recommend to an active user. The user can either like (e.g., read, click, buy, etc.) or dislike the items in the list, or take no action. Content-based filtering (CB) and collaborative filtering (CF) are well-known examples of recommendation approaches. As demonstrated in KDD Cup [29] and Netflix competition [13], the most successful recommendation technique used is collaborative filtering which exploits the users' opinions (e.g., movie ratings) and/or purchasing (e.g., watching, reading) history in order to extract a set of interesting items for each user. In fictionalization based CF methods, both users and items are mapped into a latent feature space based on observed ratings that are later used to make predictions.

Despite significant improvements on recommendation approaches, and in particular collaborative filtering based methods, these systems suffer from a few inherent limitations that must be addressed. In particular, these techniques suffer from data sparsity in real-world scenarios and fail to address users who rated few or no items or items that have not been rated by any user– commonly known as cold-start users and items problems, respectively. For instance, according to [101], the density of extant ratings in most commercial recommender systems is often much less than one.

In recent years there has been an upsurge of interest in exploiting social information such as trust relations among users along with rating data to improve the performance of recommender systems and resolve sparsity and cold-start problems (see e.g. [117, 106, 35] for more recent surveys). A well-known example is the `FilmTrust` system [41]. This is an online social network that provides movie rating and review features to its users. The social networking component of the website requires users to provide a trust rating for each person they add as a friend. The main motivation for exploiting trust information in recommendation process stems from the observation that the ideas we are exposed to and the choices we make are significantly influenced by our social context. For example, in [108] it was pointed out that people tend to rely more on recommendations from others they trust more than recommendations based on anonymous people similar to them. This observation, combined with the growing popularity of open social networks, has generated a rising interest in trust-enhanced recommendation systems. Based on this intuition, a few trust-aware recommendation methods have been proposed [69, 74, 51].

Recently, online networks where two opposite kind of relationships can occur have become common. For instance, the `Epinions` [44], an e-commerce site for reviewing and rating products, allows users to evaluate others based on the quality of their reviews, and make trust and distrust relations with them. Similar patterns can be found in online communities such as `Slashdot` in which millions of users post news and comment daily and are capable of tagging other users as friends/foes or fans/freaks. Additionally, users on `Wikipedia` can vote for or against the nomination of others to adminship [16]. When more users issuing distrust statements, the more important it will become to exploit this new information source in recommender systems.

It is acknowledged that along with the trust relationships, also distrust can play an important role in boosting the accuracy of recommendations [117, 116, 35]. Recently, some attempts have been made to explicitly incorporate the distrust relations in recommendation process [44, 70, 116]. This demonstrated that the recommender systems can benefit from the proper incorporation of distrust relations in social networks. However, despite these positive results, there are some unique challenges involved in distrust-enhanced recommender systems. In particular, it has proven challenging to model distrust propagation in a manner which is both logically consistent and psychologically plausible. Furthermore, the naive modeling of distrust as negative trust raises a number of challenges- both algorithmic and philosophical. Finally, it is an open challenge how to best incorporate trust and distrust relations in model-based methods, e.g., matrix factorization, simultaneously. In memory based recommender systems, the simultaneous exploitation of trust and distrust has been investigated in [118, 115].

An attempt to simultaneously exploit trust and distrust relations in factorization based method has been made very recently in [35]. In particular, a ranking model was proposed to rank the latent features of users, from the perspective of individual users, that respects their social context. Despite the encouraging improvements that has been brought by simultaneous exploitation of trust and distrust relations, the proposed algorithm suffers from two issues. First, as the number of constraint triplets imposed from social regularization of latent features can increase cubically in the number of users in the social network, it is computationally difficult to make their idea scalable to large social graphs. Second, the algorithm proposed in [35] only considers the trusted and distrusted friends of each user in order to regularize the latent features. Thus, it ignores the neutral friends – the users who has no relation to the user. As a result, the neutral friends might appear before the trusted friends in the ranked list. Therefore neutral friends might become more influential than trusted friends and impact the recommendation negatively.

In this paper we describe and analyze a fairly general and flexible method to learn and infer from rating data, along with trust and distrust relationships between users. The main building block of our proposed algorithm, dubbed `PushTrust`, is an efficient algorithm to rank the latent features of users by leveraging their social context. In ranking of latent features, we wish to make sure that the top portion of the list includes the trusted friends and the distrusted friends are pushed to the bottom of list. Also, we would like the neutral friends appear after trusted and before distrusted friends in the list. Compared to the method proposed in [35], the key features of the `PushTrust` algorithm are: (i) its quadratic time complexity in the number of users, and (ii) its ability to take the neutral friends of users into the consideration in regularizing the latent features of users. Thorough experiments on the Epinions dataset demonstrate the merits of the proposed algorithm in boosting the quality of recommendations and dealing with data sparsity and cold-start users problems.

Outline. The sections is organized as follows. In Section 5 we gave a formal definition of the setting and briefly reviewed the existing social enhanced matrix factorization methods. The `PushTrust` algorithm and its associated optimization method are presented in Section 5.2 and 5.3. Experiments are provided in Section 8.2

8.2 Experiments

In this section, we conduct several experiments to compare the recommendation qualities of our `PushTrust` algorithm with other state-of-the-art recommendation methods. We begin with a brief description of the Epinions dataset and the evaluation metrics, and then introduce the baseline methods, followed by discussion of our experimental results.

8.2.1 The Epinions dataset

To evaluate the proposed algorithm on trust- and distrust-aware recommendations, we use the Epinions dataset, the only social rating network dataset publicly available. Moreover, an important reason why we choose the Epinion dataset is that user trust and distrust information is included in this dataset. Epinions allows users to evaluate other users based on the quality of their reviews. It additionally provides trust and distrust evaluations in addition to ratings. The social relations in in Epinions are directed.

Table 8.1 shows the statistics of the Epinions dataset used in our experiments. To conduct the coming experiments, we sampled a subset of Epinions dataset with $n = 121,240$ users and $m = 685,621$ different items. The total number of observed ratings in the sampled dataset is 12,721,437 which approximately includes 0.02% of all entries in the rating matrix $\mathbf{R}$. This demonstrates the sparsity of the rating matrix. The social network includes 481,799 and 96,823 trust and distrust relations between users, respectively.

8.2.2 Evaluation metrics

We adopt the widely used the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) [50] metrics for prediction accuracy. The MAE metric is very appropriate and

Table 8.1: Statistics of rating data and social network in the sample Epinions dataset used in our experiments.

Statistic	Quantity
Number of users	121,240
Number of items	685,621
Number of ratings	12,721,437
Number of trust relations	481,799
Number of distrust relations	96,823

useful measure for evaluating prediction accuracy in offline tests [50]. Let $\mathcal{T}$ denote the set of ratings to be predicted, i.e., $\mathcal{T} = \{(i,j) \in [n] \times [m], \mathbf{R}_{ij} \text{ needs to be predicted}\}$ and let $\widehat{\mathbf{R}}$ denote the prediction matrix obtained by a recommendation algorithm. Then, the MAE and RMSE metrics are defined as:

$$\text{MAE} = \frac{1}{|\mathcal{T}|} \sum_{(i,j)\in\mathcal{T}} |\mathbf{R}_{ij} - \widehat{\mathbf{R}}_{ij}|,$$

and

$$\text{RMSE} = \sqrt{\sum_{(i,j)\in\mathcal{T}} \left(\mathbf{R}_{ij} - \widehat{\mathbf{R}}_{ij}\right)^2 / |\mathcal{T}|}.$$

The first measure (MAE) considers every error of equal value, while the second one (RMSE) emphasizes larger errors.

8.2.3 Baseline algorithms

In order to demonstrate the benefits of our approach, we compare our model with the following factorization based methods with and without social regularization.

- **MF** (matrix factorization based recommender): This algorithm is the basic matrix factorization in Eq. (5.1), which does not take the social data into account.

- **MF-T** (matrix factorization with trust information): The matrix factorization algorithm which exploits the trust relations between users in factorization [70] as formulated in Eq. (5.2).
- **MF-D** (matrix factorization with distrust information): The algorithm proposed in [70] to only exploit distrust relations as formulated in Eq. (5.3).
- **MF-TD** (matrix factorization with trust and distrust information): The algorithm proposed in [35] to incorporate both trust and distrust relationships as formulated in Eq. (5.4).
- **PushTrust**: The algorithm proposed in this paper.

Experimental setup. For testing our algorithm, we perform 5-fold cross validation in our experiments. In each fold 80% of the rating data was randomly selected as the training set and the remaining 20% as the test data. After optimization over training data, the extracted latent models are used to predict the test data.

8.2.4 Comparison to baseline algorithms

We now focus on check how do the trust, distrust and neutral relationships between users affect the recommendation accuracy. In other words, we would like to experimentally evaluate whether incorporating different social relationships can indeed enhance the trust-based recommendation process. In this section, we present results on accuracy of rating prediction in terms of RMSE and MAE. We compare our approach with the four state-of-the-art algorithms discussed above: MF, MF-T, MF-D, and MF-TD on the Epinions.

We run the algorithms with different latent vector dimensions, $k = 5$ and $k = 10$. We evaluate these algorithms by measuring both MAE and RMSE errors. The parameters of all

Table 8.2: Comparison with other methods in terms of MAE and RMSE errors.

k	Method	MAE	RMSE
5	MF	0.9512	1.2612
	MF-T	0.8735	1.2030
	MF-D	0.9137	1.2230
	MF-TD	0.8312	1.1023
	PushTrust	**0.8010**	**1.0802**
10	MF	0.9763	1.2725
	MF-T	0.8930	1.2134
	MF-D	0.9234	1.2300
	MF-TD	0.8432	1.1450
	PushTrust	**0.8216**	**1.1230**

algorithms are tuned to achieve the best performance. Table 8.2 reports the MAE and RMSE of all comparison algorithms on the Epinions dataset. We unsurprisingly observed that the performance of algorithms exploiting social information are better than the pure matrix factorization based algorithm. Additionally the proposed algorithm outperforms the other baseline algorithms in all cases. It is also surprising to see that increasing k in the Epinions dataset did not improve the results while we know that by increasing k, the flexibility of the algorithm will be increased which should yield more accurate results. The Epinions dataset is small and increasing k creates more parameters in the model which leads to overfitting. We also note that MF-T performs better than the MF-D due to large number of trust relations in dataset compared to the number of distrust relations.

8.2.5 Experiments on handling cold-start users

Handling cold-start users (i.e., users with few ratings or new users) is one the main challenges in existing recommender systems. For example, more than 50 % of users in the Epinions are cold-start users. Hence efficiency of any recommendation algorithm for cold-start users becomes very important. In this set of experiments, the performance of proposed algorithm

Table 8.3: The accuracy of handling cold-start users and the effect of social relations. The number of latent features in thes experiments is $k = 5$.

% Cold-start Users	Measure	MF	MF-T	MF-D	MF-TD	PushTrust
30%	MAE	0.9923	0.9124	0.9721	0.8603	0.8345
	RMSE	1.7211	1.5562	1.6433	1.4602	1.3362
20%	MAE	0.9812	0.8905	0.9405	0.8542	0.8268
	RMSE	1.6962	1.4339	1.5250	1.2630	1.2430
10%	MAE	0.9634	0.8802	0.9233	0.8480	0.8204
	RMSE	1.3222	1.2921	1.3105	1.1688	1.1207

is evaluated on clod-start users and compared to baseline algorithms. To evaluate different algorithms we randomly select 30%, 20%, and 10% of the users, to be cold-start users. For cold-start users, we do not include any rating in the training data and consider all ratings made by cold-start users as testing data.

Table 8.3 shows the performance of above mentioned algorithms. As it is clear from the Table 8.3, when the number of cold-start users is high, exploiting the trust and distrust relationships significantly improves the performance of recommendation. This result is interesting as it reveals that the lack of rating information for cold-start and new users can be alleviated by incorporating the social relations of users. The improvement becomes significant by exploiting both trust and distrust relationships which amplifies the importance of social data when exploited in an appropriate way.

The results reported in that Table 8.3 imply that `PushTrust` is able to handle cold-start users more effectively than other algorithms. We also note that `PushTrust` outperforms the recently developed MF-TD algorithm. This is because the MF-TD ignores the neutral friends but `PushTrust` is designed to push these users to appear after the trusted friends and before the distrusted friends in the ranked list of latent features. We note that without considering neutral friends, their latent features might become even closer than the trusted friends of a specific user and consequently influence the predictions negatively. This crucial difference shows significant improvement in the performance as revealed by the results in Table 8.3.

Chapter 9

Network Completion

9.1 Introduction

Gathering complete knowledge of the structure of very large networked datasets is of pivotal importance to most network analysis applications. However, determining this structure in networks, such as: social, biological, and publication networks, is very difficult due to the sheer size of the data coupled with its general sparsity. Because of these difficulties, it is reasonable to try and make structural measurements using a partially, and partially observed, sample of a network and extrapolate the network's properties based on this sample. Such inferences are known as *network completion* or *survey sampling.* In a graph, this is done by selecting a fixed number of nodes at random and observing all, or a uniform sample, of the links incident to these nodes. In this work, we attempt network completion by recovering the topology of an undirected network by observing only a partially-observed random sample from the network with "enough" links. [55].

The need for effective and efficient network completion algorithms occurs in a wide variety of settings such a information retrieval, social network analysis, and computational biology. With the explosion of big data, possession of data is not often an issue, but rather characterization of said data is very difficult. For information retrieval problems, there may be natural structure in the set, but only a small sampling may be feasibly obtained to determine the placement of the objects into various classes. Here network completion's attempt is to use that small sample to determine the rest of the network. Similar to the information retrieval problem, the massive user adoption of social networks have both increased the reward in understanding the underlying social structure and also the difficulty in doing so. There are billions of users among Facebook, Twitter, and other social networks. Inferring the full network from a small sample of links between users, or finding the implicit social networking structure between them, is a challenging problem and the subject of recent research [86]. In computational biology [7], most of the crucial questions relate to understanding the complex interactions between entities, such as genes or proteins. These interactions form large networks and their complex structure aids biologists in determining the role of proteins in a biological system. These structural insights allow for the formulation, and experimental testing, of specific hypotheses about gene function. Unfortunately, the available experimental techniques can not sample the complete system, but rather only a tiny fraction of it. In building a complete biological network, again the network completion problem arises.

Many methods exist to analyze, contextualize, and estimate connections in large graphs and the temptation to apply them to our network completion problem is strong. They often use well established statistical and machine learning tools and they work very well under many precisely defined conditions. However, they struggle to handle the cases in which we are interested due to size and randomness restrictions.

Each of the examples of problems that may be aided with network completion techniques, seemingly could rely on simply choosing very good subsets of the given data. From this statistical standpoint, the general trend to cope with the difficulty of understanding the structure of large-scale networks has been on finding effective sampling methods therein. The idea here is that a small, but carefully chosen sample can be used to represent the population. There are specialized techniques such as stratified sampling and cluster sampling that improve the precision or efficiency of the sampling process [36]. However, due to security, user privacy, and data aggregation overhead the existence of missing data is almost inevitable. This absent data misleads estimation of full network properties. Also, in many applications such as social recommendation, a global structure of the network is required to be inferred. Since we do not observe any links for unsampled nodes, the statistical models are not feasible and, consequentially, they do not perform well for these types of problems.

Another attempt at solving this problem uses learning techniques. Under the scope of learning, the network completion problem may be cast as an inference problem. Through these inferences, it aims to learn the structure of a network from a sampled subgraph. Yet, there are two major issues in learning the network topology from random samples. First, the links present in the observable sample are not chosen uniformly. This is typically required in order to apply most well-known techniques such as matrix factorization or matrix completion [93]. Secondly, there are huge number of nodes in the network for which no edge is sampled. This, again, makes these standard methods infeasible.

Considering the aforementioned limitations, we aim to exploit auxiliary information to improve the network completion. Usually we have some information about the nodes of the partially observed network. Furthermore, exploiting other sources of information about the nodes provides useful information about the underlying structure of the network. For example, in a social network, the similarity between users based on their profile information is often seen as a major factor for connectivity in social networks. According to the homophily principle, people tend to connect to those with who are sharing similar interests, tastes, beliefs, social backgrounds, popularity, race and etc [75]. In biological networks, the interactions between proteins or other molecules require an exact or complementary fit of their complex surfaces. This can be treated anonymously with similarity in the context of connectivity or micro-array expression data [107]. Therefore, if the rich auxiliary information is appropriately integrated into network models, the network model can necessarily have more predictive power.

More generally, using auxiliary information may take the form of identifying similar users, so that the unobserved links can be approximated with the observed ones, or determining graph structures that should exist in unobserved nodes. This similarity information about nodes can complement the network structure, leading to more precise prediction of links. Moreover, if one source of information is missing or noisy, the other can complement it. Thus, it is important to consider both sources of information together in completion, leading to an interesting research question as:

> *How to effectively incorporate side information about nodes to learn the structure of a network from a sampled subgraph of the underlying network?*

This problem is the main focus of this chapter, and our goal is to reconstruct the underlying network without any knowledge about the topological features of the underlying network, but only based on a partially observed subgraph of the network and auxiliary features of nodes.

To complete an unobserved network, we propose an efficient algorithm that significantly departs from the existing methods with side information such as shared subspace learning [32] and matrix completion with transduction [42]. Our proposed algorithm consists of two *decoupled* stages: completion and transduction. In the first step, we only complete the largest sub-network that can be carried out perfectly following the matrix completion theory. Specifically, under some mild assumptions on the number of sampled links and coherence of the matrix it may be perfectly completed (e.g., [93]). Then, the recovered sub-network along with available similarity side information about nodes is utilized to transduct the knowledge to fully unobserved nodes. We provide theoretical performance guarantees on the estimation error of the proposed algorithm. This is in contrast to most existing methods that do not come with theoretical support. Experimental results show that by simultaneously exploiting the sampled subgraph as well as the similarity information between nodes extracted from other sources of information, our method performs significantly better than natural alternatives and the state-of-the-art methods.

Outline. The remainder of the chapter is organized as follows. In Section 3.2.1, we begin by establishing notation and formally describing our setting. We then in Section 3.2.2 describe the proposed algorithm and prove our main theoretical result on its performance in Section 3.3. The experimental results are provided in Section 9.2.

9.2 Experimental Evaluation

To better understand the performance of the proposed algorithm, we divide our experiments into two different scenarios. In the first scenario, we apply the proposed algorithm on a few well-known social network data sets and compare them with the baseline link prediction algorithms. In the second scenario, we compare the proposed algorithm with the state-of-the-art algorithms for network completion . The main reason for splitting our experiments into two parts is the unique challenges that exist in network completion due to the sparseness information of the links, which makes it a challenging problem. We note that different algorithms might achieve totally different performances on these two scenarios. The application of the proposed algorithms to link prediction and network completion on data sets will be discussed in Sections 9.2.5 and 9.2.6. We begin by introducing the data sets we have used in our experiments. Then compare the proposed algorithm to a number of well-known algorithms according to their quality on both synthetic and multiple popular real world social networks.

9.2.1 Datasets

We have selected four real world datasets on which we performed our experiments. Two of these datasets, which are Epinions and Weibo will be used in our first application, which is link prediction problem. The other two datasets, Facebook and Google+, are used in our second application which is network prediction problem.

Facebook dataset. Facebook is one of the most popular real world social network datasets. On Facebook, people have directed friendship relationships with each other and each user has a profile containing information about the user. For each user (a node of our network) features such as gender, job title, age and etc are extracted from their profiles. The network of users is produced by combining ego-networks of Facebook dataset available at the Stanford Large network Dataset Collection [1]. The statistics of datasets are summarized in Table 9.1. To generate the similarity matrix between users, we used the cosine similarity metric between the extracted features and computed the pairwise similarities for all users.

Google+ dataset. The second real social network that we experimented on is Google+. It is similar to Facebook in terms of relationships between users and each user has her own profile. The features of users, such as gender, age and etc, are extracted from their profile

[1] `http://snap.stanford.edu/data`

and the ego-networks of users are combined together to have a network of users along with their features. This dataset is also available at the Stanford Large network Dataset Collection [2] and the statistics are shown in Table 9.1. To compute the similarities between users, we used the cosine similarity to calculate the pairwise similarity between all users.

Epinions dataset. Epinions is an explicit trust/distrust social network creating a directed network. Since the number of distrust relations is very few, we only consider explicit trust relationships between users. The statistics of this dataset are summarized in Table 9.1. This dataset is available through Jiliang Tang at Arizona State University [3].

Generally, the Epinions dataset is used to predict ratings given the trust graph of its users as side information. We, conversely, are predicting the existence of links via a link prediction process with our algorithm while using the rating information as side information. For extracting the features, each user is endowed with a collection of their ratings of the items. These ratings contain scores and timestamps. To extract more features, we also included information about the items that users were interested in, such as names and categories. Features further include a normalized triple for each item category. This gives the total number of reviews in that category along with the number of positive and negative reviews per category normalized by the maximum value of each present in the data. For pairwise similarities between users, we used cosine similarity to generate the similarity matrix of users.

Tencent Weibo dataset. Tencent Weibo is a Chinese microblogging site similar to Twitter. There is a network structure where users can ‘follow’ another to create a link. In addition

[2] `http://snap.stanford.edu/data`

[3] `http://www.public.asu.edu/~jtang20/datasetcode/truststudy.htm`

to the large network, the side information is quite rich.

We extracted the features for the anonymized users that includes personal information (age, gender, and interests), social information (number of messages sent, messages reposted, users followed, etc.), and item classification (categories and keywords). A follower-followee directed graph is given, and we use our algorithm to predict the existence of a directed edge that represents whether or not a given user will follow another. We further only considered positive links (where a user follows another), and ignored all negative links (where a user explicitly choses not to follow another) for prediction. The data is available from the KDD cup [4].

The original Tencent Weibo social directed graph has 1.4 million users and over 73 million links. We wanted to chose a small subset of approximately 4,000 users with many observed links among them and a reasonable amount of side information. To achieve this we used a modified breadth-first search that penalized adding nodes that did not contain links going back into the already explored nodes. Once this process terminated, other (possibly isolated) nodes at random were added if they had a large amount of side-information until our graph's order threshold was met. The observed average ratio of directed links to nodes for random directed graphs of order approximately 4,000 was 0.46. With our method we found a directed subgraph of 4,052 vertices with a significantly higher than average ratio of directed links to nodes at 0.98. The statistics of data-subsets are summarized in Table 9.1. To generate the similarity matrix of users, we used the cosine similarity between all pairs of users.

[4] `http://www.kddcup2012.org/c/kddcup2012-track1/data`

Table 9.1: Statistics of Facebook and Google+ datasets.

Dataset	# of Nodes	# of links	# of Node Features
Facebook	4,089	170,174	175
Google+	250,469	30,230,905	690
Epinions	90,879	365,422	90,087
Weibo	4,052	3,957	16,786

9.2.2 Evaluation Metrics

Evaluation metrics. We evaluate the performance of different algorithms by measuring discrepancy between the original and completed matrices. We adopt the widely used Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) metrics [55] for evaluation. These measured are formally defined as follows. Let $\mathbf{A}$ and $\widehat{\mathbf{A}}$ denote the full and estimated adjacency matrices, respectively. Let $\mathcal{T}$ be the set of unobserved test links. Then,

$$\text{MAE} = \frac{\sum_{(i,j)\in\mathcal{T}} |\mathbf{A}_{ij} - \widehat{\mathbf{A}}_{ij}|}{|\mathcal{T}|},$$

The RMSE metric is defined as:

$$\text{RMSE} = \sqrt{\frac{\sum_{(i,j)\in\mathcal{T}} \left(\mathbf{A}_{ij} - \widehat{\mathbf{A}}_{ij}\right)^2}{|\mathcal{T}|}}.$$

9.2.3 Baseline Algorithms

To evaluate the performance of our proposed algorithm, we consider a variety of baseline approaches. We categorize our baseline algorithms into two categories, first set includes those that are used in the link prediction experiments and the second set includes those that are used in our network completion experiments.

Link prediction. The baseline algorithms are chosen from variety types of link prediction algorithms algorithms to have a fair comparison and all implementation are available in MyMediaLite Package [38].

- **BPRMF** Matrix factorization with Bayesian personalized ranking (BPR) from implicit feedback produces rankings from pairwise classifications. The matrix factorization model provides item prediction optimized for BPR.
- **BiPolar Slope One (BPSO):** Slope One rating prediction methods weighted by bipolar frequency.
- **Matrix Factorization (MF):** Matrix factorization where learning is provided by stochastic gradient descent factoring the observed ratings into user and item factor matrices.
- **Slope One (SO)** Slope One rating prediction method with frequency rating.
- **User-Item Baseline (UIB):** Assigns an average rating value, and regularization, for baseline predictions. Uses the average rating value, plus a regularized user and item bias for prediction.
- **CoClustering (CC):** Performs simultaneous clustering on both the rows and the columns of the rating matrix.
- **Latent Feature log-linear Model (LFM):** Rating prediction method that uses a log-linear model on the latent features of the system. Latent-feature log linear model

- **Biased Matrix Factorization (BMF):** Matrix factorization with parameters for biases of users and items. Utilizes learning provided by stochastic gradient descent.

- **SVD Plus Plus (SPP):** Singular value decomposition matrix factorization method that makes user of implicit feedback. Further considered what items and users each user has rated.

- **Sigmoid SVD Plus Plus (SSPP):** Singular value decomposition matrix factorization method that makes user of implicit feedback and utilizes a sigmoid function.

- **SigmoidItem Asymmetric Factor Model (SFM):** Asymmetric factor model that represents the items based on how the users rated them.

Network completion. The baseline algorithms are chosen from three different types of network completion algorithms to have a fair comparison: (i) methods that only utilize the observed links, (ii) methods that use network structure as well as node features, and (iii) methods that learn shared space.

- **MF:** The first class of baseline algorithms consider only the network structure and ignore the node features. We select the Matrix Completion (MF) algorithm in this class [110] for our comparison.

- **MF-NF:** Second class of baseline algorithms include methods that employ both node features and the structure of network. We choose a matrix factorization based algorithm that factorizes the adjacency matrix and combines the latent features with explicit features of nodes and links using a bilinear regression model [79]. The implementation of this algorithm is provided by the authors [5].

[5] `http://cseweb.ucsd.edu/~akmenon/code/`

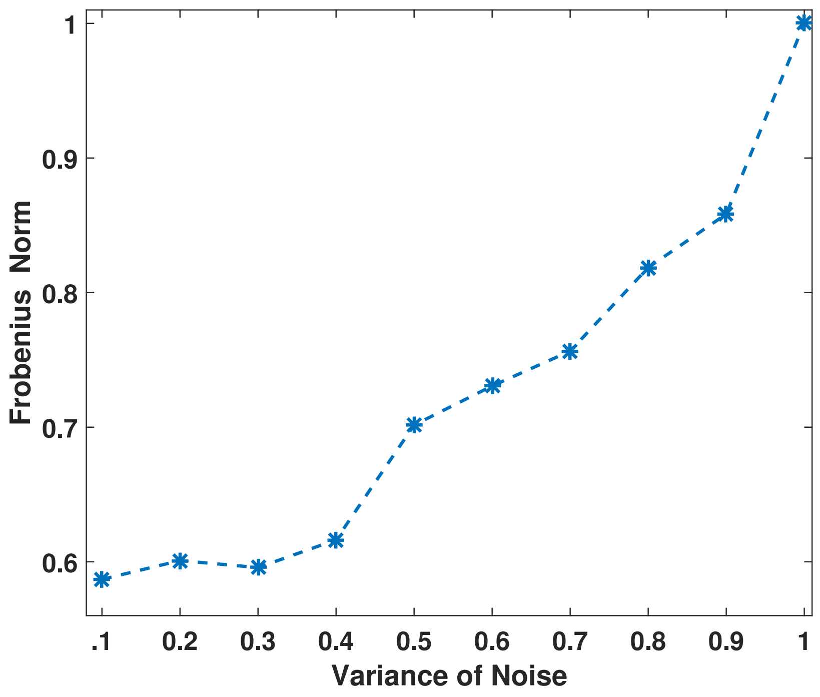

Figure 9.1: The recovery error of the proposed MC-DT algorithm, i.e. $\|\mathbf{A} - \widehat{\mathbf{A}}\|_{\mathrm{F}}$, for different noise variances.

- **MF-SS:** The third class of baseline algorithms combine the network structure with node features by sharing a subspace as formulated in Eq. (3.8). We refer to this algorithm by Matrix Factorization with Subspace Sharing (MF-SS) [32].

- **MC-DT:** Finally, this refers to the algorithm proposed in this paper which is referred to as Matrix Completion with Decoupled Transduction (MC-DT).

9.2.4 Experiments on synthetic dataset

In this section we present the results on a synthetic dataset. To do so, we generated a adjacency matrix $\mathbf{A}$ with 2,048 nodes. To generate the network, we fixed the rank to $r = 10$, i.e., the number of connected components, and evenly distributed the nodes into the components. Then we generated a pairwise similarity matrix for nodes by adding a noise term to the adjacency matrix $\mathbf{S} = \mathbf{A} + \mathbf{N}$, where each entry of noise matrix follows a uniform distribution $\mathbf{N}_{ij} \sim \mathrm{U}(0, 0.5)$. To generate an incomplete network in our dataset, we randomly removed the 20% of all links so that the network has 119,999 links (out of 2,048 nodes and 149,999 links).

Effect of noise in side information. To better understand the effect of similarity information on the recovery error, we also investigated the effect of noise in similarity matrix on the performance of MC-DT algorithm. As demonstrated in Theorem 3.3.1, the recovery error has a linear dependency on the error contributed by the error component of similarity matrix $\mathbf{A}_{\mathrm{E}}$. To empirically verify this on synthetic dataset, we added noise to similarity matrix that follows a zero mean Gaussian distribution $\mathcal{N}(0, \sigma)$ with different values of variance σ, while keeping the size of observed submatrix fixed to $m = 400$. In particular, we changed the variance from $\sigma = 0.1$ to $\sigma = 1$ with step size of 0.1 and calculated the recovery error. As Figure 9.1 shows, by increasing the noise, the recovery error increases linearly. This is consistent with our theoretical result.

Effect training size. We further investigate the size of observed submatrix $\mathbf{O}$ on recovery error of different algorithm on synthetic data. We changed the size of observed submatrix m within the range of 200 to 1600 with step-size of 200 and reported the recovery error for different algorithms in Figure 9.2. From Figure 9.2 it can be observed that by increasing m, the recovery error decreases for all of the methods. Hence, the fewer unobserved elements we have, the lower the recovery error is. The proposed MC-DT algorithm performs the best, verifying its reliable performance. The significant difference between the recovery error of MC-DT and three other algorithms implies the effectiveness of MC-DT in exploiting

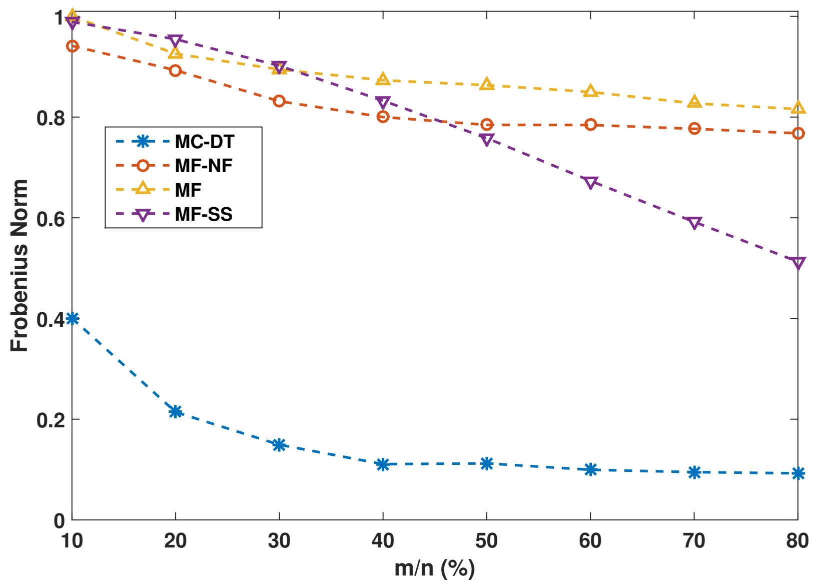

Figure 9.2: The recovery error of different algorithms on a synthetic dataset for different sizes of partially observed submatrix with m nodes.

similarity information. We note that since MC-DT ignores the missing part of the network and completes the submatrix purely from the observed part of the network, the completion error becomes zero and is not propagated in transduction stage.

It is of interest to note that by increasing the size of observed submatrix, i.e., as m approaches n, the effect of similarity information on decreasing recovery error for all methods became less influential. This follows from the fact that existence of links in the network provide much richer information than the pairwise similarity between users.

9.2.5 Performance evaluation on link prediction

In this section we demonstrate the results of applying our proposed algorithm along with different baseline algorithms on the link prediction problem. We demonstrate the results on two real-datasets, which are Epinions and Weibo.

9.2.5.1 Link prediction on Epinions

Epinions is a consumer review website, where users may read reviews about items to help them decide on their purchase. Epinions, by allowing users to have a trust/distrust relationship with each other, created a social network of users in addition to the reviewing platform. In this section we aimed to predict the existence of links while utilizing the rating information as the side information. In order to have a richer side information, we considered all available information about the items such as category, scores, timestamps.

We compared our algorithm with a number of baseline algorithms. To do this comparison, we first created a user-user binary matrix. Herein, 1 indicated a trust relation between two users and 0 indicated unobserved relations. To create the test set, if we wanted, for example, 20% of the data be considered as our test set, we randomly replaced 20% of the observed entries by 0 and kept 70% of the remaining entries unchanged as our training. The final 10% remained as our validation sets.

To compare our results with other baselines, and also explore the effect of training size, we performed our experiments on different sizes of training, validation and test sets. For the training set, we considered 40%, 50%, 60%, 70% and 80% of the data. In each case 10% of the remaining data was validation set and the rest became our test set. For each training size, we performed the experiment five times and reported the average values. Table 9.2 shows the results RMSE and MAE on Epinions with different training sizes. As it is shown, our proposed algorithm outperformed all other baseline algorithms in all training sizes. Generally, perhaps unsuprisingly, by increasing the training size, the error also reduced. However as the training size increased there is not a significant drop in the error. This allowed us to conclude that we select smaller training sizes and still maintain accuracy while also

Table 9.2: Link prediction results on Epinions dataset and the effects of training size.

RMSE

Method	Parameters	Train Percentage				
		40%	50%	60%	70%	80%
BPSO	r=20, λ_u=15, λ_i=10, T=50	0.7708	0.7683	0.7683	0.7626	0.7610
MF	r=20, λ=0.015, γ=0.01, T=50	0.7063	0.7127	0.7191	0.7173	0.7269
SO	—	0.7851	0.7780	0.7726	0.7650	0.7611
UIB	—	0.7026	0.7022	0.7032	0.6993	0.7022
CC	C_u=10,C_i=20, T=50	0.7849	0.7776	0.7712	0.7679	0.7625
LFM	r=10, β_λ=0.01, $\lambda_{u,i}$=0.05, γ=0.01, T=50	0.7212	0.7110	0.7188	0.7150	0.7122
BMF	r=10, β_λ=0.01, $\lambda_{u,i}$=0.05, T=50	0.7086	0.7084	0.7094	0.7061	0.7087
SPP	r=10, λ=0.05, β_λ=0.01, T=50	0.6897	0.6903	0.6926	0.6892	0.6929
SSPP	r=10, λ=0.05, β_λ=0.01, T=50	0.9879	0.9867	0.9851	0.9836	0.9856
SFM	r=10, λ=0.015, β_λ=0.33, γ=0.00, T=50	0.6738	0.6743	0.6770	0.6731	0.6777
MC-DT	r = 37	**0.6623**	**0.6495**	**0.6428**	**0.6430**	**0.6551**

MAE

Method	Parameters	Train Percentage				
		40%	50%	60%	70%	80%
BPSO	r=20, λ_u=15, λ_i=10, T=50	0.9404	0.9318	0.9245	0.9156	0.9078
MF	r=20, λ=0.015, γ=0.01, T=50	0.8363	0.8422	0.8474	0.8468	0.8543
SO	—	0.9515	0.9370	0.9244	0.9129	0.9047
UIB	—	0.8280	0.8278	0.8280	0.8259	0.8276
CC	C_u=10,C_i=20, T=50	0.9285	0.9160	0.9071	0.9013	0.8932
LFM	r=10, β_λ=0.01, $\lambda_{u,i}$=0.05, γ=0.01, T=50	0.8425	0.8340	0.8402	0.8377	0.8342
BMF	r=10, β_λ=0.01, $\lambda_{u,i}$=0.05, T=50	0.8318	0.8317	0.8319	0.8302	0.8315
SPP	r=10, λ=0.05, β_λ=0.01, T=50	0.8211	0.8215	0.8227	0.8206	0.8227
SSPP	r=10, λ=0.05, β_λ=0.01, T=50	1.2617	1.2605	1.2599	1.2566	1.2596
SFM	r=10, λ=0.015, β_λ=0.33, γ=0.00, T=50	0.8165	0.8167	0.8181	0.8155	0.8182
MC-DT	r = 37	**0.7625**	**0.7591**	**0.7562**	**0.7562**	**0.7573**

being able to compute solutions faster and avoid overfitting. This further assists in situations where there is not enough data available to use large training sets.

9.2.5.2 Link prediction on Weibo

Weibo is a social network for connecting users together and is similar to the Twitter. As it is explained in section 9.2.1, we selected a subset of this dataset with about 4,000 users. From these, we extracted all available features and created a similarity matrix between users. In this dataset, we aim to predict the existence of links between users while using the similarity matrix created from these features.

We performed the experiments on different training sizes similar to the Epinions. Table 9.3 shows the results of RMSE and MAE measures on Weibo dataset. The results shown in the Table 9.3 confirmed that our proposed algorithm is the best performing algorithm among the baseline algorithms. One can see that there is significant gap between the results of our proposed algorithm and other baseline algorithms. This was not the case in Epinions dataset results. This significant difference in accuracy was attributed to having more available side information available in Weibo than what was available in Epinions. We can also conclude that after having 50% as training and 10% as validation, there is not a significant drop in the results.

9.2.6 Performance evaluation on Network Completion

We now turn to comparing the proposed algorithms to the state-of-the-art algorithms for network completion. We demonstrate the results on two large real-datasets: Facebook and Google+. In this scenario, we utilized different training sizes as well. We divided the dataset into training, validation and test subsets while we divided the users into these three subsets. Because of this division, instead of by the observed links, there are no available ratings in our training subsets for the users on our test subset. For example, when the training size is 20%, we randomly selected 20% of the nodes and the corresponding links from the each social network to predict the rest of network. We evaluated the performance of MC-DT along with the baseline network completion algorithms on these two datasets.

In Figures 9.3 and 9.4, the RMSE and MAE of different algorithms on Facebook dataset are shown, respectively. As it can be observed from these plots, improvement of all methods gradually decreased as more of the network structure was observed.

Table 9.3: Link prediction results on Weibo dataset and the effects of training size.

RMSE

Method	Parameters	Train Percentage				
		40%	50%	60%	70%	80%
BPSO	r=20, λ_u=15, λ_i=10, T=50	0.5023	0.5027	0.5017	0.5041	0.5063
MF	r=20, λ=0.015, γ=0.01, T=50	0.4992	0.4999	0.4991	0.5000	0.5018
SO	—	0.5017	0.5034	0.4988	0.4999	0.5072
UIB	—	0.5007	0.5002	0.4996	0.4983	0.5000
CC	C_u=10,C_i=20, T=50	0.5020	0.5004	0.5036	0.4950	0.5022
LFM	r=10, β_λ=0.01, $\lambda_{u,i}$=0.05, γ=0.01, T=50	0.5008	0.5007	0.4997	0.4984	0.5000
BMF	r=10, β_λ=0.01, $\lambda_{u,i}$=0.05, T=50	0.5004	0.5003	0.4998	0.4986	0.5000
SPP	r=10, λ=0.05, β_λ=0.01, T=50	0.4996	0.5002	0.4998	0.4987	0.5002
SSPP	r=10, λ=0.05, β_λ=0.01, T=50	0.4991	0.5015	0.4943	0.4885	0.4884
SFM	r=10, λ=0.015, β_λ=0.33, γ=0.00, T=50	0.5001	0.5000	0.5001	0.4996	0.5000
MC-DT	r = 37	**0.3454**	**0.3285**	**0.3232**	**0.3237**	**0.3266**

MAE

Method	Parameters	Train Percentage				
		40%	50%	60%	70%	80%
BPSO	r=20, λ_u=15, λ_i=10, T=50	0.5509	0.5488	0.5444	0.5509	0.5496
MF	r=20, λ=0.015, γ=0.01, T=50	0.5012	0.5019	0.5013	0.5025	0.5053
SO	—	0.5489	0.5496	0.5414	0.5436	0.5488
UIB	—	0.5067	0.5066	0.5053	0.5038	0.5053
CC	C_u=10,C_i=20, T=50	0.5612	0.5574	0.5564	0.5511	0.5550
LFM	r=10, β_λ=0.01, $\lambda_{u,i}$=0.05, γ=0.01, T=50	0.5093	0.5100	0.5083	0.5068	0.5082
BMF	r=10, β_λ=0.01, $\lambda_{u,i}$=0.05, T=50	0.5021	0.5022	0.5017	0.5006	0.5019
SPP	r=10, λ=0.05, β_λ=0.01, T=50	0.5027	0.5036	0.5031	0.5020	0.5035
SSPP	r=10, λ=0.05, β_λ=0.01, T=50	0.6987	0.7003	0.6951	0.6908	0.6908
SFM	r=10, λ=0.015, β_λ=0.33, γ=0.00, T=50	0.5005	0.5002	0.5002	0.4997	0.5001
MC-DT	r = 37	**0.4624**	**0.4457**	**0.4419**	**0.4421**	**0.4437**

Table 9.4 shows the performance of the MC-DT in comparison with the other baseline algorithms on Google+'s social network. We noticed that the algorithms show similar behavior to the one presented in previous experiment. Specifically, MC-DT outperforms the other network completion algorithms. These results confirm the conclusions made in the previous experiments on the Facebook dataset.

We make several observations from results:

- MC-DT greatly outperformed the other baseline algorithms in all respects. Both the RMSE and MAE errors of the completed network are more than two times smaller for

MC-DT than the others. These experiments demonstrated the effectiveness of MC-DT in exploiting the similarity information to recover the full network.

- It can also be interesting to compare the MC-DT to the naive MF method which does not utilize node features. We notice that MC-DT achieves better performance, as it combines the information from the node features as well as the network structure. The significant gap between the performance of these two algorithms demonstrated the importance of similarity information in network completion problem.

- We note that when networks are incomplete, the performance of MC-DT degrades gracefully, as it is be able to rely on the node features information which compensate for the lack of links in the network structure.

- Finally, it is of interest to note that by comparing the results for synthetic and real datasets, it can be seen that in the synthetic dataset the decrease of error by increasing the number of observed nodes is significant initially, but it slowly shrinks afterwards. Compare this to real world datasets where the decrease is roughly linear.

- Considering the dependency of our theoretical upper bound on recovery error, this observation is completely consistent with our theoretical result.

  The result of Figure 9.2 shows that significant of intuitive: Even though the network contains many missing links, MC-DT still outperforms other methods significantly by better leveraging the information present in the node features.

  We note that since MC-DT ignores the missing part of the network and completes the sub-matirx purely from the observed part of the network, the completion error becomes zero and is not propagated in transduction stage.

Table 9.4: Comparison of different algorithms on Google+ with deferent sizes of observed nodes.

Method	RMSE (30%)	MAE (30%)	RMSE (60%)	MAE (60%)
MF	0.97201	0.89132	0.81749	0.76601
MF-NF	0.86384	0.82094	0.71505	0.68935
MF-SS	0.81368	0.78820	0.58369	0.68160
MC-DT	**0.78806**	**0.50186**	**0.50851**	**0.30011**

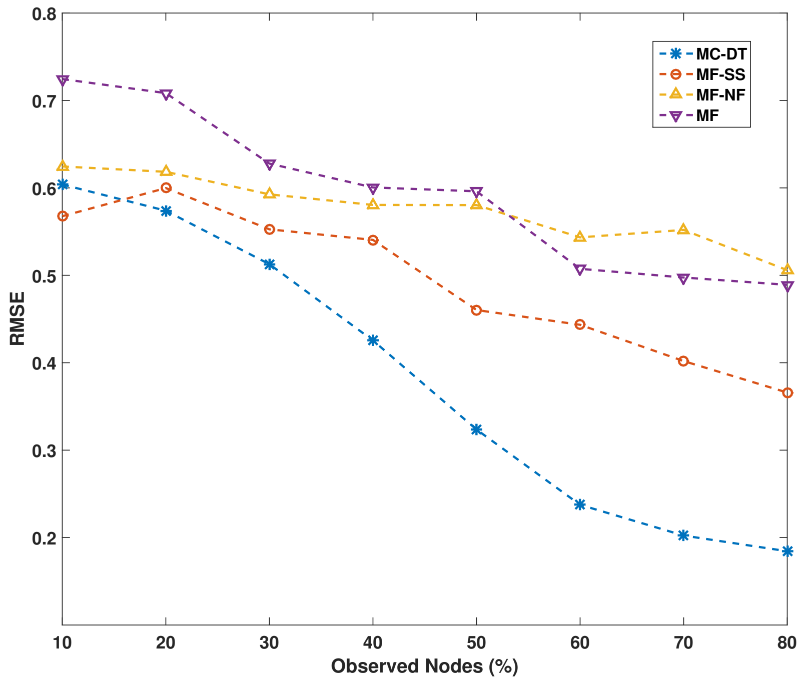

Figure 9.3: The recovery of four algorithms on Facebook dataset measured in RMSE under different percentage of observed nodes.

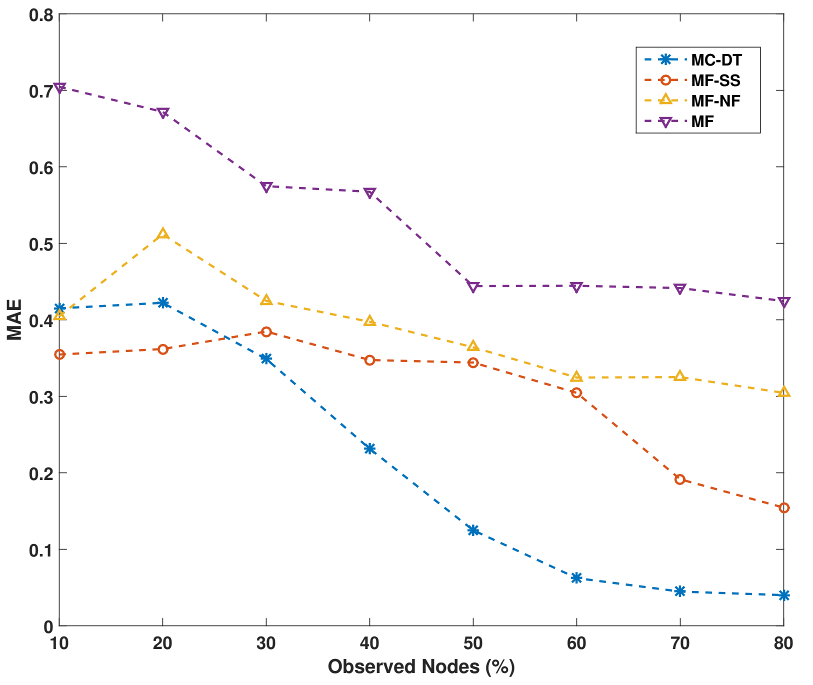

Figure 9.4: The recovery of four algorithms on Facebook dataset measured in MAE under different percentage of observed nodes.

Chapter 10

Conclusions

Due to internet information overload on the web, finding the most relevant information is becoming a challenging task. Recommender systems have played an important role in reducing the negative impacts of information overload on Web and help users to find the useful information. There are many different recommenders that all are trying to suggest the most relevant information to users. However most of them suffer from the data sparsity and its extreme case, which is cold-start.

We have proposed a novel factorization model, that explicitly exploits the similarity information about users and items to alleviate the data sparsity challenge. Two key features of our proposed framework are the completion of a sub-matrix of the rating matrix, which is generated by excluding the cold-start users and items from the set of users and items, and transduction of knowledge from recovered sub-matrix of existing users and items to those of cold-start. In particular, it decouples the completion from the knowledge transduction, which prevents the error propagation of completion and transduction. Another major application of this algorithm is for completing the networks, which throughly discussed and experimentally evaluated.

We also introduced a semi-supervised collaborative ranking model by leveraging side information about both observed and missing ratings in collaboratively learning the ranking model. In the learned model, unrated items are conservatively pushed after the relevant and before the relevant items in the ranked list of items for each individual user. This crucial difference greatly boosts the performance and limits the bias caused by learning only from sparse non-random observed ratings. The proposed algorithm is compared with many baseline algorithms on three real world datasets that demonstrated the effectiveness of proposed algorithm in addressing cold-start problem and mitigating the data sparsity problem, while being robust to sampling of missing ratings.

We have also presented a novel approach for recommendation with social information by collaboratively ranking the latent features of users in matrix factorization by exploiting the social context of users. In contrast to social regularization based methods which are able to exploit either trust or distrust relations, and exclude the neutral friends in regularization, the proposed `PushTrust` algorithm is able to simultaneously exploit the trust and distrust relationships between users and considers neutral friends in ranking. Finally, the potential of distrust as a side information to improve accuracy of recommender systems and overcome the cold-start problems in traditional recommender systems is experimentally investigated on Epinions dataset.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. *The Journal of Machine Learning Research*, 10:803–826, 2009.

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

[3] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *SIGKDD*, pages 19–28. ACM, 2009.

[4] D. Agarwal and S. Merugu. Predictive discrete latent factor models for large scale dyadic data. In *ACM SIGKDD*, pages 26–35. ACM, 2007.

[5] S. Agarwal. The infinite push: A new support vector ranking algorithm that directly optimizes accuracy at the absolute top of the list. In *SDM*, pages 839–850. SIAM, 2011.

[6] M. Aghagolzadeh, I. Barjasteh, and H. Radha. Transitivity matrix of social network graphs. In *2012 IEEE Statistical Signal Processing Workshop (SSP)*.

[7] A. Annibale and A. Coolen. What you see is not what you get: how sampling affects macroscopic features of biological networks. *Interface Focus*, 1(6):836–856, 2011.

[8] I. Barjasteh, R. Forsati, A.-H. Esfahanian, and H. Radha. Semi-supervised collaborative ranking with push at top. *arXiv preprint arXiv:1511.05266*, 2015.

[9] I. Barjasteh, R. Forsati, F. Masrour, A.-H. Esfahanian, and H. Radha. Cold-start item and user recommendation with decoupled completion and transduction. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 91–98. ACM, 2015.

[10] I. Barjasteh, R. Forsati, D. Ross, A.-H. Esfahanian, and H. Radha. Cold-start recommendation with provable guarantees: A decoupled approach. *IEEE Transactions on Knowledge and Data Engineering*, 28(6):1462–1474, 2016.

[11] I. Barjasteh, Y. Liu, and H. Radha. Trending videos: Measurement and analysis. *arXiv preprint arXiv:1409.7733*, 2014.

[12] J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *ICML*, page 9. ACM, 2004.

[13] R. M. Bell and Y. Koren. Lessons from the netflix prize challenge. *SIGKDD Explorations Newsletter*, 9(2), 2007.

[14] D. Billsus and M. J. Pazzani. User modeling for adaptive news access. *User modeling and user-adapted interaction*, 10(2-3):147–180, 2000.

[15] C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[16] M. Burke and R. Kraut. Mopping up: modeling wikipedia promotion decisions. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, pages 27–36. ACM, 2008.

[17] R. Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[18] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[19] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

[20] E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *Information Theory, IEEE Transactions on*, 56(5):2053–2080, 2010.

[21] K.-Y. Chiang, C.-J. Hsieh, N. Natarajan, I. S. Dhillon, and A. Tewari. Prediction and clustering in signed networks: a local to global perspective. *JMLR*, 15(1):1177–1213, 2014.

[22] K. Christakopoulou and A. Banerjee. Collaborative ranking with a push at the top. In *WWW*, pages 205–215. International World Wide Web Conferences Steering Committee, 2015.

[23] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR workshop on recommender systems*, volume 60. Citeseer, 1999.

[24] G. Contardo, L. Denoyer, and T. Artieres. Representation learning for cold-start recommendation. *arXiv preprint arXiv:1412.7156*, 2014.

[25] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *ACM RecSys*, pages 39–46. ACM, 2010.

[26] M. Dadashi, I. Barjasteh, and M. Jalili. Rewiring dynamical networks with prescribed degree distribution for enhancing synchronizability. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 20(4):043119, 2010.

[27] S. Das, H. Salehi, Y. Shi, S. Chakrabartty, R. Burgueno, and S. Biswas. Towards packet-less ultrasonic sensor networks for energy-harvesting structures. *Computer Communications*, 2016.

[28] R. Devooght, N. Kourtellis, and A. Mantrach. Dynamic matrix factorization with priors on unknown values. In *ACM SIGKDD*, pages 189–198. ACM, 2015.

[29] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer. The yahoo! music dataset and kdd-cup'11. In *KDD Cup*, pages 8–18, 2012.

[30] A. Elbadrawy and G. Karypis. Feature-based similarity models for top-n recommendation of new items. 2014.

[31] A. Elbadrawy and G. Karypis. User-specific feature-based similarity models for top-n recommendation of new items. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):33, 2015.

[32] Y. Fang and L. Si. Matrix co-factorization for recommendation with rich side information and implicit feedback. In *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, pages 65–69. ACM, 2011.

[33] R. Forsati, I. Barjasteh, F. Masrour, A.-H. Esfahanian, and H. Radha. Pushtrust: An efficient recommendation algorithm by leveraging trust and distrust relations. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 51–58. ACM, 2015.

[34] R. Forsati, I. Barjasteh, D. Ross, A.-H. Esfahanian, and H. Radha. Network completion by leveraging similarity of nodes. *Social Network Analysis and Mining*, 6(1):102, 2016.

[35] R. Forsati, M. Mahdavi, M. Shamsfard, and M. Sarwat. Matrix factorization with explicit trust and distrust side information for improved social recommendation. *ACM Transactions on Information Systems*, 32(4):17, 2014.

[36] O. Frank. Network sampling and model fitting. *Models and methods in social network analysis*, pages 31–56, 2005.

[37] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *ICDM*. IEEE, 2010.

[38] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. MyMediaLite: A free recommender system library. In *ACM, Recommender Systems, 2011.*

[39] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *ICDM*, 2005.

[40] A. A. Gittens. *Topics in randomized numerical linear algebra.* PhD thesis, California Institute of Technology, 2013.

[41] J. Golbeck and J. Hendler. Filmtrust: Movie recommendations using trust in web-based social networks. In *Proceedings of the IEEE Consumer communications and networking conference*, volume 96. Citeseer, 2006.

[42] A. Goldberg, B. Recht, J. Xu, R. Nowak, and X. Zhu. Transduction with matrix completion: Three birds with one stone. In *NIPS*, pages 757–765, 2010.

[43] Q. Gu and J. Zhou. Learning the shared subspace for multi-task clustering and transductive transfer classification. In *ICDM'09*, pages 159–168. IEEE, 2009.

[44] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th International Conference on World Wide Web*, pages 403–412. ACM, 2004.

[45] R. Guimerà and M. Sales-Pardo. Missing and spurious interactions and the reconstruction of complex networks. *Proceedings of the National Academy of Sciences*, 106(52):22073–22078, 2009.

[46] A. Gunawardana and C. Meek. A unified approach to building hybrid recommender systems. In *Proceedings of the third ACM, Recommender systems*, 2009.

[47] S. K. Gupta, D. Phung, B. Adams, T. Tran, and S. Venkatesh. Nonnegative shared subspace learning and its application to social media retrieval. In *ACM SIGKDD*, pages 1169–1178. ACM, 2010.

[48] S. K. Gupta, D. Phung, B. Adams, and S. Venkatesh. Regularized nonnegative shared subspace learning. *Data mining and knowledge discovery*, 26(1):57–97, 2013.

[49] S. Hanneke and E. P. Xing. Network completion and survey sampling. In *AISTAT*, pages 209–215, 2009.

[50] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM TOIS*, 22(1):5–53, 2004.

[51] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 135–142. ACM, 2010.

[52] M. Jamali and M. Ester. A transitivity aware matrix factorization model for recommendation in social networks. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pages 2644–2649. AAAI Press, 2011.

[53] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[54] K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *ACM SIGIR*, pages 41–48. ACM, 2000.

[55] M. Kim and J. Leskovec. The network completion problem: Inferring missing nodes and edges in networks. In *SDM*, pages 47–58. SIAM, 2011.

[56] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD'08*, pages 426–34.

[57] Y. Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(1):1, 2010.

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

[59] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer. Local collaborative ranking. In *Proceedings of the 23rd international conference on World wide web*, pages 85–96. ACM, 2014.

[60] D. Lemire and A. Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *SDM*, volume 5, pages 1–5. SIAM, 2005.

[61] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[62] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4):2065–2073, 2014.

[63] J. Lin, K. Sugiyama, M.-Y. Kan, and T.-S. Chua. Addressing cold-start in app recommendation: latent user models constructed from twitter followers. In *ACM SIGIR*, pages 283–292. ACM, 2013.

[64] G. Ling, M. R. Lyu, and I. King. Ratings meet reviews, a combined approach to recommend. In *ACM Conference on Recommender Systems*, pages 105–112. ACM, 2014.

[65] J. Liu, C. Wu, and W. Liu. Bayesian probabilistic matrix factorization with social relations and item contents for recommendation. *Decision Support Systems*, 2013.

[66] N. N. Liu, X. Meng, C. Liu, and Q. Yang. Wisdom of the better few: cold start recommendation via representative based rating elicitation. In *ACM Conference on Recommender Systems*, pages 37–44. ACM, 2011.

[67] W. Liu, J. Wang, and S.-F. Chang. Robust and scalable graph-based semisupervised learning. *Proceedings of the IEEE*, 100(9):2624–2638, 2012.

[68] M. Long, J. Wang, G. Ding, W. Cheng, X. Zhang, and W. Wang. Dual transfer learning. In *SDM*, pages 540–551. SIAM, 2012.

[69] H. Ma, I. King, and M. R. Lyu. Learning to recommend with social trust ensemble. In *ACM SIGIR*, pages 203–210. ACM, 2009.

[70] H. Ma, M. R. Lyu, and I. King. Learning to recommend with trust and distrust relationships. In *Proceedings of the third ACM conference on Recommender systems*, pages 189–196. ACM, 2009.

[71] H. Ma, H. Yang, M. R. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 931–940. ACM, 2008.

[72] 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*, pages 287–296. ACM, 2011.

[73] F. Masrour, I. Barjasteh, R. Forsati, A.-H. Esfahanian, and R. Hayder. Network completion with node similarity: a matrix completion approach with provable guarantees. In *ASONAM*, 2015.

[74] P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. In *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, pages 492–508. Springer, 2004.

[75] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, pages 415–444, 2001.

[76] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *AAAI/IAAI*, pages 187–192, 2002.

[77] A. K. Menon, K.-P. Chitrapura, S. Garg, D. Agarwal, and N. Kota. Response prediction using collaborative filtering with hierarchies and side-information. In *ACM SIGKDD*, pages 141–149. ACM, 2011.

[78] A. K. Menon and C. Elkan. A log-linear model with latent features for dyadic prediction. In *ICDM*, pages 364–373. IEEE, 2010.

[79] A. K. Menon and C. Elkan. Link prediction via matrix factorization. In *Machine Learning and Knowledge Discovery in Databases*, pages 437–452. Springer, 2011.

[80] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264, 2007.

[81] J. S. Morgan, I. Barjasteh, C. Lampe, and H. Radha. The entropy of attention and popularity in youtube videos. *arXiv preprint arXiv:1412.1185*, 2014.

[82] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[83] Y. Nesterov. *Introductory lectures on convex optimization*, volume 87. Springer Science & Business Media, 2004.

[84] Y. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[85] W. Pan, E. W. Xiang, N. N. Liu, and Q. Yang. Transfer learning in collaborative filtering for sparsity reduction. In *AAAI*, volume 10, pages 230–235, 2010.

[86] M. Papagelis, G. Das, and N. Koudas. Sampling online social networks. *Knowledge and Data Engineering, IEEE Transactions on*, 25(3):662–676, 2013.

[87] S.-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In *ACM Conference on Recommender Systems*, pages 21–28. ACM, 2009.

[88] S.-T. Park, D. Pennock, O. Madani, N. Good, and D. DeCoste. Naïve filterbots for robust cold-start recommendations. pages 699–705, 2006.

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

[90] M. J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999.

[91] A. Popescul, D. M. Pennock, and S. Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *UAI*, pages 437–444, 2001.

[92] I. Porteous, A. U. Asuncion, and M. Welling. Bayesian matrix factorization with side information and dirichlet process mixtures. In *AAAI*, 2010.

[93] B. Recht. A simpler approach to matrix completion. *The Journal of Machine Learning Research*, 12:3413–3430, 2011.

[94] S. Rendle. Factorization machines. In *ICDM*, pages 995–1000. IEEE, 2010.

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

[96] 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*, pages 251–258. ACM, 2008.

[97] J. D. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *ICML*, pages 713–719. ACM, 2005.

[98] S. Roweis. Nips dataset (2002). *http://www. cs. nyu. edu/˜ roweis.*

[99] C. Rudin. The p-norm push: A simple convex ranking algorithm that concentrates at the top of the list. *The Journal of Machine Learning Research*, 10:2233–2271, 2009.

[100] L. Safoury and A. Salah. Exploiting user demographic attributes for solving cold-start problem in recommender system. *Lecture Notes on Software Engineering*, 1(3):303–307, 2013.

[101] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295. ACM, 2001.

[102] M. Saveski and A. Mantrach. Item cold-start recommendations: learning local collective embeddings. In *ACM Conference on Recommender Systems*, pages 89–96. ACM, 2014.

[103] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *ACM SIGIR*, pages 253–260. ACM, 2002.

[104] H. Shan and A. Banerjee. Generalized probabilistic matrix factorizations for collaborative filtering. In *ICDM*, pages 1025–1030. IEEE, 2010.

[105] M. Sharma, J. Zhou, J. Hu, and G. Karypis. Feature-based factorized bilinear similarity model for cold-start top-n item recommendation. In *SDM*, 2015.

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

[107] M. Shiga, I. Takigawa, and H. Mamitsuka. Annotating gene function by combining expression data with a modular gene network. *Bioinformatics*, 23(13):i468–i478, 2007.

[108] R. R. Sinha and K. Swearingen. Comparing recommendations made by online systems and friends.

[109] L. H. Son. Dealing with the new user cold-start problem in recommender systems: A comparative review. *Information Systems*, 2014.

[110] N. Srebro, J. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. pages 1329–1336. NIPS, 2004.

[111] H. Steck. Training and testing of recommender systems on data missing not at random. In *KDD*, pages 713–722. ACM, 2010.

[112] H. Steck. Gaussian ranking by matrix factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 115–122. ACM, 2015.

[113] M. Sun, F. Li, J. Lee, K. Zhou, G. Lebanon, and H. Zha. Learning multiple-question decision trees for cold-start recommendation. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 445–454. ACM, 2013.

[114] M. Trevisiol, L. M. Aiello, R. Schifanella, and A. Jaimes. Cold-start news recommendation with domain-dependent browse graph. In *ACM Conference on Recommender Systems*, volume 14, 2014.

[115] N. Verbiest, C. Cornelis, P. Victor, and E. Herrera-Viedma. Trust and distrust aggregation enhanced with path length incorporation. *Fuzzy Sets and Systems*, 202:61–74, 2012.

[116] P. Victor, C. Cornelis, M. D. Cock, and A. Teredesai. Trust- and distrust-based recommendations for controversial reviews. *IEEE Intelligent Systems*, 26(1):48–55, 2011.

[117] P. Victor, C. Cornelis, and M. De Cock. *Trust networks for recommender systems*, volume 4. Springer, 2011.

[118] P. Victor, N. Verbiest, C. Cornelis, and M. D. Cock. Enhancing the trust-based recommendation process with explicit distrust. *ACM Transactions on the Web (TWEB)*, 7(2):6, 2013.

[119] M. Volkovs and R. S. Zemel. Collaborative ranking with 17 parameters. In *Advances in Neural Information Processing Systems*, pages 2294–2302, 2012.

[120] D. Zhang, Q. Zou, and H. Xiong. Cruc: Cold-start recommendations using collaborative filtering in internet of things. *arXiv preprint arXiv:1306.0165*, 2013.

[121] K. Zhou, S.-H. Yang, and H. Zha. Functional matrix factorizations for cold-start recommendation. In *ACM SIGIR*, pages 315–324. ACM, 2011.

[122] T. Zhou, H. Shan, A. Banerjee, and G. Sapiro. Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *SDM*, volume 12, pages 403–414. SIAM, 2012.