Learning to Recommend

MA, Hao

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
December 2009

Thesis/Assessment Committee Members

Professor Jeffrey Xu YU (Chair)
Professor Irwin KING (Thesis Supervisor)
Professor Michael R. LYU (Thesis Supervisor)
Professor Yufei TAO (Committee Member)
Professor Qiang YANG (External Examiner)

Abstract of thesis entitled:
Learning to Recommend
Submitted by MA, Hao
for the degree of Doctor of Philosophy
at The Chinese University of Hong Kong in December 2009

Recommender Systems are becoming increasingly indispensable nowadays since they focus on solving the information overload problem, by providing users with more proactive and personalized information services. Typically, recommender systems are based on *Collaborative Filtering*, which is a technique that automatically predicts the interest of an active user by collecting rating information from other similar users or items. Due to their potential commercial values and the associated great research challenges, Recommender systems have been extensively studied by both academia and industry recently.

However, the data sparsity problem of the involved user-item matrix seriously affects the recommendation quality. Many existing approaches to recommender systems cannot easily deal with users who have made very few ratings. The objective of this thesis is to study how to build effective and efficient approaches to improve the recommendation performance.

In this thesis, we first propose two collaborative filtering methods which only utilize the user-item matrix for recommendations. The first method is a neighborhood-based collaborative filtering method which designs an effective missing data prediction algorithm to improve recommendation quality, while the second one is a model-based collaborative filtering method

which employs matrix factorization technique to make the recommendation more accurate.

In view of the exponential growth of information generated by online users, social contextual information analysis is becoming important for many Web applications. Hence, based on the assumption that users can be easily influenced by the friends they trust and prefer their friends' recommendations, we propose two recommendation algorithms by incorporating users' social trust information. These two methods are based on probabilistic matrix factorization. The complexity analysis indicates that our approaches can be applied to very large datasets since they scale linearly with the number of observations, while the experimental results show that our methods perform better than the state-of-the-art approaches.

As one of the social relations, "distrust" also performs an important role in online Web sites. We also observe that distrust information can also be incorporated to improve recommendation quality. Hence, the last part of this thesis studies the problem on how to improve recommender system by considering explicit distrust information among users. We make the assumption that users' distrust relations can be interpreted as the "dissimilar" relations since user u_i distrusts user u_d indicates that user u_i disagrees with most of the opinions issued by user u_d. Based on this intuition, the distrust relations between users can be easily modeled by adding the regularization term into the objective functions of the user-item matrix factorization. The experiments on the Epinions dataset indicate that distrust information is at least as important as trust information.

論文題目： 學習推薦

作者 ： 馬好

學校 ： 香港中文大學

學系 ： 計算器科學及工程學系

修讀學位： 哲學博士

摘要 ：

因為能夠提供給在綫用戶個性化的信息服務，並且能夠解決互聯網資訊過載的問題，推薦系統已經越來越收歡迎，並且必不可少。典型的推薦系統是基於協同過濾的。這種技術能夠根據其他相似用戶的評分記錄來自動預測當前用戶的興趣。由於其重要的商業價值和研究價值，推薦系統已經被工業界和學術界進行了深入的研究。

然而，用戶-項目矩陣的數據稀疏問題嚴重影響了推薦質量。很多的推薦算法都不能準確的給評分很少的用戶評分。因此，本論文的目標是研究怎樣設計更加有效的推薦方法來改善推薦的性能。

本論文首先提出了兩種基於用戶-項目矩陣的協同過濾方法。第一種方法屬於基於鄰居的協同過濾方法，這個方法設計了一個有效的預測缺失數據的算法來提高推薦質量；第二種方法是基於模型的協同過濾的方法，這種方法應用了矩陣分解技術使得推薦更為準確。

隨著因特網上用戶產生的數據的大量增加，社交相關的信息分析在很多在綫應用中被廣泛研究。因此，基於用戶很容易就能被在綫好友影響的猜想，本論文

接著提出了基於用戶社交“信任”信息的兩種新穎的推薦算法。這兩種方法提出了概率矩陣分解框架來進行推薦。複雜度分心顯示這兩種方法能夠應用到十分大的數據集上，同時，實驗結果證明我們提出的方法比其他的先進算法更加優秀。

作為一種社會關係，“不信任”關係在互聯網上也是十分重要的。我們同時也觀察到“不信任”的信息也能用於改善推薦系統。因此，本論文的最後一個部份提出了一種算法使用用戶的“不信任”信息來提高推薦質量。這種算法是基於一下的假設：用戶u_i不信任用戶u_d表明了用戶u_i不同意用戶u_d的大多數觀點。基於這個直覺，用戶之間不信任的關係可以被建模為矩陣分解目標方程的正則項。使用 Epinions 數據集的實驗結果顯示在推薦系統中，“不信任”關係和“信任”關係一樣重要。

Acknowledgement

I would like to express my sincere gratitude and appreciation to my supervisors, Prof. Irwin King and Prof. Michael R. Lyu. I gain too much from their guidance not only on knowledge and attitude in doing research, but also on the presentation, teaching, and English writing skills. I will always be grateful for their supervision, encouragement and support at all levels.

I am grateful to my thesis committee members, Prof. Jeffrey Xu Yu and Prof. Yufei Tao for their helpful comments and suggestions about this thesis. My special thanks to Prof. Qiang Yang who kindly served as the external committee for this thesis. I would like to thank my mentors, Dr. Raman Chandrasekar, Dr. Chao Liu and Dr. Yi-Min Wang, for their guidance, support, insightful opinions and valuable suggestions when I was visiting Microsoft Research Redmond as research intern.

I thank Haixuan Yang, Jianke Zhu, Zibin Zheng, Tom Chao Zhou and Shikui Tu for their effort and constructive discussions in conducting the research work in this thesis. I also thank my colleagues in the machine learning and web intelligence group, Kaizhu Huang, Zenglin Xu, Allen Lin, Haiqin Yang, Hongbo Deng, Xin Xin. I also appreciate the help from my office-mates and friends, Xinyu Chen, Xiaoqi Li, Yangfan Zhou, Wujie Zheng, Junjie Xiong, Tu Zhou, Yingni She and many others.

Last but not least, I want to thank my wife, my sister, my brother-in-law and my parents. Without their deep love and constant support, this thesis would never have been completed.

To my lovely wife and my beloved parents.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

As the exponential growth of information generated on the World Wide Web, the *Information Filtering* techniques like *Recommender Systems* have become more and more important and popular. Recommender systems form a specific type of information filtering technique that attempts to suggest information items (movies, books, music, news, Web pages, images, etc.) that are likely to interest the users. Typically, recommender systems are based on *Collaborative Filtering*, which is a technique that automatically predicts the interest of an active user by collecting rating information from other similar users or items. The underlying assumption of collaborative filtering is that the active user will prefer those items which other similar users prefer [68]. Based on this simple but effective intuition, collaborative filtering has been widely employed in some large, well-known commercial systems, including product recommendation at Amazon[1], movie recommendation at Netflix[2], etc.

Due to the potential commercial values and the great research challenges, recommendation techniques have drawn much attention in data mining [9, 59], information retrieval [7, 26, 45, 50,

[1]http://www.amazon.com
[2]http://www.netflix.com

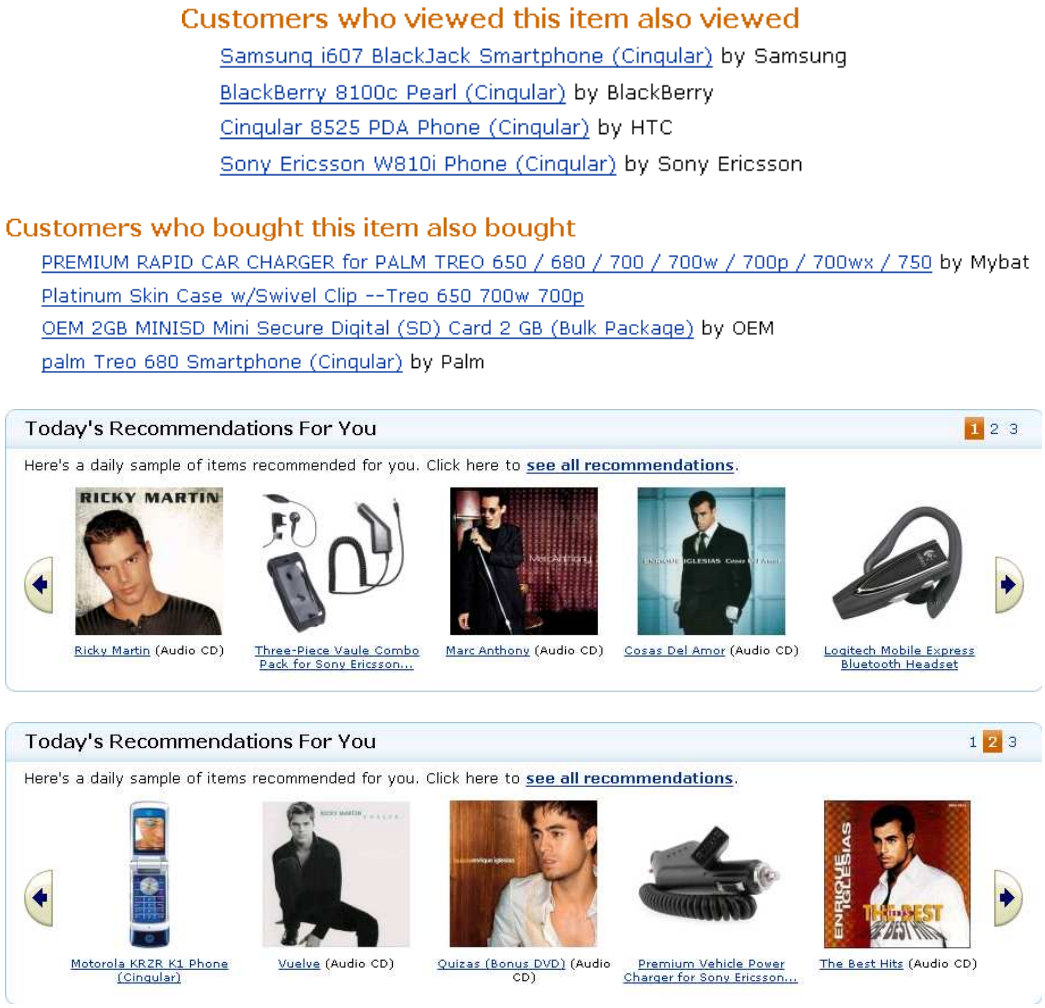


Figure 1.1: Recommendations from Amazon

58, 131] and machine learning [75, 94, 98, 99, 100, 133] communities. Recommendation algorithms suggesting personalized recommendations greatly increase the likelihood of customers making their purchases online. Fig. 1.1 shows some recommendation examples from Amazon.

A number of algorithms have been proposed to improve both the recommendation quality and the scalability problems. These collaborative filtering algorithms can be divided into two main categories: neighborhood-based (or memory-based) and model-

based approaches [16, 102]. Different methods make different assumptions. The neighborhood-based recommendation algorithms assume that those who agreed in the past tend to agree again in the future. They usually fall into two classes: user-based approaches [16, 44] and item-based approaches [28, 102]. To predict a rating for an item from a user, user-based methods find other similar users and leverage their ratings to the item for prediction, while item-based methods use the ratings to other similar items from the user instead [21]. In addition to the neighborhood-based approach, the model-based approaches employ the observed user-item ratings to train a predefined model. Algorithms in this category include clustering methods [124], Bayesian model [128], aspect model [48], etc.

Although recommendation algorithms have been widely used in recommendation systems [65, 95], the problem of inaccurate recommendation results still exists in both neighborhood-based methods and model-based methods. The fundamental problem of these approaches is the data sparsity of the user-item matrix. The density of available ratings in commercial recommender systems is often less than 1% [102] or even much less. In such circumstance, neighborhood-based [53, 65, 68, 119] collaborative filtering algorithms fail to find similar users, since the methods of computing similarities, such as the Pearson Correlation Coefficient (PCC) or the Cosine method, assume that two users have rated at least some items in common. Moreover, almost all of model-based [47, 48, 99, 107] collaborative filtering algorithms cannot handle users who rated only a few items.

Many recent algorithms have been proposed to alleviate the data sparsity problem. In [119], Wang et al. proposed a generative probabilistic framework to exploit more of the data available in the user-item matrix by fusing all ratings with a predictive value for a recommendation to be made. Xue et al. [124] proposed a framework for collaborative filtering which combines

the strengths of memory-based approaches and model-based approaches by introducing a smoothing-based method, and solved the data sparsity problem by predicting all the missing data in a user-item matrix. Although the simulation showed that this approach can achieve better performance than other collaborative filtering algorithms, the cluster-based smoothing algorithm limited the diversity of users in each cluster and predicting all the missing data in the user-item matrix could bring negative influence for the recommendation of active users.

Based on the above analysis, in order to improve the recommendation quality, we need to solve the data sparsity problem. In this thesis, we propose five effective and efficient methods to make the recommendations more accurate.

The first two algorithms purely based on user-item matrix, and do not include any other data sources. The first algorithm [68] is a memory-based collaborative filtering algorithm which focuses on recommending products or items based on the past behavior of similar users. Notable similarity computation algorithms include Pearson Correlation Coefficient (PCC) [95] and Vector Space Similarity (VSS) algorithm [17]. PCC-based collaborative filtering generally can achieve higher performance than the other popular algorithm VSS, since it considers the differences of user rating styles. In order to generate more accurate recommendations, Amazon also extended this method to calculate the implicit relations between items or products, which is called item-based method. Item-based methods share the same idea with user-based methods. The only difference is user-based methods try to find the similar users for an active user but item-based methods try to find the similar items for each item. The second algorithm [72] is a model-based collaborative filtering algorithm which employs semi-nonnegative matrix factorization techniques to improve recommendation quality.

Different with the first two algorithms, the rest three ap-

proaches [69, 70, 74] incorporates social relations between users. These relations are normally assigned by online users explicitly. Actually, thanks to the popularity of the Web 2.0 applications, recommender systems are now associated with various kinds of social context information, including users' social trust network, social distrust network, tags issued by users or associated with items, etc. This contextual information contains abundant additional information about the interests of users or properties of items, hence providing a huge opportunity to improve the recommendation quality. For example, in users' social trust network, users tend to share their similar interests with the friends they trust. In reality, we always turn to friends we trust for movie, music or book recommendations, and our tastes and characters can be easily affected by the company we keep.

Traditional recommender systems assume that users are independent and identically distributed. This assumption ignores the social trust relationships among the users. But the fact is, offline, social recommendation is an everyday occurrence. For example, when you ask a trusted friend for a recommendation of a movie to watch or a good restaurant to dine, you are essentially soliciting a verbal social recommendation. In [110], Sinha et al. have demonstrated that, given a choice between recommendations from trusted friends and those from recommender systems, in terms of quality and usefulness, trusted friends' recommendations are preferred, even though the recommendations given by the recommender systems have a high novelty factor. Trusted friends are seen as more qualified to make good and useful recommendations compared to traditional recommender systems [8]. From this point of view, the traditional recommender systems that ignore the social network structure of the users may no longer be suitable.

1.2 Thesis Contributions

The main contributions of this thesis can be described as follows:

(1) **Effective Missing Data Prediction**

In order to extract implicit social relations between users, we first use PCC-based significance weighting to compute similarities between users, which overcomes the potential decrease of similarity accuracy. We also extend this method to calculate similarities between items. Second, we propose an effective missing data prediction algorithm which exploits the information both from users and items. Moreover, this algorithm will predict the missing data of a user-item matrix if and only if we think it will bring positive influence for the recommendation of active users instead of predicting every missing data of the user-item matrix. The simulation shows our novel approach achieves better performance than other state-of-the-art collaborative filtering approaches.

(2) **Recommend with Global Consistency** We propose a semi-nonnegative matrix factorization method with global statistical consistency. The major contribution of our work is twofold: (1) We endow a new understanding on the generation or latent compositions of the user-item rating matrix. Under the new interpretation, our work can be formulated as the semi-nonnegative matrix factorization problem. (2) Moreover, we propose a novel method of imposing the consistency between the statistics given by the predicted values and the statistics given by the data. We further develop an optimization algorithm to determine the model complexity automatically. The complexity of our method is linear with the number of the observed ratings, hence it is scalable to very large datasets.

(3) **Social Recommendation**

We propose a framework to integrate social contextual information and the user-item rating matrix, based on a probabilistic factor analysis. We connect these different data resources through the shared user latent feature space (or item latent feature space), that is, the user latent feature space in the social contextual information is the same as in the user-item rating matrix. By performing factor analysis based on probabilistic matrix factorization, the low-rank user latent feature space and item latent feature space are learned in order to make recommendations. The experimental results on the Epinions[3] and Movielens[4] datasets show that our method outperforms the state-of-the-art collaborative filtering algorithms, especially when active users have very few ratings. Moreover, the complexity analysis indicates that our approach can be applied to very large datasets since it scales linearly with the number of observations.

(4) **Recommend with Social Trust Ensemble**

Aiming at modeling the recommender systems more accurately and realistically, we endow a novel understanding to all the ratings in the user-item matrix R. We interpret the rating R_{ij} in the user-item matrix as the representation mixed by both the user u_i's taste and his/her trusted friends tastes on the item v_j. This assumption naturally employs both the user-item matrix and the users' social trust network for the recommendations.

In terms of the users' own tastes, we factorize the user-item matrix and learn two low-dimensional matrices, which are user-specific latent matrix and item-specific latent matrix.

[3]http://www.epinions.com
[4]http://www.grouplens.org/node/73

For the social trust graph, based on the intuition that users always prefer the items recommended by the friends they trust, we infer and formulate the recommendation problem purely based on their trusted friends' favors. Then, by employing a probabilistic framework, we fuse the users and their trusted friends' tastes together by an ensemble parameter. Finally, by performing a simple gradient descent on the objective function, we learn the latent low-dimensional user-specific and item-specific matrices for the prediction of users' favors on different items.

(5) **Recommend with Social Distrust**

We elaborate how user distrust information can benefit the recommender systems. Users' distrust relations can be interpreted as the "dissimilar" relations since user u_i distrusts user u_d indicates that user u_i disagrees with most of the opinions issued by user u_d. Different with distrust, users' trust relations can be modeled as the "similar" relations due to the reason that user u_i trusts user u_t means that user u_i agrees with most of the opinions issued by u_t. Based on the above intuitions, the distrust and trust relations between users can be easily modeled by adding the regularization terms into the objective functions of the user-item matrix factorization. By performing a simple gradient descent on the objective function, we can learn the latent low-dimensional user-specific and item-specific matrices for the prediction of users' favors on different items.

1.3 Thesis Organization

The rest of this thesis is organized as follows:

- Chapter 2

In this chapter, we briefly review some background knowledge and related work in the field of recommender systems.

- Chapter 3

  This chapter focuses the neighborhood-based collaborative filtering problems on two crucial factors: (1) similarity computation between users or items and (2) missing data prediction algorithms. First, we use the enhanced Pearson Correlation Coefficient (PCC) algorithm by adding one parameter which overcomes the potential decrease of accuracy when computing the similarity of users (implicit social relations) or items. Second, we propose an effective missing data prediction algorithm, in which information of both users and items is taken into account. In this algorithm, we set the similarity threshold for users and items respectively, and the prediction algorithm will determine whether predicting the missing data or not. We also address how to predict the missing data by employing a combination of user and item information. Finally, empirical studies on dataset MovieLens have shown that our newly proposed method is more robust against data sparsity.

- Chapter 4

  In this chapter, we propose a Semi-Nonnegative Matrix Factorization with Global Statistical Consistency (SNGSC) approach for collaborative filtering. First, we endow a new understanding on the latent compositions of the ratings, which is based on the following assumptions: (1) there are totally a number of d types of items; (2) on each type of items, every user has a confidence value indicating the taste of this user on the type; (3) each item also has a quality value on each type. Based on these assumptions, we formulate the collaborative filtering algorithm as the Semi-Nonnegative Matrix Factorization problem, and propose an

optimization formulation with sensitive analysis. Second, based on the observation that the statistics of the predicted ratings are not consistent with the statistics of the training data, we propose to impose the consistency between them. This consideration generates very good performance when the dataset is spare. Furthermore, we develop an algorithm to determine the model complexity automatically. The complexity of our method is linear with the number of the observed ratings, which can be applied to very large datasets.

- Chapter 5

  In this chapter, based on the assumption that users' decisions can be easily influenced by the friends they trust, we propose a factor analysis approach based on probabilistic matrix factorization to alleviate the data sparsity and poor prediction accuracy problems by incorporating social trust information. This method is quite general, and we also can extend this approach to improve recommender systems with social tags that are issued by users.

- Chapter 6

  Although the users' social trust network is integrated into the recommender systems by factorizing the social trust graph in Chapter 4, the real world recommendation processes are not reflected in the model. This drawback not only causes lack of interpretability in the model, but also affects the recommendation qualities. A more novel and realistic approach is needed to model the trust-aware recommendation problem. In this chapter, aiming at modeling recommender systems more accurately and realistically, we propose a novel probabilistic factor analysis framework, which naturally fuses the users' tastes and their trusted friends' favors together. In this framework, we coin the

term *Social Trust Ensemble* to represent the formulation of the social trust restrictions on the recommender systems.

- Chapter 7

  In this chapter, we prove that not only social trust information can be used to improve recommender systems, social distrust information is also a very important source. We model users' distrust relations as the "dissimilar" relations based on the intuition that distrust can be interpreted as disagree in most circumstances. We also extend this idea to model trust relations as the "similar" relations. The experimental results show that the distrust relations among users are as important as the trust relations.

- Chapter 8

  The last chapter summarizes this thesis and addresses some future directions that can be further explored.

In order to make each of these chapters self-contained, some critical contents, e.g., model definitions or motivations having appeared in previous chapters, may be briefly reiterated in some chapters.

□ **End of chapter.**

Chapter 2

Background Review

Generally speaking, the concept of Recommendation is very broad. Lots of research problems can be classified as recommendation problems, including search ranking [18, 19, 22, 67, 92], query suggestion [5, 25, 32, 34, 37, 52, 71, 123], tag recommendation [41, 108, 112, 132], Web service recommendation [129], marketing candidates selection [29, 30, 73, 97, 111], question answering [52], etc. However, in this thesis, we only focus on Recommender Systems, which form a specific type of information filtering technique that attempts to present information items (movies, music, books, news, images, web pages, etc.) that are likely of interest to the user.

Recommender systems have become an important research area since the appearance of the first papers on collaborative filtering in the mid-1990s [46, 95, 106]. They are becoming increasingly indispensable nowadays since they focus on solving the information overload problem by providing users with more proactive and personalized information services. In this chapter, we briefly review some backgrounds about recommender systems, including (1) traditional recommender systems which are mainly based on collaborative filtering techniques, and (2) social-based recommender systems which have drawn lots of attention recently.

2.1 Traditional Recommender Systems

As reported in [2], although the roots of recommender systems can be traced back to the extensive work in cognitive science [96], approximation theory [91], information retrieval [101], forecasting theories [4], and also have links to management science [80] and to consumer choice modeling in marketing [64], recommender systems emerged as an independent research area in the mid-1990s when researchers started focusing on recommendation problems that explicitly rely on the ratings structure. The missing ratings of the not-yet-rated items can be estimated in many different ways using methods from machine learning, approximation theory, and various heuristics [2].

Normally, recommender systems can be classified into two categories:

- Content-based filtering: The user will be recommended items similar to the ones the user preferred in the past;
- Collaborative filtering: The user will be recommended items that people with similar tastes and preferences liked in the past.

In this chapter, we mainly focus on collaborative filtering since this method is the most popular and effective method, which is widely analyzed in both industry and academia. According to [17], algorithms for collaborative filtering can be grouped into two general classes: memory-based (or neighborhood-based) and model-based.

2.1.1 Memory-based Methods

The memory-based approaches [17, 27, 39, 44, 53, 57, 65, 81, 95, 102] are the most popular prediction methods and are widely adopted in commercial collaborative filtering systems [65, 95].

The most analyzed examples of memory-based collaborative filtering include user-based approaches [17, 44, 53, 124] and item-based approaches [28, 65, 102].

[44] presented an algorithmic framework for performing collaborative filtering and new algorithmic elements that increase the accuracy of collaborative prediction algorithms. This work also presented a set of recommendations on selection of the right collaborative filtering algorithmic components. [65] reviewed the famous Amazon item-to-item collaborative filtering method.

User-based approaches predict the ratings of active users based on the ratings of similar users found, and item-based approaches predict the ratings of active users based on the information of similar items computed. User-based and item-based approaches often use PCC (Pearson Correlation Coefficient) algorithm [95] and VSS (Vector Space Similarity) algorithm [17] as the similarity computation methods. PCC-based collaborative filtering generally can achieve higher performance than the other popular algorithm VSS, since it considers the differences of user rating styles.

Given a recommendation system consists of M users and N items, the relationship between users and items is denoted by an $M \times N$ matrix, called the user-item matrix. Every entry in this matrix $r_{m,n}$ represents the score value, r, that user m rates an item n, where $r \in \{1, 2, ..., r_{max}\}$. If user m does not rate the item n, then $r_{m,n} = 0$.

User-based collaborative filtering engaging PCC was used in a number of recommendation systems [106], since it can be easily implemented and can achieve high accuracy when comparing with other similarity computation methods. In user-based collaborative filtering, PCC is employed to define the similarity between two users a and u based on the items they rated in common:

$$Sim(a,u) = \frac{\sum_{i \in I(a) \cap I(u)} (r_{a,i} - \overline{r}_a) \cdot (r_{u,i} - \overline{r}_u)}{\sqrt{\sum_{i \in I(a) \cap I(u)} (r_{a,i} - \overline{r}_a)^2} \cdot \sqrt{\sum_{i \in I(a) \cap I(u)} (r_{u,i} - \overline{r}_u)^2}}, \tag{2.1}$$

where $Sim(a,u)$ denotes the similarity between user a and user u, and i belongs to the subset of items which user a and user u both rated. $r_{a,i}$ is the rate user a gave item i, and $\overline{r}_a$ represents the average rate of user a. From this definition, user similarity $Sim(a,u)$ is ranging from $[-1,1]$, and a larger value means users a and u are more similar.

Item-based methods such as [28, 102] are similar to user-based approaches, and the difference is that item-based methods employ the similarity between the items instead of users. The basic idea in similarity computation between two items i and j is to first isolate the users who have rated both of these items and then apply a similarity computation technique to determine the similarity $Sim(i,j)$ [102]. The PCC-based similarity computation between two items i and j can be described as:

$$Sim(i,j) = \frac{\sum_{u \in U(i) \cap U(j)} (r_{u,i} - \overline{r}_i) \cdot (r_{u,j} - \overline{r}_j)}{\sqrt{\sum_{u \in U(i) \cap U(j)} (r_{u,i} - \overline{r}_i)^2} \cdot \sqrt{\sum_{u \in U(i) \cap U(j)} (r_{u,j} - \overline{r}_j)^2}}, \tag{2.2}$$

where $Sim(i,j)$ is the similarity between item i and item j, and u belongs to the subset of users who both rated item i and item j. $r_{u,i}$ is the rate user u gave item i, and $\overline{r}_i$ represents the average rate of item i. Like user similarity, item similarity $Sim(i,j)$ is also ranging from $[-1,1]$.

In the VSS approach, the two users a and u are treated as two vectors in m-dimensional space. Then, the similarity between two vectors can be measured by computing the cosine of the angle between them:

$$\begin{aligned} Sim(a,u) &= cos(\overrightarrow{a},\overrightarrow{u}) = \frac{\overrightarrow{a}\cdot\overrightarrow{u}}{||\overrightarrow{a}||_2 \times ||\overrightarrow{u}||_2} \\ &= \frac{\sum_{i\in I(a)\cap I(u)} r_{a,i}\cdot r_{u,i}}{\sqrt{\sum_{i\in I(a)\cap I(u)} r_{a,i}^2}\cdot\sqrt{\sum_{i\in I(a)\cap I(u)} r_{u,i}^2}}, \end{aligned} \tag{2.3}$$

where $\overrightarrow{a}\cdot\overrightarrow{u}$ denotes the dot-product between the vectors $\overrightarrow{a}$ and $\overrightarrow{u}$.

Once the similarities are calculated, we can easily calculate the values of missing rating $r_{u,i}$ for user u and item i by aggregating the ratings of some other (usually, the N most similar) users for the same item i:

$$r_{u,i} = \text{aggr}_{u'\in U'} r_{u',i}, \tag{2.4}$$

where U' denotes the set of N users that are the most similar to user u and who have rated item i. Some examples of the aggregation function are [2]:

$$r_{u,i} = \frac{1}{N}\sum_{u'\in U'} r_{u',i}, \tag{2.5}$$

$$r_{u,i} = \frac{1}{\sum_{u'\in U'}|Sim(u,u')|}\sum_{u'\in U'} Sim(u,u')\times r_{u',i}, \tag{2.6}$$

$$r_{u,i} = \overline{r}_u + \frac{1}{\sum_{u'\in U'}|Sim(u,u')|}\sum_{u'\in U'} Sim(u,u')\times(r_{u',i}-\overline{r}_{u'}), \tag{2.7}$$

where $\overline{r}_u$ is the average rating of user u.

2.1.2 Model-based Methods

In contrast to the memory-based approaches, the model-based approaches [13, 20, 36, 40, 75, 86, 119] to collaborative filtering use the observed user-item ratings to train a compact model that explains the given data, so that ratings could be predicted via the model instead of directly manipulating the original rating database as the memory-based approaches do [66]. [13] proposed a collaborative filtering method in a machine learning framework, where various machine learning techniques coupled with feature extraction techniques (such as singular value decomposition) can be used. [20] introduced a peer-to-peer protocol for collaborative filtering which protects the privacy of individual data. This work also presented a new collaborative filtering algorithm based on factor analysis which appears to be the most accurate method for collaborative filtering. The new algorithm has other advantages in speed and storage over previous algorithms. It is based on a careful probabilistic model of user choice, and on a probabilistically sound approach to dealing with missing data.

There have been several other model-based collaborative recommendation approaches proposed in the literature. Algorithms in this category include the aspect models [47, 48, 107], Bayesian model [24], relevance models [120, 121], latent class models [49, 54, 76, 107, 104], matrix factorization models [14, 40, 94, 103] and clustering models [6, 35, 56, 87, 116, 117]. [47] proposed an algorithm based on a generalization of probabilistic latent semantic analysis to continuous-valued response variables. [24] proposed a Bayesian approach for the problem of predicting the missing ratings from the observed ratings. This approach incorporates similarity by assuming the set of judges can be par-

titioned into groups which share the same ratings probability distribution. This leads to a predictive distribution of missing ratings based on the posterior distribution of the groupings and associated ratings probabilities. Markov chain Monte Carlo methods and a hybrid search algorithm are used to obtain predictions of the missing ratings. [121] presented a probabilistic user-to-item relevance framework that introduces the concept of relevance into the related problem of collaborative filtering. Experimental results complement the theoretical insights with improved recommendation accuracy. The unified model is more robust to data sparsity, because the different types of ratings are used in concert. [54] conducted a broad and systematic study on different mixture models for collaborative filtering. It discussed general issues related to using a mixture model for collaborative filtering, and proposed three properties that a graphical model is expected to satisfy. Using these properties, this work thoroughly examined five different mixture models, including Bayesian Clustering (BC), Aspect Model (AM), Flexible Mixture Model (FMM), Joint Mixture Model (JMM), and the Decoupled Model (DM). Experiments over two datasets of movie ratings under different configurations show that in general, whether a model satisfies the proposed properties tends to be correlated with its performance. In particular, the Decoupled Model, which satisfies all the three desired properties, outperforms the other mixture models as well as many other existing approaches for collaborative filtering. [56] presented an algorithm for collaborative filtering based on hierarchical clustering, which tried to balance both robustness and accuracy of predictions, especially when few data were available.

More recently, a significant amount of research has been done in trying to model the recommendation process using more complex probabilistic models. For instance, Shani et al. [105] view the recommendation process as a sequential decision problem

and propose using Markov decision processes for generating recommendations. [75] proposed a combination method of multinomial mixture and aspect models using generative semantics of Latent Dirichlet Allocation. Similarly, Si and Jin [107] also use probabilistic latent semantic analysis to propose a flexible mixture model that allows modeling the classes of users and items explicitly with two sets of latent variables. Furthermore, Kumar et al. [62] use a simple probabilistic model to demonstrate that collaborative filtering is valuable with relatively little data on each user, and that, in certain restricted settings, simple collaborative filtering algorithms are almost as effective as the best possible algorithms in terms of utility [2].

Recently, due to the efficiency in dealing with large datasets, several low-dimensional matrix approximation methods [94, 98, 99, 113] have been proposed for collaborative filtering. These methods focus on fitting a factor model to the data, and use it in order to make further predictions.

Low-rank matrix approximations based on minimizing the sum-squared errors can be easily solved using Singular Value Decomposition (SVD), and a simple and efficient Expectation Maximization (EM) algorithm for solving weighted low-rank approximation is proposed in [113]. In [114], Srebro et al. proposed a matrix factorization method to constrain the norms of U and V instead of their dimensionality. Salakhutdinov et al. presented a probabilistic linear model with Gaussian observation noise in [99]. In [98], the Gaussian-Wishart priors are placed on the user and item hyperparameters. Although low-dimensional methods are proved to be very effective and efficient, these methods still suffer several disadvantages that are unveiled. In the SVD method, as well as other well-known methods such as the weighted low-rank approximation method [113], Probabilistic Principal Component Analysis (PPCA) [115], Probabilistic Matrix Factorization (PMF) [99] and Constrained Probabilistic Ma-

trix Factorization [99], the latent features are uninterpretable, and there is no range constraint bound on the latent features vectors. The lack of interpretability results in the improper modeling of the latent factors, hence downgrades the recommendation accuracy. In [127], a nonnegative constraint is imposed on both user-specific features U and item-specific features V (Nonnegative Matrix Factorization).

Actually, the fundamental problem to low-rank matrix factorization is to learn two latent feature spaces of users U and items V. The most fundamental technique is Regularized Matrix Factorization.

Regularized Matrix Factorization

Consider an $m \times n$ user-item rating matrix R, the matrix factorization method employs a rank-l matrix $X = U^T V$ to fit it, where $U \in \mathbb{R}^{l \times m}$ and $V \in \mathbb{R}^{l \times n}$. From the above definition, we can see that the low-dimensional matrices U and V are unknown, and need to be estimated. Moreover, this feature representations have clear physical meanings. In this linear factor model, each factor is a preference vector, and a user's preferences correspond to a linear combination of these factor vectors, with user-specific coefficients. More specifically, each row of U performs as a "feature vector", and each row of V is a linear predictor, predicting the entries in the corresponding column of R based on the "features" in U.

To find matrices U and V, we can solve the following optimization problem:

$$\begin{aligned} \min_{U,V} \mathcal{L}(R, U, V) &= \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij}^{R} (R_{ij} - U_i^T V_j)^2 \\ &+ \frac{\lambda_U}{2} \|U\|_F^2 + \frac{\lambda_V}{2} \|V\|_F^2, \end{aligned} \tag{2.8}$$

where I_{ij}^{R} is the indicator function that is equal to 1 if user u_i

rated item v_j and equal to 0 otherwise, and $\|\cdot\|_F^2$ denotes the Frobenius norm.

A local minimum of the objective function given by Eq. (2.8) can be found by performing gradient descent in U_i, V_j,

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial U_i} &= \sum_{j=1}^{n} I_{ij}^R (U_i^T V_j - R_{ij}) V_j + \lambda_U U_i, \\ \frac{\partial \mathcal{L}}{\partial V_j} &= \sum_{i=1}^{m} I_{ij}^R (U_i^T V_j - R_{ij}) U_i + \lambda_V V_j. \end{aligned} \tag{2.9}$$

2.2 Netflix Prize Competition

The Netflix Prize competition is an important event related to recommendation technologies. It is started and supported by Netflix, a company providing online movie rental services. In October 2006, this company released a large movie rating dataset containing about 100 million ratings from over 480 thousand randomly selected customers on nearly 18 thousand movie items. In Netflix Prize competition, RMSE (Root Mean Square Error)is adopted for performance evaluation and the algorithms in the competition are allowed to output real valued ratings. Lots of new concepts and methods have been proposed during this contest [10, 11, 12, 60, 61, 98, 99, 125, 126, 134].

In [60], Koren proposed a model to combine the latent factor model, which directly profile both users and products, and neighborhood model, which analyze similarities between products or users. The factor and neighborhood models is smoothly merged, thereby building a more accurate combined model. The accuracy improvements are achieved by extending the models to exploit both explicit and implicit feedback by the users. The experiments show that on Netflix data, the proposed method are better than those previously published on this dataset. Most

recently, [61], Koren proposed another collaborative filtering method based on the intuition that customer preferences for products are drifting over time. The proposed model can track the time changing behavior throughout the life span of the data. The experiments results show that this method is better than other state-of-the-art methods.

2.3 Social-based Recommender Systems

Recall that all the above methods for recommender systems are based on the assumption that users are independent and identically distributed, and ignores the social trust relationships between users, which is not consistent with the reality that we normally ask trusted friends for recommendations.

In the most recent research work conducted in [109], by analyzing the who talks to whom social network on the MSN instant messenger[1] over 10 million people with their related search records on the Live Search Engine[2], Singla and Richardson revealed that people who chat with each other (using instant messaging) are more likely to share interests (their Web searches are the same or topically similar). Therefore, to improve the recommendation accuracy, in modern recommender systems, both social network structure and user-item rating matrix should be taken into consideration.

Based on this intuition, many researchers have recently started to analyze trust-based recommender systems.

Trust is type of social relations, and a wide range of research [1, 3] of turst begins from sociologist Gambetta's definition of trust [33]. Trust models have been applied to a wide range of contexts, ranging from onlin reputation systems to dynamic networks [23] and mobile environments [93]; a survey of

[1] http://www.msn.com
[2] http://www.live.com

trust in online service provision can be found in [55].

Recently, trust modeling has been extensively studied in recommender systems [3, 8, 15, 38, 51, 77, 78, 82, 83, 84, 85, 89, 90, 122, 118].

Andersen et al. in [3] developed a set of five natural axioms that a trust-based recommendation system might be expected to satisfy, and then proved that no system can simultaneously satisfy all the axioms. Apparently, this work is out of the scope of this paper since we focus on how to employ both social trust network and user-item matrix to provide more accurate and realistic recommendations. In [77], a trust-aware collaborative filtering method for recommender systems is proposed. In this work, the collaborative filtering process is informed by the reputation of users which is computed by propagating trust. Trust values are computed in addition to similarity measures between users. The experiments on a large real dataset show that this work increases the coverage (number of ratings that are predictable) while not reducing the accuracy (the error of predictions). Bedi et al. [8] proposed a trust-based recommender system for the Semantic Web. This system runs on a server with the knowledge distributed over the network in the form of ontologies, and uses the Web of trust to generate the recommendations.

These methods are all neighborhood-based methods which employ only heuristic algorithms to generate recommendations. There are several problems with this approach, however. The relationship between the trust network and the user-item matrix have not been studied systematically. Moreover, these methods are not scalable to very large datasets since they may need to calculate the pairwise user similarities and pairwise user trust scores. In this thesis, we propose three effective and efficient model-based methods to help solve these problems.

□ **End of chapter.**

Chapter 3

Effective Missing Data Prediction

This chapter focuses the memory-based collaborative filtering problems on two crucial factors: (1) similarity computation between users or items and (2) missing data prediction algorithms. First, we use the enhanced Pearson Correlation Coefficient (PCC) algorithm by adding one parameter which overcomes the potential decrease of accuracy when computing the similarity of users or items. Second, we propose an effective missing data prediction algorithm, in which information of both users and items is taken into account. In this algorithm, we set the similarity threshold for users and items respectively, and the prediction algorithm will determine whether predicting the missing data or not. We also address how to predict the missing data by employing a combination of user and item information. Finally, empirical studies on dataset MovieLens have shown that our newly proposed method outperforms other state-of-the-art collaborative filtering algorithms and it is more robust against data sparsity.

3.1 Similarity Computation

This section briefly introduces the similarity computation methods in traditional user-based and item-based collaborative filtering [17, 28, 44, 102] as well as the method proposed in this chapter. Given a recommendation system consists of M users and N items, the relationship between users and items is denoted by an $M \times N$ matrix, called the user-item matrix. Every entry in this matrix $r_{m,n}$ represents the score value, r, that user m rates an item n, where $r \in \{1, 2, ..., r_{max}\}$. If user m does not rate the item n, then $r_{m,n} = 0$.

3.1.1 Pearson Correlation Coefficient

User-based collaborative filtering engaging PCC was used in a number of recommendation systems [106], since it can be easily implemented and can achieve high accuracy when comparing with other similarity computation methods. In user-based collaborative filtering, PCC is employed to define the similarity between two users a and u based on the items they rated in common:

$$Sim(a,u) = \frac{\sum_{i \in I(a) \cap I(u)} (r_{a,i} - \overline{r}_a) \cdot (r_{u,i} - \overline{r}_u)}{\sqrt{\sum_{i \in I(a) \cap I(u)} (r_{a,i} - \overline{r}_a)^2} \cdot \sqrt{\sum_{i \in I(a) \cap I(u)} (r_{u,i} - \overline{r}_u)^2}}, \tag{3.1}$$

where $Sim(a,u)$ denotes the similarity between user a and user u, and i belongs to the subset of items which user a and user u both rated. $r_{a,i}$ is the rate user a gave item i, and $\overline{r}_a$ represents the average rate of user a. From this definition, user similarity $Sim(a,u)$ is ranging from $[-1, 1]$, and a larger value means users a and u are more similar.

Item-based methods such as [28, 102] are similar to user-based

approaches, and the difference is that item-based methods employ the similarity between the items instead of users. The basic idea in similarity computation between two items i and j is to first isolate the users who have rated both of these items and then apply a similarity computation technique to determine the similarity $Sim(i, j)$ [102]. The PCC-based similarity computation between two items i and j can be described as:

$$Sim(i,j) = \frac{\sum_{u \in U(i) \cap U(j)} (r_{u,i} - \overline{r}_i) \cdot (r_{u,j} - \overline{r}_j)}{\sqrt{\sum_{u \in U(i) \cap U(j)} (r_{u,i} - \overline{r}_i)^2} \cdot \sqrt{\sum_{u \in U(i) \cap U(j)} (r_{u,j} - \overline{r}_j)^2}}, \tag{3.2}$$

where $Sim(i, j)$ is the similarity between item i and item j, and u belongs to the subset of users who both rated item i and item j. $r_{u,i}$ is the rate user u gave item i, and $\overline{r}_i$ represents the average rate of item i. Like user similarity, item similarity $Sim(i, j)$ is also ranging from $[-1, 1]$.

3.1.2 Significance Weighting

PCC-based collaborative filtering generally can achieve higher performance than other popular algorithms like VSS [17], since it considers the factor of the differences of user rating styles. However PCC will overestimate the similarities of users who happen to have rated a few items identically, but may not have similar overall preferences [79]. Herlocker et al. [43, 44] proposed to add a correlation significance weighting factor that would devalue similarity weights that were based on a small number of co-rated items. Herlocker's latest research work [79] proposed to use the following modified similarity computation equation:

$$Sim'(a,u) = \frac{Max(|I_a \cap I_u|, \gamma)}{\gamma} \cdot Sim(a,u). \tag{3.3}$$

This equation overcomes the problem when only few items are rated in common but in case that when $|I_a \cap I_u|$ is much higher than γ, the similarity $Sim'(a, u)$ will be larger than 1, and even surpass 2 or 3 in worse cases. We use the following equation to solve this problem:

$$Sim'(a, u) = \frac{Min(|I_a \cap I_u|, \gamma)}{\gamma} \cdot Sim(a, u), \tag{3.4}$$

where $|I_a \cap I_u|$ is the number of items which user a and user u rated in common. This change bounds the similarity $Sim'(a, u)$ to the interval $[0, 1]$. Then the similarity between items could be defined as:

$$Sim'(i, j) = \frac{Min(|U_i \cap U_j|, \delta)}{\delta} \cdot Sim(i, j), \tag{3.5}$$

where $|U_i \cap U_j|$ is the number of users who rated both item i and item j.

3.2 Collaborative Filtering Framework

In pratice, the user-item matrix of commercial recommendation system is very sparse and the density of available ratings is often less than 1% [102]. Sparse matrix directly leads to the prediction inaccuracy in traditional user-based or item-based collaborative filtering. Some work applies data smoothing methods to fill the missing values of the user-item matrix. In [124], Xue et al. proposed a cluster-based smoothing method which clusters the users using K-means first, and then predicts all the missing data based on the ratings of Top-N most similar users in the similar clusters. The simulation shows this method could generate better results than other collaborative filtering algorithms. But cluster-based method limits the diversity of users in each cluster, and the clustering results of K-means relies on the preselected K users. Furthermore, if a user does not have enough

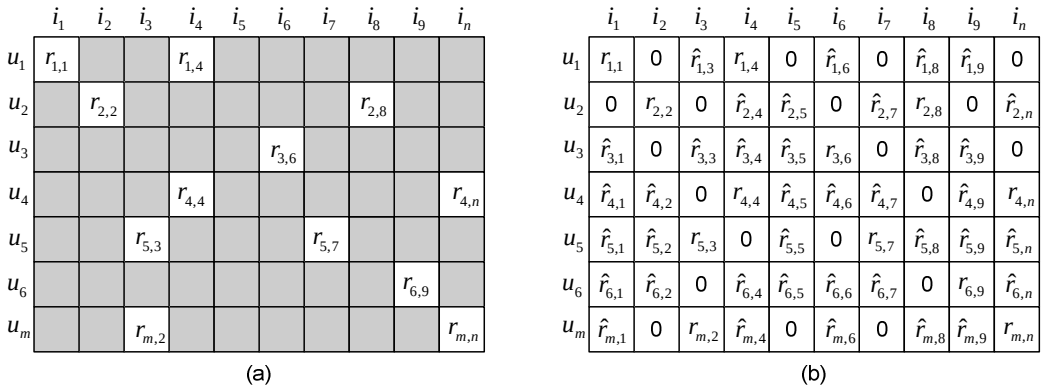


Figure 3.1: (a) The user-item matrix ($m \times n$) before missing data prediction. (b) The user-item matrix ($m \times n$) after missing data prediction.

similar users, then Top-N algorithm generates a lot of dissimilar users which definitely will decrease the prediction accuracy of the active users.

According to the analysis above, we propose a novel effective missing data prediction algorithm which predicts the missing data when it fits the criteria we set. Otherwise, we will not predict the missing data and keep the value of the missing data to be zero. As illustrated in Fig. 3.1(a), before we predict the missing data, the user-item matrix is a very sparse matrix and every user only rates few items with $r_{u,i}$; at the same time, other unrated data are covered with shade. Using this sparse matrix to predict ratings for active users always results in giving bad recommendations to the active users. In our approach, we evaluate every shaded block (missing data) using the available information in Fig. 3.1(a). For every shaded block, if our algorithm achieves confidence in the prediction, then we give this shaded block a predicted rating value $\widehat{r}_{u,i}$. Otherwise, we set the value of this missing data to zero, as seen in Fig. 3.1(b).

Accordingly, the collaborative filtering is simplified into two simple questions. The first is "Under what circumstance does our algorithm have confidence to predict the shaded block?"

and the second is "How to predict?". The following will answer these two questions.

3.2.1 Similar Neighbors Selection

Similar neighbors selection is a very important step in predicting missing data. If selected neighbors are dissimilar with the current user, then the prediction of missing data of this user is inaccurate and will finally affect the prediction results of the active users. In order to overcome the flaws of Top-N neighbors selection algorithms, we introduce a threshold η. If the similarity between the neighbor and the current user is larger than η, then this neighbor is selected as the similar user.

For every missing data $r_{u,i}$, a set of similar users $S(u)$ towards user u can be generated according to:

$$S(u) = \{u_a | Sim'(u_a, u) > \eta, u_a \neq u\}, \tag{3.6}$$

where $Sim'(u_a, u)$ is computed using Eq. (3.4). At the same time, for every missing data $r_{u,i}$, a set of similar items $S(i)$ towards item i can be generated according to:

$$S(i) = \{i_k | Sim'(i_k, i) > \theta, i_k \neq i\}, \tag{3.7}$$

where θ is the item similarity threshold, and $Sim'(i_k, i)$ is computed by Eq. (3.5). The selection of η and θ is an important step since a very big value will always cause the shortage of similar users or items, and a relative small value will bring too many similar users or items.

According to Eq.(3.6) and Eq.(3.7), we define that our algorithm will lack enough confidence to predict the missing data $r_{u,i}$ if and only if $S(u) = \emptyset \wedge S(i) = \emptyset$, which means that user u does not have similar users and item i does not have similar items either. Then our algorithm sets the value of this missing data to zero. Otherwise, it will predict the missing data $r_{u,i}$ following the algorithm described in Section 3.2.2.

3.2.2 Missing Data Prediction

User-based collaborative filtering predicts the missing data using the ratings of similar users and item-based collaborative filtering predicts the missing data using the ratings of similar items. Actually, although users have their own rating style, if an item is a very popular item and has obtained a very high average rating from other users, then the active user will have a high probability to give this item a good rating too. Hence, predicting missing data only using user-based approaches or only using item-based approaches will potentially ignore valuable information that will make the prediction more accurate. We propose to systematically combine user-based and item-based approaches, and take advantage of user correlations and item correlations in the user-item matrix.

Given the missing data $r_{u,i}$, according to Eq. (3.6) and Eq. (3.7), if $S(u) \neq \emptyset \wedge S(i) \neq \emptyset$, the prediction of missing data $P(r_{u,i})$ is defined as:

$$P(r_{u,i}) = \lambda \times (\overline{u} + \frac{\sum_{u_a \in S(u)} Sim'(u_a, u) \cdot (r_{u_a,i} - \overline{u}_a)}{\sum_{u_a \in S(u)} Sim'(u_a, u)}) + (1 - \lambda) \times (\overline{i} + \frac{\sum_{i_k \in S(i)} Sim'(i_k, i) \cdot (r_{u,i_k} - \overline{i}_k)}{\sum_{i_k \in S(i)} Sim'(i_k, i)}), \quad (3.8)$$

where λ is the parameter in the range of $[0, 1]$. The use of parameter λ allows us to determine how the prediction relies on user-based prediction and item-based prediction. $\lambda = 1$ states that $P(r_{u,i})$ depends completely upon ratings from user-based prediction and $\lambda = 0$ states that $P(r_{u,i})$ depends completely upon ratings from item-based prediction.

In practice, some users do not have similar users and the similarities between these users and all other users are less than the threshold η. Top-N algorithms will ignore this problem and still choose the top n most similar users to predict the missing data. This will definitely decrease the prediction quality of the missing data. In order to predict the missing data as accurate as possible, in case some users do not have similar users, we use the information of similar items instead of users to predict the missing data, and vice versa, as seen in Eq. (3.9) and Eq. (3.10). This consideration inspires us to fully utilize the information of user-item matrix as follows:

If $S(u) \neq \emptyset \wedge S(i) = \emptyset$, the prediction of missing data $P(r_{u,i})$ is defined as:

$$P(r_{u,i}) = \overline{u} + \frac{\sum_{u_a \in S(u)} Sim'(u_a, u) \cdot (r_{u_a,i} - \overline{u}_a)}{\sum_{u_a \in S(u)} Sim'(u_a, u)}. \tag{3.9}$$

If $S(u) = \emptyset \wedge S(i) \neq \emptyset$, the prediction of missing data $P(r_{u,i})$ is defined as:

$$P(r_{u,i}) = \overline{i} + \frac{\sum_{i_k \in S(i)} Sim'(i_k, i) \cdot (r_{u,i_k} - \overline{i}_k)}{\sum_{i_k \in S(i)} Sim'(i_k, i)}. \tag{3.10}$$

The last possibility is given the missing data $r_{u,i}$, user u does not have similar users and at the same time, item i also does not have similar items. In this situation, we choose not to predict the missing data; otherwise, it will bring negative influence to the prediction of the missing data $r_{u,i}$. That is:

If $S(u) = \emptyset \wedge S(i) = \emptyset$, the prediction of missing data $P(r_{u,i})$ is defined as:

$$P(r_{u,i}) = 0. \tag{3.11}$$

This consideration is different from all other existing prediction or smoothing methods. They always try to predict all the missing data in the user-item matrix, which will predict some missing data with bad quality.

3.2.3 Prediction for Active Users

After the missing data is predicted in the user-item matrix, the next step is to predict the ratings for the active users. The prediction process is almost the same as predicting the missing data, and the only difference is in the case for a given active user a; namely, if $S(a) = \emptyset \wedge S(i) = \emptyset$, then predicts the missing data using the following equation:

$$P(r_{a,i}) = \lambda \times \overline{r}_a + (1 - \lambda) \times \overline{r}_i. \tag{3.12}$$

In other situations, if (1) $S(u) \neq \emptyset \wedge S(i) \neq \emptyset$, (2) $S(u) \neq \emptyset \wedge S(i) = \emptyset$ or (3) $S(u) = \emptyset \wedge S(i) \neq \emptyset$, we use Eq. (3.8), Eq. (3.9) and Eq. (3.10) to predict $r_{a,i}$, respectively.

3.2.4 Parameter Discussion

The thresholds γ and δ introduced in Section 3.1 are employed to avoid overestimating the users similarity and items similarity, when there are only few ratings in common. If we set γ and δ too high, most of the similarities between users or items need to be multiplied with the significance weight, and it is not the results we expect. However, if we set γ and δ too low, it is also not reasonable because the overestimate problem still exists. Tuning these parameters is important to achieving a good prediction results.

The thresholds η and θ introduced in Section 3.2.1 also play an important role in our collaborative filtering algorithm. If η and θ are set too high, less missing data need to be predicted; if they are set too low, a lot of missing data need to be predicted.

Table 3.1: The relationship between parameters with other CF approaches

Lambda	Eta	Theta	Related CF Approaches
1	1	1	User-based CF without missing data prediction
0	1	1	Item-based CF without missing data prediction
1	0	0	User-based CF with all the missing data predicted
0	0	0	Item-based CF with all the missing data predicted

In the case when $\eta = 1$ and $\theta = 1$, our approach will not predict any missing data, and this algorithm becomes the general collaborative filtering without data smoothing. In the case when $\eta = 0$ and $\theta = 0$, our approach will predict all the missing data, and this algorithm converges to the Top-N neighbors selection algorithms, except the number N here includes all the neighbors. In order to simplify our model, we set $\eta = \theta$ in all the simulations.

Finally, parameter λ introduced in Section 3.2.2 is the last parameter we need to tune, and it is also the most important one. λ determines how closely the rating prediction relies on user information or item information. As discussed before, $\lambda = 1$ states that $P(r_{u,i})$ depends completely upon ratings from user-based prediction and $\lambda = 0$ states that $P(r_{u,i})$ depends completely upon ratings from item-based prediction. This physical interpretation also helps us to tune λ accordingly.

With the changes of parameters, several other famous collaborative filtering methods become special cases in our approach as illustrated in Table 3.1.

3.3 Empirical Analysis

We conduct several experiments to measure the recommendation quality of our new approach for collaborative filtering with other methods, and address the experiments as the following questions:

1. How does our approach compare with traditional user-based and item-based collaborative filtering methods?

2. What is the performance comparison between our effective missing data prediction approach and other algorithms which predict every missing data?

3. How does significance weighting affect the accuracy of prediction?

4. How do the thresholds η and θ affect the accuracy of prediction? How many missing data are predicted by our algorithm, and what is the comparison of our algorithm with the algorithms that predict all the missing data or no missing data?

5. How does the parameter λ affect the accuracy of prediction?

6. How does our approach compare with the published state-of-the-art collaborative filtering algorithms?

In the following, Section 3.3.3 gives answers to questions 1 and 6, Section 3.3.4 addresses question 2, and Section 3.3.5 describes experiment for the questions 3 to 5.

3.3.1 Dataset

Two datasets from movie rating are applied in our experiments: MovieLens[1] and EachMovie[2]. We only report the simulation results of MovieLens due to the space limitation. Similar results can be observed from the EachMovie application.

MovieLens is a famous Web-based research recommender system. It contains 100,000 ratings (1-5 scales) rated by 943 users

[1]http://www.cs.umn.edu/Research/GroupLens/.

[2]http://www.research.digital.com/SRC/EachMovie/. It is retired by Hewlett-Packard (HP), but a postprocessed copy can be found on http://guir.berkeley.edu/projects/swami/.

Table 3.2: Statistics of Dataset MovieLens

Statistics	User	Item
Min. Num. of Ratings	20	1
Max. Num. of Ratings	737	583
Avg. Num. of Ratings	106.04	59.45

on 1682 movies, and each user at least rated 20 movies. The density of the user-item matrix is:

$$\frac{100000}{943 \times 1682} = 6.30\%.$$

The statistics of dataset MovieLens is summarized in Table 3.2.

We extract a subset of 500 users from the dataset, and divide it into two parts: select 300 users as the training users (100, 200, 300 users respectively), and the rest 200 users as the active (testing) users. As to the active users, we vary the number of rated items provided by the active users from 5, 10, to 20, and give the name Given5, Given10 and Given20, respectively.

3.3.2 Metrics

We use the Mean Absolute Error (MAE) metrics to measure the prediction quality of our proposed approach with other collaborative filtering methods. MAE is defined as:

$$MAE = \frac{\sum_{u,i} |r_{u,i} - \widehat{r}_{u,i}|}{N}, \tag{3.13}$$

where $r_{u,i}$ denotes the rating that user u gave to item i, and $\widehat{r}_{u,i}$ denotes the rating that user u gave to item i which is predicted by our approach, and N denotes the number of tested ratings.

3.3.3 Comparison

In order to show the performance increase of our effective missing data prediction (EMDP) algorithm, we compare our algo-

Table 3.3: MAE comparison with other approaches (A smaller MAE value means a better performance).

Training Users	Methods	Given5	Given10	Given20
MovieLens 300	EMDP	**0.784**	**0.765**	**0.755**
	UPCC	0.838	0.814	0.802
	IPCC	0.870	0.838	0.813
MovieLens 200	EMDP	**0.796**	**0.770**	**0.761**
	UPCC	0.843	0.822	0.807
	IPCC	0.855	0.834	0.812
MovieLens 100	EMDP	**0.811**	**0.778**	**0.769**
	UPCC	0.876	0.847	0.811
	IPCC	0.890	0.850	0.824

Table 3.4: MAE comparison with state-of-the-arts algorithms (A smaller MAE value means a better performance).

Num. of Training Users	100			200			300		
Given Ratings	5	10	20	5	10	20	5	10	20
EMDP	**0.807**	**0.769**	**0.765**	**0.793**	**0.760**	**0.751**	**0.788**	**0.754**	**0.746**
SF	0.847	0.774	0.792	0.827	0.773	0.783	0.804	0.761	0.769
SCBPCC	0.848	0.819	0.789	0.831	0.813	0.784	0.822	0.810	0.778
AM	0.963	0.922	0.887	0.849	0.837	0.815	0.820	0.822	0.796
PD	0.849	0.817	0.808	0.836	0.815	0.792	0.827	0.815	0.789
PCC	0.874	0.836	0.818	0.859	0.829	0.813	0.849	0.841	0.820

rithm with some traditional algorithms: user-based algorithm using PCC (UPCC) and item-based algorithm using PCC (IPCC). The parameters or thresholds for the experiments are empirically set as follows: $\lambda = 0.7$, $\gamma = 30$, $\delta = 25$, $\eta = \theta = 0.4$.

In Table 3.3, we observe that our new approach significantly improves the recommendation quality of collaborative filtering, and outperforms UPCC and IPCC consistently.

Next, in order to compare our approach with other state-of-the-arts algorithms, we follow the exact evaluation procedures

which were described in [119, 124] by extracting a subset of 500 users with more than 40 ratings. Table 3.4 summarizes our experimental results. We compare with the following algorithms: Similarity Fusion (SF) [119], Smoothing and Cluster-Based PCC (SCBPCC) [124], the Aspect Model (AM) [48], Personality Diagnosis (PD) [88] and the user-based PCC [17]. Our method outperforms all other competitive algorithms in various configurations.

3.3.4 Impact of Missing Data Prediction

Our algorithm incorporates the option not to predict the missing data if it does not meet the criteria set in Section 3.2.1 and Section 3.2.2. In addition, it alleviates the potential negative influences from bad prediction on the missing data. To demonstrate the effectiveness of our approach, we first conduct a set of simulations on our effective missing data prediction approach. The number of training users is 300, where we set $\gamma = 30$, $\delta = 25$, $\eta = \theta = 0.5$, and vary λ from zero to one with a step value of 0.05. We then plot the graph with the ratings of active users of Given5, Given10 and Given20, respectively. As to the method in predicting every missing data (PEMD), we use the same algorithm, and keep the configurations the same as EMDP except for Eq. (3.11). In PEMD, when $S(u) = \emptyset$ and $S(i) = \emptyset$, we predict the missing data $r_{u,i}$ using the nearest neighbors of the missing data instead of setting the value to zero. In this experiment, we set the number of nearest neighbors to 10. The intention of this experiment is to compare the performance of our EMDP algorithm with PEMD under the same configurations. In other words, we intend to determine the effectiveness of our missing data prediction algorithm, and whether our approach is better than the approach which will predict every missing data or not.

In Fig. 3.2, the star, up triangle, and diamond in solid line

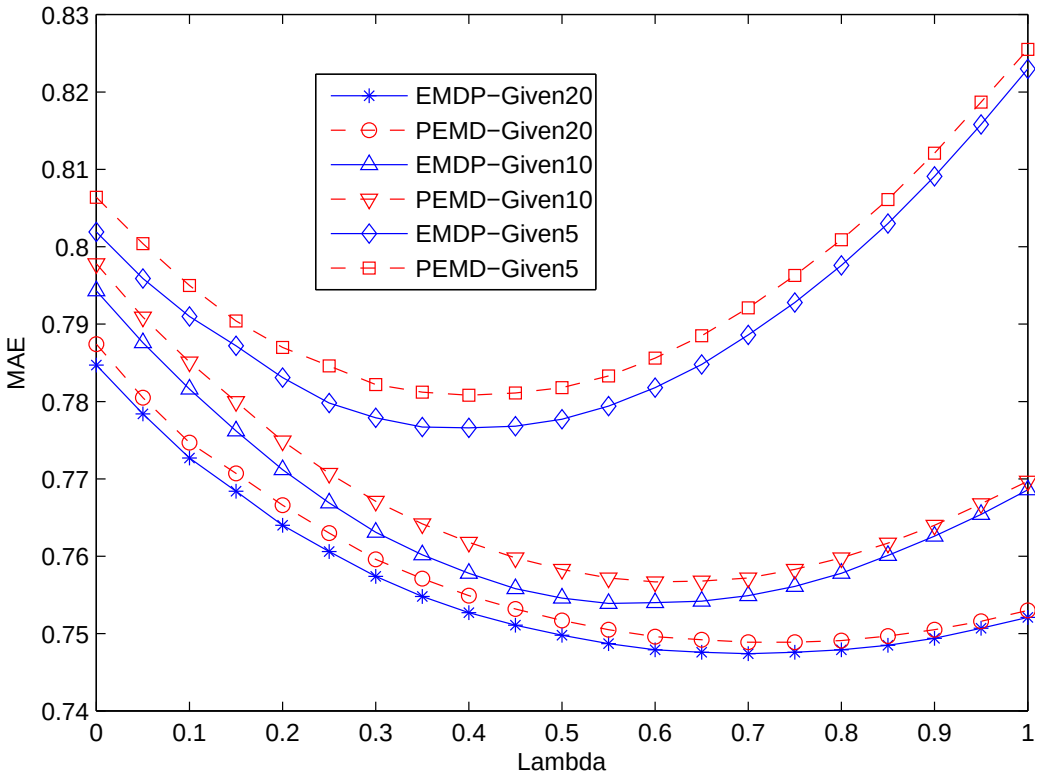


Figure 3.2: MAE Comparison of EMDP and PEMD (A smaller MAE value means a better performance).

represent the EMDP algorithm in Given20, Given10 and Given5 ratings respectively, and the circle, down triangle, and square in dashed line represent the PEMD algorithm in Given20, Given10 and Given5 ratings respectively. All the solid lines are below the respectively comparative dashed lines, indicating our effective missing data prediction algorithm performs better than the algorithm which predict every missing data, and predicting missing data selectively is indeed a more effective method.

3.3.5 Impact of Parameters

γ and δ in Significance Weighting

Significance weighting makes the similarity computation more reasonable in practice and devalues some similarities which look

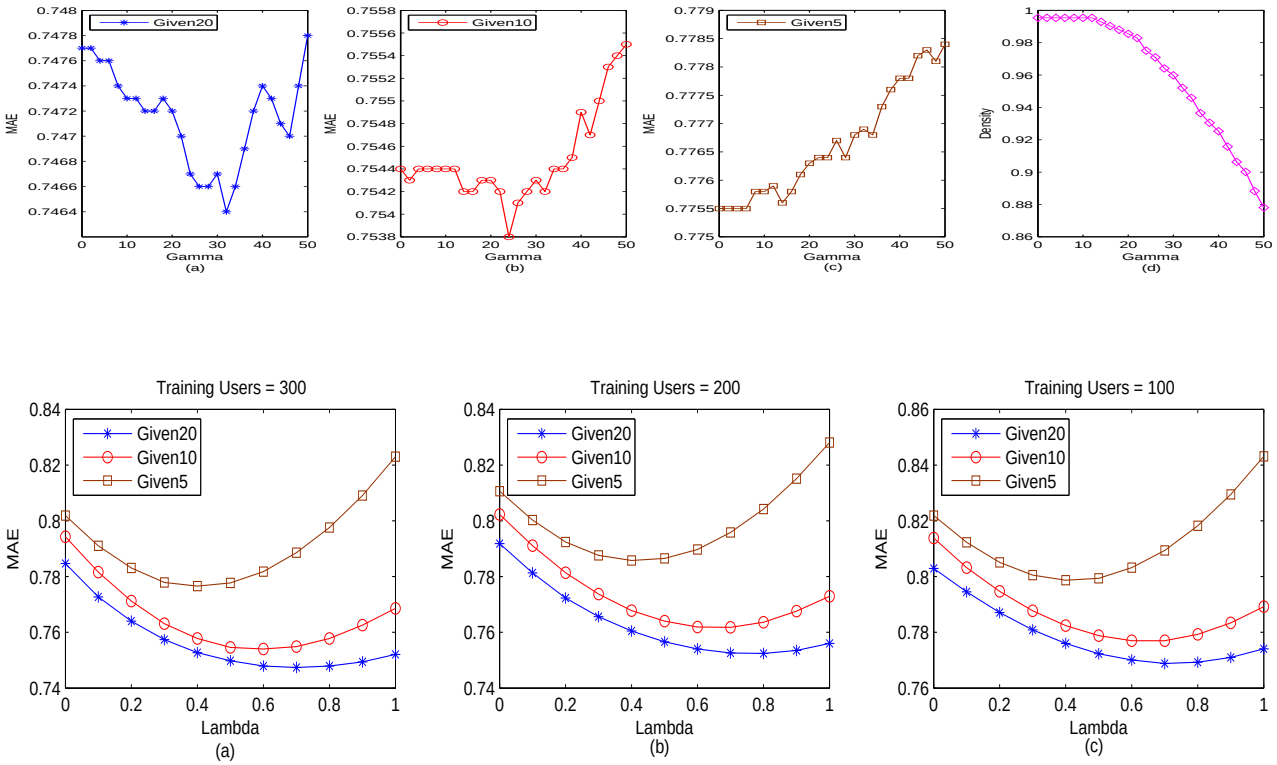


Figure 3.4: Impact of Lambda on MAE

similar but are actually not, and the simulation results in Fig. 3.3 shows the significance weighting will promote the collaborative filtering performance.

In this experiment, we first evaluate the influence of γ, and select 300 training users, then set $\lambda = 0.7$, $\eta = \theta = 0.5$, $\delta = 26$. We vary the range of γ from 0 to 50 with a step value of 2. Fig. 3.3(a),(b),(c) shows how γ affects MAE when given ratings 20, 10, 5 respectively, and Fig. 3.3(d) shows that the value of γ also impacts the density of the user-item matrix in the process of missing data prediction. The density of the user-item matrix will decrease according to the increase of the value of γ. More experiments show that δ has the same features and impacts on MAE and matrix density as γ; however, we do not include the simulation results due to the space limitation.

Impact of λ

Parameter λ balances the information from users and items. It takes advantages from these two types of collaborative filtering methods. If $\lambda = 1$, we only extract information from users, and if $\lambda = 0$, we only mine valuable information from items. In other cases, we fuse information from users and items to predict the missing data and furthermore, to predict for active users.

Fig. 3.4 shows the impacts of λ on MAE. In this experiment, we test 300 training users, 200 training users and 100 training users and report the experiment results in Fig. 3.4(a), Fig. 3.4(b) and Fig. 3.4(c) respectively. The initial values of other parameters or thresholds are: $\eta = \theta = 0.5$, $\gamma = 30$, $\delta = 25$.

Observed from Fig. 3.4, we draw the conclusion that the value of λ impacts the recommendation results significantly, which demonstrates that combining the user-based method with the item-based method will greatly improve the recommendation accuracy. Another interesting observation is when following the increase of the number of ratings given (from 5 to 10, and from 10 to 20), the value of $\arg\min_\lambda(MAE)$ of each curve in Fig. 3.4 shifts from 0.3 to 0.8 smoothly. This implies the information for users is more important than that for items if more ratings for active users are given. On the other hand, the information for items would be more important if less ratings for active users are available; however, less ratings for active users will lead to more inaccuracy of the recommendation results.

Impact of η and θ

η and θ also play a very important role in our collaborative filtering approach. As discussed in Section 3.2, η and θ directly determine how many missing data need to be predicted. If η and θ are set too high, most of the missing data cannot be predicted since many users will not have similar users, and many items will

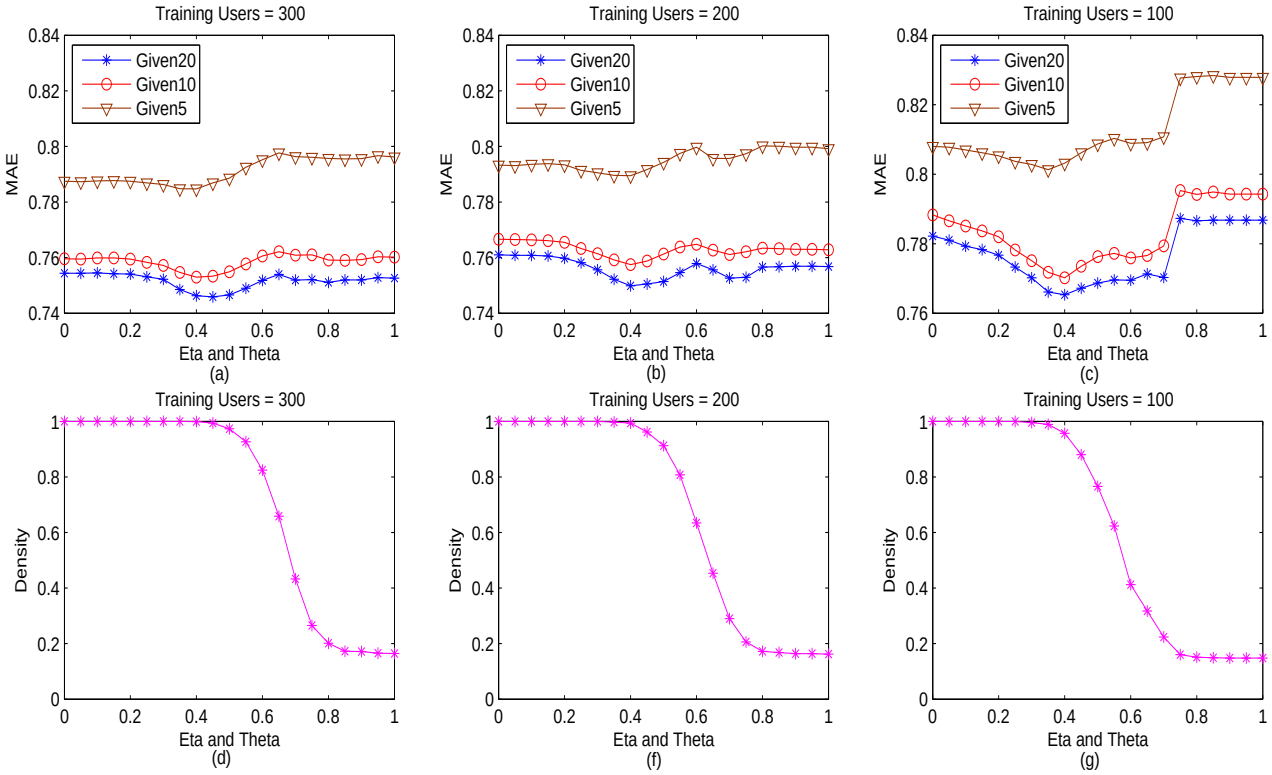


Figure 3.5: Impact of Eta and Theta on MAE and Density

not have similar items either. On the other hand, if η and θ are set too low, every user or item will obtain too many similar users or items, which causes the computation inaccuracy and increases the computing cost. Accordingly, selecting proper values for η and θ is as critical as determining the value for λ. In order to simplify our model, we set $\eta = \theta$ as employed in our experiments.

In the next experiment, we select 500 users from MovieLens dataset and extract 300 users for training users and other 200 as the active users. The initial values for every parameter and threshold are: $\lambda = 0.7$, $\gamma = 30$, $\delta = 25$. We vary the values of η and θ from 0 to 1 with a step value of 0.05. For each training user set (100, 200, 300 users respectively), we compute the MAE and density of the user-item matrix. The results are showed in Fig. 3.5.

As showed in Fig. 3.5(a), given 300 training users and given 20 ratings for every active user, this algorithm will achieve the best performance around $\eta = \theta = 0.50$, and the related density of user-item matrix in Fig. 3.5(d) is 92.64% which shows that 7.36% missing data of this user-item matrix are not predicted. In this experiment, the number of data that was not predicted is $0.0736 \times 500 \times 1000 = 36800$. We observe that around $\eta = \theta = 0.70$, this algorithm already achieves a very good MAE value which is almost the same as the best MAE values in Fig. 3.5(b). The related matrix density is 29.00%, which illustrates that more than 70% data of user-item matrix are not predicted. Nevertheless, the algorithm can already achieve satisfactory performance.

3.4 Summary

In this chapter, we propose an effective missing data prediction algorithm for collaborative filtering. By judging whether a user (an item) has other similar users (items), our approach determines whether to predict the missing data and how to predict the missing data by using information of users, items or both. Traditional user-based collaborative filtering and item-based collaborative filtering approaches are two subsets of our new approach. Empirical analysis shows that our proposed EMDP algorithm for collaborative filtering outperforms other state-of-the-art collaborative filtering approaches.

□ **End of chapter.**

Chapter 4

Recommend with Global Consistency

Recently, due to its efficiency in handling very large datasets, low-dimensional factor models have become one of the most popular approaches in the model-based collaborative filtering algorithms. The premise behind a low-dimensional 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 [94].

Although these methods can effectively predict missing values, several disadvantages are unveiled, which will potentially decrease the prediction accuracy. First, in low-rank factor-based approaches, both item factor vectors and user-specific coefficients are understood as latent factors which have no physical meanings, and hence uninterpretable. Moreover, the lack of interpretability will result in the improper modeling of the latent factors. For example, these latent factors in [98, 99] are set to be in the Euclidean space, while they are nonnegative in [127]. Second, due to the sparsity of the user-item rating matrix (the density of available ratings in commercial recommender systems is often less than 1% [102]), many matrix factorization methods fail to provide accurate recommendations. In the sparse user-item rating matrix, the ratings for training the user features are

rare, hence the learned user features and the coefficients cannot accurately reflect the taste of users, which will result in the bad prediction accuracy.

In this chapter, aiming at providing solutions for the issues analyzed above, we propose a Semi-Nonnegative Matrix Factorization with Global Statistical Consistency (SNGSC) approach for collaborative filtering.

4.1 Framework

4.1.1 Problem Definition

Without loss of generality, in this chapter, we use the movie recommender systems as the example. In a collaborative prediction movie recommendation system, the inputs to the system are user ratings on the movies the users have already seen. Prediction of user preferences on the movies they have not yet seen are then based on patterns in the partially observed rating matrix $X \in R_+^{n \times m}$, where n is the number of users, and m is the number of movies. The value X_{ij} indicates the score of item j rated by user i. This approach contrasts with feature-based approach where predictions are made based on features of the movies (e.g. genre, year, actors, external reviews) and the users (e.g. age, gender, explicitly specified preferences, social trust networks [69, 74]). Users "collaborate" by sharing their ratings instead of relying on external information [94].

Table 4.1 and Table 4.2 are the toy examples on the problem we study. As illustrated in Table 4.1, each user (from u_1 to u_6) rated some items (from i_1 to i_8) on a 5-point integer scale to express the extent of favor of each item. The problem we study in this chapter is how to predict the missing values of the user-item matrix effectively and efficiently.

Table 4.1: User-Item Matrix

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	5	2		3		4		
u_2	4	3			5			
u_3	4		2				2	4
u_4								
u_5	5	1	2		4	3		
u_6	4	3		2	4		3	5

Table 4.2: Predicted User-Item Matrix

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	5	2	**2.5**	3	**4.8**	4	**2.2**	**4.8**
u_2	4	3	**2.4**	**2.9**	5	**4.1**	**2.6**	**4.7**
u_3	4	**1.7**	2	**3.2**	**3.9**	**3.0**	2	4
u_4	**4.8**	**2.1**	**2.7**	**2.6**	**4.7**	**3.8**	**2.4**	**4.9**
u_5	5	1	2	**3.4**	4	3	**1.5**	**4.6**
u_6	4	3	**2.9**	2	4	**3.4**	3	5

4.1.2 How is user-item matrix X generated?

The $n \times m$ matrix X contains the ratings of users on items. X is generated by the users who rate the movies according to their overall feeling about the movies that they have seen. By anatomizing their overall feeling, we give a detailed analysis on the rating process as follows.

Each user has a different taste on different type of genre, actors, or something else. But with the only given rating matrix, the information for genre or actors is unknown, so we assume there are d different unknown types of objects, which are named as latent types. We further assume that user i has confidence U_{ik} ($U_{ik} \in R_+$) on k-th type, and U_{ik} is also the taste of user i

in ranking objects of type k; on the other hand, on k-th type, each item j has a "true" quality value V_{jk} ($V_{jk} \in R$). So to user i, item j should be rated by user i as $U_{ik} * V_{jk}$. As a result, on k-th type, if both the quality of object j and the taste of user i are high, then user i will rate object j with a high score.

These d latent types may have cross-effects on each other. For example, War type movies may also belong to classic Hollywood sub-category. Considering the cross-effects, we assume a symmetric non-negative matrix $\Sigma_{d\times d}$, in which $\Sigma_{kl} = \Sigma_{lk}$ denotes the cross-effect between type k and l, and $\Sigma_{kk} = \lambda_k$. Ideally, we hope that the d latent types are independent, and their significance can be ordered, i.e., nonnegative significance values $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$ can be assigned to the d latent types.

Consequently, on type k, user i rates item j with a score

$$\sum_{l=1}^{d} U_{ik} * V_{jl} * \Sigma_{kl},$$

where the quality V_{jl} of item j on type l is transferred to quality $V_{jl} * \Sigma_{kl}$ by Σ_{kl}. Note that, if $\Sigma_{d\times d}$ is diagonal, then it becomes $\sum_{l=1}^{d} \lambda_k * U_{ik} * V_{jk}$. Accumulating all the different unknown types, we obtain that

$$\sum_{k=1}^{d}\sum_{l=1}^{d} U_{ik} * V_{jl} * \Sigma_{kl} = (U\Sigma V^T)_{ij},$$

where U_k is the vector consisting of U_{ik}, V_k is the vector consisting of V_{jk}, and $U = (U_1, U_2, \ldots, U_d)$ and $V = (V_1, V_2, \ldots, V_d)$. We consider factorizations of the form $X \approx U\Sigma V^T$, where $U \in R_+^{n\times d}$, $\Sigma \in R_+^{d\times d}$, and $V \in R^{m\times d}$.

Remark. According to the physical meaning of U and V, U is nonnegative while V should be unrestricted. For example, a movie may be very bad so that everyone dislikes it, and hence the quality of this movie can be scored as -1. The confidence

is the ability of a user to rate a movie, and so should not be negative. To explain it further, if the confidence of a user is also set as -1, then the product of -1 and -1 will be 1, which means that a user with low confidence rates a bad movie with a high score, which is not true in reality. On the contrary, the setting $U_i \in R_+^n$ avoids such unreasonable cases, leading to the advantage of the interpretability of U.

4.1.3 Sensitivity Analysis

We find U, Σ, and V so that $P = U\Sigma V^T$ approximates X well. But it is not preferable that small changes (due to computing errors or error propagated from observation errors in X) in these three matrices result in a big change in their product. Since the derivatives with respect to the variables U, Σ, and V mean the change rate, we examine the square sum of the corresponding derivatives. Let the notation $||\cdot||_F$ denote the Frobenius norm.

By $\frac{\partial (BA)_{ij}}{\partial B_{mn}} = \delta_{im}(A)_{nj}$, we have

$$\begin{aligned}
\sum_{ijmn}\left(\frac{\partial (U\Sigma V^T)_{ij}}{\partial U_{mk}}\right)^2 &= \sum_{ijmk}\left(\delta_{im}(\Sigma V^T)_{kj}\right)^2 \\
&= \sum_{ijk}\left((\Sigma V^T)_{kj}\right)^2 \\
&= n\sum_{jk}\left((\Sigma V^T)_{kj}\right)^2 \\
&= n||\Sigma V^T||_F^2. \qquad (4.1)
\end{aligned}$$

Similarly we have

$$\sum_{ijmn}\left(\frac{\partial (U\Sigma V^T)_{ij}}{\partial V_{mk}}\right)^2 = m||U\Sigma||_F^2, \qquad (4.2)$$

$$\sum_{ijmn}\left(\frac{\partial (U\Sigma V^T)_{ij}}{\partial \Sigma_{mk}}\right)^2 = d||U||_F^2||V||_F^2. \qquad (4.3)$$

4.1.4 Optimization Problem

Considering both the approximation $X \approx U\Sigma V^T$ and the sensitivity analysis, a factorization problem can be cast as an optimization problem.

$$\begin{aligned} \min_{U,\Sigma,V} & \sum_{(i,j)\in OI} (X_{ij} - (U\Sigma V^T)_{ij})^2 \\ + & \quad \lambda \left(n||\Sigma V^T||_F^2 + m||U\Sigma||_F^2 + d||U||_F^2||V||_F^2 \right), \\ s.t. & \quad U \geq 0, \\ & \quad \Sigma \geq 0. \end{aligned} \tag{4.4}$$

where λ is a hyperparameter that controls the balance between the approximation and the sensitivity, and OI denote the set of observed index pairs.

4.1.5 Problem Simplification and Solution

Let U_i's and V_i's be the columns of U and V respectively. Without loss of generality, we set $||U_k||_F = 1, ||V_k||_F = 1$ for $1 \leq k \leq d$. As a result, $||U||_F^2 = d, ||V||_F^2 = d$. For the purpose of simplifying the solution, we further assume that $\Sigma_{d\times d}$ is diagonal, i.e., $\Sigma_{d\times d} = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_d)$. Consequently,

$$||\Sigma V^T||_F^2 = \sum_{k=1}^{d} \lambda_k^2,$$

and

$$||U\Sigma||_F^2 = \sum_{k=1}^{d} \lambda_k^2.$$

In order to simplify the notation, we denote $U\Sigma$ as U, then Σ disappears, and the conditions $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$ can be changed to $||U_1||_F \geq ||U_2||_F \geq \ldots \geq ||U_d||_F$. Based on the above simplification, Eq. (4.4) can be reformulated as follows.

Given an $n \times m$ nonnegative matrix X, solve

$$\begin{aligned} \min_{U_k,V_k} \quad & \sum_{(i,j)\in OI} \left(X_{ij} - \sum_{k=1}^{d} (U_k V_k^T)_{ij} \right)^2 \\ + \quad & \lambda n \sum_{k=1}^{d} ||U_k||_F^2 + \lambda m \sum_{k=1}^{d} ||V_k||_F^2 + \lambda d^3, \\ s.t. \quad & U_k \geq 0, \\ & ||U_k||_F \geq ||U_{k+1}||_F, \\ & ||V_k||_F = 1. \end{aligned} \tag{4.5}$$

In order to obtain the most informative latent features and find the dimension d, we fit the incomplete matrix X step by step in such a way that when U_k and V_k are learned, U_j $(j \leq k-1)$ and V_j $(j \leq k-1)$ are fixed, and we only learn U_k and V_k based on the residual R. R is defined as

$$R = X - \sum_{j=1}^{k-1} U_j V_j^T$$

on OI, and $R = 0$ on others for convenience. The process continues until there is no useful information retained in R. When the process stops, the dimension can be determined. So we only focus on the following problem:

$$\begin{aligned} \min_{U_k,V_k} \quad & \sum_{(i,j)\in OI} \left(R_{ij} - (U_k V_k^T)_{ij} \right)^2 \\ + \quad & \lambda n ||U_k||_F^2 + \lambda m ||V_k||_F^2, \\ s.t. \quad & U_k \geq 0, \\ & ||U_{k-1}||_F \geq ||U_k||_F, \\ & ||V_k||_F = 1. \end{aligned} \tag{4.6}$$

Note that the elements in R may be negative. If we ignore the

variant λ_k, the Lagrangian of the above problem is

$$\begin{aligned} J &= \sum_{(i,j)\in OI} \left(R_{ij} - (U_k V_k^T)_{ij}\right)^2 \\ &+ \lambda(m+n)||U_k||_F^2 \\ &+ \mu_k(U_k^T U_k - U_{k-1}^T U_{k-1}) \\ &+ \nu_k(V_k^T V_k - 1) - Y^T U_K, \end{aligned} \tag{4.7}$$

where $Y \in R_+^n$, and $\mu_k \in R_+$. Let the i-th element of U_k, the j-th element of V_k, and the i-th element of Y be U_{ki}, V_{kj} and Y_i respectively. In order to solve this problem, take derivative on J with respect to U_{ki} and V_j. We have

$$\begin{aligned} \frac{\partial J}{\partial U_{ki}} &= \sum_{j:(i,j)\in OI} 2(R_{ij} - U_{ki}V_{kj})(-V_{kj}) \\ &+ 2\mu_k U_{ki} - Y_i = 0, \end{aligned} \tag{4.8}$$

$$\begin{aligned} \frac{\partial J}{\partial V_{kj}} &= \sum_{i:(i,j)\in OI} 2(R_{ij} - U_{ki}V_{kj})(-U_{ki}) \\ &+ 2\nu_k V_{kj} = 0. \end{aligned} \tag{4.9}$$

If U_k is given, then minimizing the quadratic function in Eq. (4.7), we obtain that

$$V_{kj} = \frac{\sum_{i:(i,j)\in OI} R_{ij}U_{ki}}{\sum_{i:(i,j)\in OI} U_{ki}^2 + \nu_k}, \tag{4.10}$$

where ν_k is a parameter such that $||V_k||_F = 1$.

If V_k is given, considering the constraints that $U_k \geq 0$ and $||U_{k-1}||_F \geq ||U_k||_F$, we obtain

$$\begin{aligned} U_{ki} &= \frac{\sum_{j:(i,j)\in OI} R_{ij}V_{kj} + Y_i/2}{\sum_{j:(i,j)\in OI} V_{kj}^2 + \mu_k} \\ &= \frac{(\sum_{j:(i,j)\in OI} R_{ij}V_{kj})_+}{\sum_{j:(i,j)\in OI} V_{kj}^2 + \mu_k}, \end{aligned} \tag{4.11}$$

where Y_i is the minimum positive number such that

$$\sum_{j:(i,j)\in OI} R_{ij}V_{kj} + Y_i/2 \geq 0,$$

i.e.,

$$Y_i = 0 \text{ if } \sum_{j:(i,j)\in OI} R_{ij}V_{kj} \geq 0,$$

and

$$Y_i = -\sum_{j:(i,j)\in OI} R_{ij}V_{kj} \text{ if } \sum_{j:(i,j)\in OI} R_{ij}V_{kj} < 0,$$

and μ_k is the minimum positive number such that

$$||U_k||_F \leq ||U_{k-1}||_F .$$

We name our algorithm as Semi-Nonnegative Matrix Factorization with Global Statistical Consistency (SNGSC). In Algorithm 1, we summarize a learning algorithm by employing Eq. (4.10) and Eq. (4.11). The criterion that no useful information can be mined in R is specified in our experiments as: the difference between the mean residual $\frac{1}{|OI|}\sum_{(i,j)\in OI}|R_{ij}|$ in the current dimension d and that in the previous dimension is smaller than 0.0005.

From the algorithm, we can see the time complexity of SNGSC is linear on the number of ratings, i.e., O($|OI|$), because we only need to calculate the multiplications when the ratings values are not missing. Moreover, with the proper physical meaning in U and V, our algorithm is expected to achieve more accurate results.

Algorithm 1: SNGSC Learning Algorithm

Input: Incomplete matrix $X \geq 0$

Output: d, $\{U_k\}_{k=1}^{d}$, and $\{V_k\}_{k=1}^{d}$

Initialize $d = 0$, $k = 1$.
repeat
  if $k == 1$ **then**
    $R = X$
  else
    $R = R - U_{k-1}V_{k-1}^T$
  end if
  repeat
    for $j = 1$ TO m **do**
      $V_{kj} = \frac{\sum_{i:(i,j)\in OI} R_{ij}U_{ki}}{\sum_{i:(i,j)\in OI} U_{ki}^2 + \nu_k}$
    end for
    for $i = 1$ TO n **do**
      $U_{ki} = \frac{(\sum_{j:(i,j)\in OI} R_{ij}V_{kj})_+}{\sum_{j:(i,j)\in OI} V_{kj}^2 + \mu_k}$
    end for
  until Converge
  $k = k + 1$
until No useful information can be mined in R
$d = k - 1$

4.2 Consistency with Global Information

Until now, we only constrain the expression $\sum_{k=1}^{d}(U_k V_k^T)$ in Eq. (4.5) by fitting its values on the user-item pairs with the training data. However, we observe that this partial constraint cannot make the values $\sum_{k=1}^{d}(U_k V_k^T)$ follow the global statistics such as the first moment and the second moment. The previous low-dimensional factor models share this problem because no action is taken on controlling the global statistics. For example, the mean of ratings in EachMovie Data is 0.607357 (after scaling

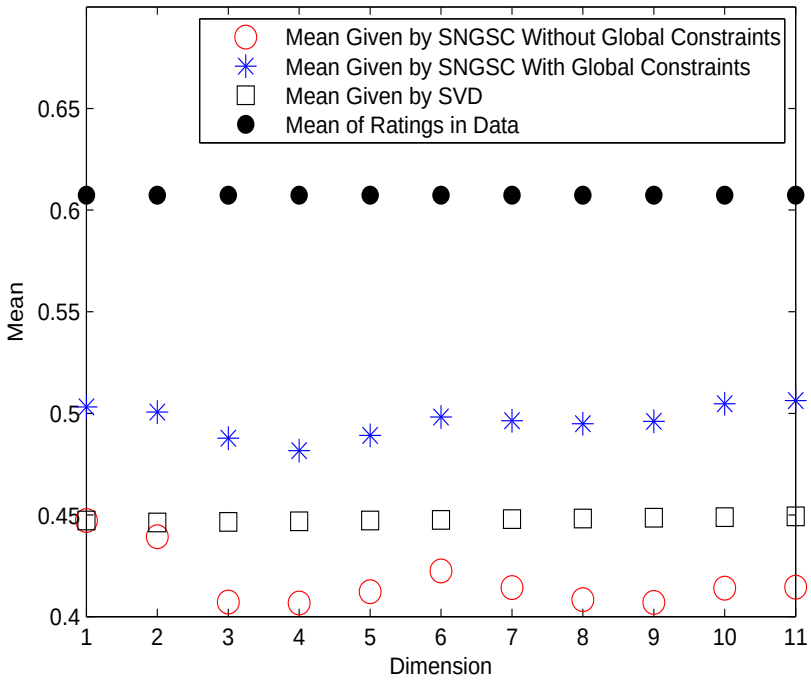


Figure 4.1: An illustration showing the problem of SNGSC and SVD without controlling the global statistics. The means predicted by models are far away from the true means.

to the interval [0,1]), but the mean given by SVD and SNGSC is far away from the true mean. In Figure 4.1, we demonstrate this problem.

Based on the above observation, we propose to impose the consistency on SNGSC between the predicted statistics and those given in the data samples. Ideally we should consider moments of all orders and the data priors, but considering the computation cost and the model complexity, we only include the first moment $\bar{X}$–the mean of ratings in this chapter. The predicted values are given by $\sum_{k=1}^{d}(U_k V_k^T)$, and hence the predicted mean by the model is

$$\frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{d}(U_{ki}V_{kj}) = \sum_{k=1}^{d}(\bar{U}_k\bar{V}_k^T),$$

where $\bar{U}_k$ and $\bar{V}_k$ are the vector means of U_k and V_k respectively.

Let η be the parameter balancing the tradeoff of fitting the data and fitting the mean of ratings. Then we should optimize

$$\begin{aligned}
\min_{U_k,V_k} & \sum_{(i,j)\in OI} \left(R_{ij} - (U_k V_k^T)_{ij}\right)^2 \\
+ \quad & \lambda n ||U_k||_F^2 + \lambda m ||U_k||_F^2 \\
+ \quad & \eta \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} (\sum_{l=1}^{k} U_{li} V_{lj} - \bar{X})^2, \\
s.t. \quad & U_k \geq 0, \\
& ||U_{k-1}||_F \geq ||U_k||_F, \\
& ||V_k||_F = 1. \qquad (4.12)
\end{aligned}$$

When $\eta = 0$, no global information is included; when $\eta = +\infty$, all the predicted values $\sum_{l=1}^{k} U_{li} V_{lj}$ will be equal to $\bar{X}$ such that the first moment is perfectly fitted. The best η should be in the middle of these two extreme cases. In our experiments, we set $\eta = \sqrt{nm}/|OI|$ based on experiences. An ordinary calculus can result in similar equations as Eq. (4.10) and Eq. (4.11).

4.3 Experiments

In this section, we conduct several experiments to compare the recommendation quality of our approach with other state-of-the-art collaborative filtering methods. Our experiments are intended to address the following questions:

1. How does our approach compare with the published state-of-the-art collaborative filtering algorithms?

2. How does the model parameter η (the global consistency parameter) affect the accuracy of the prediction?

3. How do the non-negative constraints affect the accuracy of the recommendation quality?

4. What is the performance comparison on users with different observed ratings?

4.3.1 Description of Dataset

We evaluate our algorithms on the EachMovie dataset[1], which is commonly used in previous work [75, 94, 133]. The EachMovie dataset contains 74,424 users, 1,648 movies, and 2,811,718 ratings in the scale of zero to five. We map the ratings 0,1,2,3,4 and 5 to the interval $[0, 1]$ using the linear function $t(x) = x/5$.

As to the training data, we employ three settings: 80%, 50% and 20% for training, where 80% means we randomly select 80% ratings as training data to predict the remaining 20% ratings. Selecting 80% as training data is the standard evaluation setting which is widely employed in the previous work. However, in this chapter, we are also interested in the settings to include 50% and 20% as training data, since these two settings can be used to examine how well the algorithms are under the sparse data settings. The reported results in all of the experiments in this chapter are the average of ten runs of the algorithms on the ten random partitions of the dataset.

4.3.2 Metrics

We use the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) metrics to measure the prediction quality of our proposed approach in comparison with other collaborative filtering methods. MAE is defined as:

$$MAE = \frac{\sum_{i,j} |r_{i,j} - \widehat{r}_{i,j}|}{N}, \tag{4.13}$$

where $r_{i,j}$ denotes the rating user i gave to item j, $\widehat{r}_{i,j}$ denotes the rating user i gave to item j as predicted by our approach,

[1] http://www.research.digital.com/SRC/EachMovie/. It is retired by Hewlett-Packard (HP).

and N denotes the number of tested ratings. RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum_{i,j}(r_{i,j} - \widehat{r}_{i,j})^2}{N}}. \tag{4.14}$$

4.3.3 Performance Comparisons

We compare our SNGSC approach with other four approaches.

1. **User Mean**: This is a baseline method which predicts a user's missing rating on an item by the sample mean of this user's ratings.

2. **Item Mean**: This is a baseline method which predicts a user's missing rating on an item by the sample mean of this item's ratings.

3. **MMMF** [94, 114]: This method constrains the norms of U and V instead of their dimensionality. This corresponds to constraining the overall "strength" of the factors, rather than their number.

4. **PMF** [99]: This method proposes a probabilistic framework to employ $U_i^T V_j$ with Gaussian noise fitting each rating observation.

The prediction accuracies evaluated by Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are shown in Table 4.3. In SNGSC, the parameter λ is set to be 0.000004, and the parameter η is set to be $\sqrt{nm}/|OI|$, where $|OI|$ is the number of observed ratings. The dimensions for SNGSC are automatically determined at each of the ten runs, and they are between 25 and 30. In order to compare other algorithms fairly, we set the dimensions of MMMF and PMF to 30.

From Table 4.3, we can observe that our algorithm consistently performs better than the other methods in all the settings. When we use a sparse dataset (20% as training data), we

Table 4.3: Comparison with other popular algorithms. The reported values are the mean RMSE and MAE on the EachMovie Dataset achieved by ten runs from dividing the data into 80%, 50%, and 20% for training data, respectively.

Data	Metrics	User Mean	Item Mean	MMMF	PMF	SNGSC
80%	**RMSE**	1.426	1.386	1.173	1.151	**1.122**
	Variance	$\leq 10^{-4}$	$\leq 10^{-4}$	≤ 0.001	≤ 0.001	$\leq 10^{-5}$
	MAE	1.141	1.102	0.928	0.901	**0.860**
	Variance	$\leq 10^{-4}$	$\leq 10^{-4}$	$\leq$0.001	$\leq$0.001	$\leq 10^{-5}$
50%	**RMSE**	1.438	1.387	1.342	1.335	**1.176**
	Variance	$\leq 10^{-4}$	$\leq 10^{-4}$	≤ 0.001	≤ 0.001	$\leq 10^{-5}$
	MAE	1.149	1.103	0.978	0.963	**0.891**
	Variance	$\leq 10^{-4}$	$\leq 10^{-4}$	$\leq$0.001	$\leq$0.001	$\leq 10^{-5}$
20%	**RMSE**	1.484	1.388	1.466	1.451	**1.266**
	Variance	≤ 0.001	≤ 0.001	≤ 0.01	≤ 0.01	$\leq 10^{-4}$
	MAE	1.180	1.103	1.143	1.085	**0.973**
	Variance	≤ 0.001	≤ 0.001	$\leq$0.01	$\leq$0.01	$\leq 10^{-4}$

find that our method generates much better performance than MMMF and PMF. However, MMMF and PMF do not address the problem of sparsity, hence they even perform worse than the Item Mean method when using 20% as training data. This demonstrates the advantage of our algorithm in handling the sparsity problem.

In Figure 4.2 and Figure 4.3, we also plot the percentages of performance increase of our algorithm against other four methods in terms of RMSE and MAE on the EachMovie dataset, respectively. From these figures, we observe an interesting phenomenon: as the sparsity of the data increases, the percentages of performance increase against MMMF and PMF keep increasing. This observation again proves the advantage of our algo-

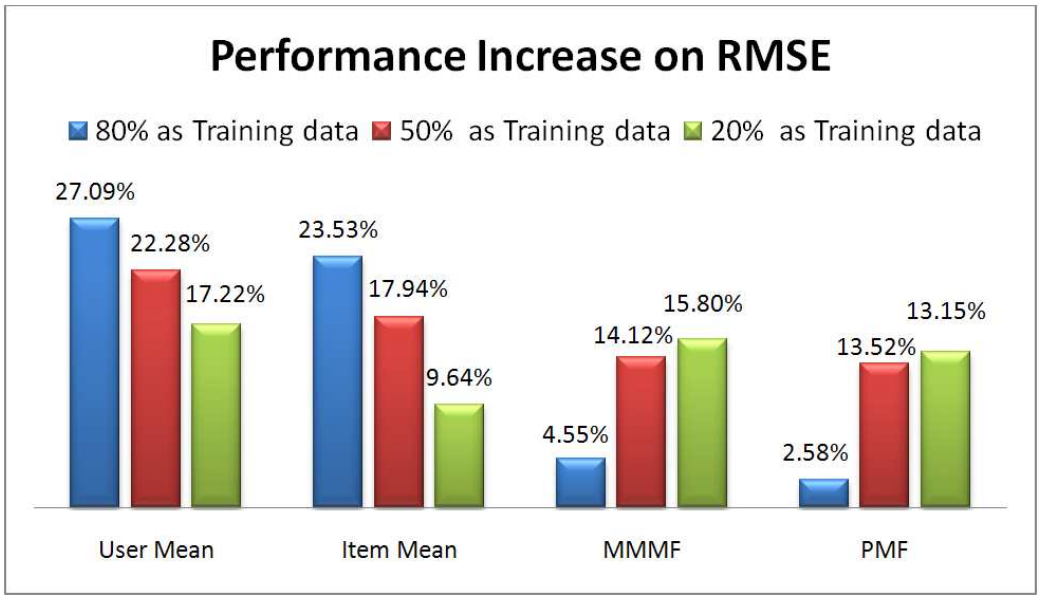


Figure 4.2: Performance Increase on RMSE (EachMovie)

rithm. On the other hand, we can also notice that as the sparsity increases, although our method still can generates much better recommendation qualities than User Mean and Item Mean methods, the percentages of performance increase against these two methods keep dropping. This observation is reasonable because our random testing data generation method does not change the distribution of the ratings. Hence, the User Mean and Item Mean algorithms should be relatively stable against the sparsity problem.

In order to show the usefulness of each key part of SNGSC, we also evaluate our algorithm on its various degraded cases as follows:

1. SNGSC-1: It is the SNGSC algorithm without the global consistency ($\eta = 0$);

2. SNGSC-2: It is the SNGSC algorithm without the non-negative constraint (a modified version of SVD with global consistency);

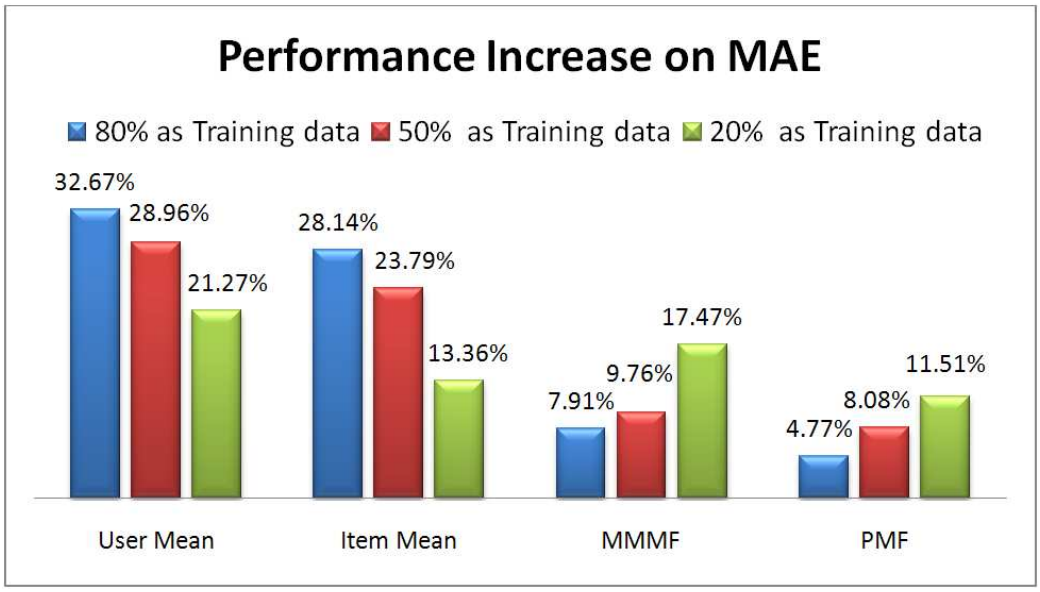


Figure 4.3: Performance Increase on MAE (EachMovie)

3. SNGSC-3: It is the SNGSC algorithm with nonnegative constraints on both U and V (a modified version of NMF with global consistency).

The results on the EachMovie dataset are reported in Table 4.4. From the results, we observe that our Semi-Nonnegative setting is the best among all these variants, which empirically demonstrates the need of introducing SNMF.

However, the global consistency achieves only a little accuracy improvement in this experimental setting (See SNGSC-1 and SNGSC). This phenomenon may be caused by the setting that majority (80%) of data is chosen as training data. In the extreme case that the rating data is very sparse and each user only rates one movie, then the latent features U and V do not have much meanings, but we can at least predict all the missing ratings as the mean of training data. We believe that the sparser the training data, the better the global consistency approach. To demonstrate the effectiveness of the global consistency approach, we run both SNGSC-1 and SNGSC in a different setting: 20%

Table 4.4: Comparison with variants of SNGSC in a setting with 80% for training and 20% for testing on the EachMovie dataset. (1) SNGSC-1: SNGSC without the global consistency ($\eta = 0$); (2) SNGSC-2: SNGSC without the nonnegative constraint (a modified version of SVD with global consistency); and (3) SNGSC-3: SNGSC with nonnegative constraints on both U and V (a modified version of NMF with global consistency).

Algorithms	**SNGSC-1**	**SNGSC-2**	**SNGSC-3**	**SNGSC**
RMSE	1.151	1.212	1.258	1.122
Variance	$\leq 10^{-5}$	$\leq$0.001	$\leq$0.001	$\leq 10^{-5}$
MAE	0.883	0.932	0.971	0.860
Variance	$\leq 10^{-5}$	$\leq$0.001	$\leq$0.001	$\leq 10^{-5}$

Table 4.5: Comparison with variants of SNGSC in a 20% for training 80% for testing setting on the EachMovie dataset.

Algorithms	**SNGSC-1**	**SNGSC-2**	**SNGSC-3**	**SNGSC**
RMSE	1.423	1.356	1.365	1.266
Variance	$\leq 10^{-4}$	$\leq$0.01	$\leq$0.01	$\leq 10^{-4}$
MAE	1.095	1.048	1.060	0.973
Variance	$\leq 10^{-4}$	$\leq$0.01	$\leq$0.01	$\leq 10^{-4}$

of the data are chosen for training and 80% for testing. The results are shown in Table 4.5. From the results, we can see SNGSC with the global consistency significantly outperforms the one without the global consistency (SNGSC-1). In such a setting, it is not surprising to see that the difference between SNGSC and SNGSC-2 is small, because the latent feature is not very meaningful and hence the sign setting is not so important; therefor, the global consistency dominates the results.

4.4 Summary

We demonstrate a Semi-Nonnegative Matrix Factorization method with Global Statistical Consistency for collaborative filtering, in which the user-specific latent feature U_{ik} includes the meaning of

the confidence of user i on the k-th latent type of the item, and the item-specific latent feature V_{jk} includes the meaning of the quality of the item j on the k-th latent type of the item. This work has showed that the latent features with physical meanings can achieve not only the model interpretability but also the prediction accuracy. Moreover, we propose a novel method that imposes the consistency between the statistics of training data and the statistics of the predicted ratings. The experimental analysis shows that our method outperforms other state-of-the-art algorithms.

For the global consistency, we only take the first step, i.e., we only make our models consistent with the first moment currently. By doing so we have already achieved promising results. In order to capitalize on these achievements, further study is needed on the following problems:

1. We would enforce the consistency with the second moment globally in the models without increasing the complexity of our models.

2. There is prior information that all values in the matrix $\sum_{k=1}^{d}(U_k V_k^T)$ should be between zero and one after the mapping. Without taking any action, prediction by $\sum_{k=1}^{d} U_k V_k^T$ will run outside of the range of valid rating values. For this, one choice is to map the values to the interval $[0, 1]$ by some nonlinear functions like logistic function. But in our setting, such a mapping does not match our intuition–the prediction on the user-item pair (i, j) results from a linear combination of the products of i's authority on a latent type and j's quality. For such a consideration, how can we put a constraint that $0 \leq \sum_{k=1}^{d}(U_k V_k^T) \leq 1$ while we can still learn the latent features dimension by dimension.

□ **End of chapter.**

Chapter 5

Social Recommendation

Traditional recommender systems assume that users are *i.i.d.* (independent and identically distributed); this assumption ignores the social interactions or connections among users. But the fact is, offline, social recommendation is an everyday occurrence. In order to reflect users' social relations in the recommendations, based on the intuition that a user's social network will affect her/his personal behaviors on the Web, in this chapter, we propose to fuse a user's social network graph with the user-item rating matrix in order to make more accurate and personalized recommendations, which is called *Social Recommendation*.

5.1 Recommendation Framework

In this section, we first design a recommendation framework by consolidating user-item rating matrix and users' social trust network in Section 5.1.1. Then in Section 5.1.2, we apply this framework to incorporating social tag information, which is another important source of social contextual information.

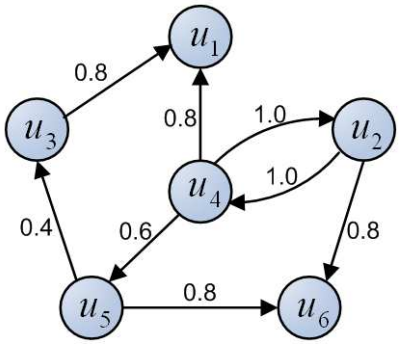
5.1.1 Recommendation with Social Trust Network

We first demonstrate our recommendation framework using a simple but illustrative toy example. Then we introduce the rec-

ommendation framework by factor analysis using probabilistic matrix factorization.

A Toy Example

Let us first consider the typical social trust network graph in Fig. 5.1(a). There are 6 users in total (nodes, from u_1 to u_6) with 8 relations (edges) between users in this graph, and each relation is associated with a weight w_{ij} in the range $[0, 1]$ to specify how much user u_i knows or trusts user u_j. In an online social network Web site, the weight w_{ij} is often explicitly stated by user u_i. As illustrated in Fig. 5.1(b), each user also rates some items (from i_1 to i_8) on a 5-point integer scale to express the extent of favor of each item. The problem we study in this chapter is how to predict the missing values of the user-item matrix effectively and efficiently by employing two different data sources. Motivated by the intuition that a user's social trust connections will affect this user's behaviors on the Web, we therefore factorize the social trust graph and user-item matrix simultaneously and seamlessly using $U^T Z$ and $U^T V$, where the shared low-dimensional matrix U denotes the user latent feature space, Z is the factor matrix in the social network graph, and V represents the low-dimensional item latent feature space. If we use 5 dimensions to perform the matrix factorization for social recommendation, we obtain



(a) Social Network Graph

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	5	2		3		4		
u_2	4	3			5			
u_3	4		2				2	4
u_4								
u_5	5	1	2		4	3		
u_6	4	3		2	4		3	5

(b) User-Item Matrix

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	5	2	**2.5**	3	**4.8**	4	**2.2**	**4.8**
u_2	4	3	**2.4**	**2.9**	5	**4.1**	**2.6**	**4.7**
u_3	4	**1.7**	2	**3.2**	**3.9**	**3.0**	2	4
u_4	**4.8**	**2.1**	**2.7**	**2.6**	**4.7**	**3.8**	**2.4**	**4.9**
u_5	5	1	2	**3.4**	4	3	**1.5**	**4.6**
u_6	4	3	**2.9**	2	4	**3.4**	3	5

(c) Predicted User-Item Matrix

Figure 5.1: Example for Toy Data

$$U = \begin{bmatrix} 1.55 & 1.22 & 0.37 & 0.81 & 0.62 & -0.01 \\ 0.36 & 0.91 & 1.21 & 0.39 & 1.10 & 0.25 \\ 0.59 & 0.20 & 0.14 & 0.83 & 0.27 & 1.51 \\ 0.39 & 1.33 & -0.43 & 0.70 & -0.90 & 0.68 \\ 1.05 & 0.11 & 0.17 & 1.18 & 1.81 & 0.40 \end{bmatrix},$$

$$V = \begin{bmatrix} 1.00 & -0.05 & -0.24 & 0.26 & 1.28 & 0.54 & -0.31 & 0.52 \\ 0.19 & -0.86 & -0.72 & 0.05 & 0.68 & 0.02 & -0.61 & 0.70 \\ 0.49 & 0.09 & -0.05 & -0.62 & 0.12 & 0.08 & 0.02 & 1.60 \\ -0.40 & 0.70 & 0.27 & -0.27 & 0.99 & 0.44 & 0.39 & 0.74 \\ 1.49 & -1.00 & 0.06 & 0.05 & 0.23 & 0.01 & -0.36 & 0.80 \end{bmatrix},$$

where U_i and V_j are the column vectors and denote the latent feature vectors of user u_i and item v_j, respectively. Note that the solutions of U and V are not unique. Then we can predict the missing value w_{ij} in Fig. 5.1(b) using $U_i^T V_j$ (before prediction, we need to first transfer the value of $U_i^T V_j$ using logistic function $g(x)$ and another mapping function $f(x)$, which will be introduced in Section 5.1.1 and Section 5.1.1 respectively). Therefore, all the missing values can be predicted using 5-dimensional matrices U and V, as shown in Fig. 5.1(c). Note that even though user u_4 does not rate any items, our approach still can predict reasonable ratings.

Since this example is a toy example, we cannot evaluate the accuracy of the prediction. However, the experimental analysis in Section 5.2 based on Epinions dataset tests the effectiveness of our approach. In the following sections, we will present the details of how we conduct factor analysis for social recommendation using probabilistic matrix factorization.

Social Network Matrix Factorization

Suppose we have a directed social network graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the vertex set $\mathcal{V} = \{v_i\}_{i=1}^{n}$ represents all the users in a social network and the edge set $\mathcal{E}$ represents the relations between users. Let $C = \{c_{ik}\}$ denote the $m \times m$ matrix of $\mathcal{G}$, which is also called the social network matrix in this chapter. For a pair of vertices, v_i and v_k, let $c_{ik} \in (0, 1]$ denote the weight associated with an edge from v_i to v_k, and $c_{ik} = 0$, otherwise. The physical meaning of the weight c_{ik} can be interpreted as how much a user i trusts or knows user k in a social network. Note that C is an asymmetric matrix, since in a social network, especially in a trust-based social network, user i trusting k does not necessary indicate user k trusts i.

The idea of social network matrix factorization is to derive a high-quality l-dimensional feature representation U of users based on analyzing the social network graph $\mathcal{G}$. Let $U \in R^{l \times m}$ and $Z \in R^{l \times m}$ be the latent user and factor feature matrices, with column vectors U_i and Z_k representing user-specific and factor-specific latent feature vectors, respectively. We define the conditional distribution over the observed social network relationships as

$$p(C|U, Z, \sigma_C^2) = \prod_{i=1}^{m} \prod_{k=1}^{m} \mathcal{N}\left[\left(c_{ik}|g(U_i^T Z_k), \sigma_C^2\right)\right]^{I_{ik}^C}, \tag{5.1}$$

where $\mathcal{N}(x|\mu, \sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and variance σ_C^2, and I_{ik}^C is the indicator function that is equal to 1 if user i trusts or knows user k and equal to 0 otherwise. The function $g(x)$ is the logistic function $g(x) = 1/(1 + \exp(-x))$, which makes it possible to bound the range of $U_i^T Z_k$ within the range $[0, 1]$. We also place zero-mean spherical Gaussian priors [31, 99] on user and factor

feature vectors:

$$p(U|\sigma_U^2) = \prod_{i=1}^{m} \mathcal{N}(U_i|0, \sigma_U^2 \mathbf{I}),$$
$$p(Z|\sigma_Z^2) = \prod_{k=1}^{m} \mathcal{N}(Z_k|0, \sigma_Z^2 \mathbf{I}). \tag{5.2}$$

Hence, through a simple Bayesian inference, we have

$$\begin{aligned} & p(U, Z|C, \sigma_C^2, \sigma_U^2, \sigma_Z^2) \\ \propto\ & p(C|U, Z, \sigma_C^2) p(U|\sigma_U^2) p(Z|\sigma_Z^2) \\ =\ & \prod_{i=1}^{m} \prod_{k=1}^{m} \mathcal{N}\left[\left(c_{ik}|g(U_i^T Z_k), \sigma_C^2\right)\right]^{I_{ik}^C} \\ \times\ & \prod_{i=1}^{m} \mathcal{N}(U_i|0, \sigma_U^2 \mathbf{I}) \times \prod_{k=1}^{m} \mathcal{N}(Z_k|0, \sigma_Z^2 \mathbf{I}). \end{aligned} \tag{5.3}$$

In online social networks, the value of c_{ik} is mostly explicitly stated by user i with respect to user k, which cannot accurately describe the relations between users since it contains noises and it ignores the graph structure information of social network. For instance, similar to the Web link adjacency graph in [130], in a trust-based social network, the confidence of trust value c_{ik} should be decreased if user i trusts a large number of users; however, the confidence of trust value c_{ik} should be increased if user k is trusted by lots of users. Hence, we employ the term c_{ik}^* which incorporates local authority and local hub values as a substitute for c_{ik} in Eq. (5.1),

$$p(C|U, Z, \sigma_C^2) = \prod_{i=1}^{m} \prod_{j=1}^{n} \mathcal{N}\left[\left(c_{ik}^*|g(U_i^T Z_k), \sigma_C^2\right)\right]^{I_{ik}^C},$$
$$c_{ik}^* = \sqrt{\frac{d^-(v_k)}{d^+(v_i) + d^-(v_k)}} \times c_{ik}, \tag{5.4}$$

where $d^+(v_i)$ represents the outdegree of node v_i, while $d^-(v_k)$ indicates the indegree of node v_k.

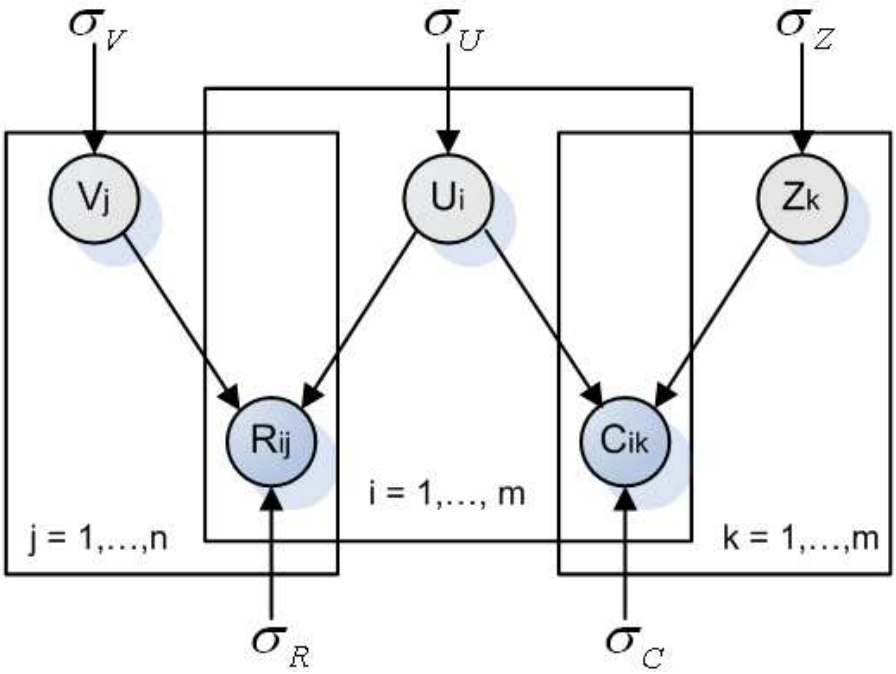


Figure 5.2: Graphical Model for Social Trust Recommendation

User-Item Matrix Factorization

Now considering the user-item matrix, suppose we have m users, n movies, and rating values within the range $[0, 1]$. Actually, most recommender systems use integer rating values from 1 to R_{max} to represent the users' judgements on the items. In this chapter, without loss of generality, we map the ratings $1, ..., R_{max}$ to the interval $[0, 1]$ using the function $f(x) = (x-1)/(R_{max}-1)$. Let R_{ij} represent the rating of user i for movie j, and $U \in R^{l \times m}$ and $V \in R^{l \times n}$ be latent user and movie feature matrices, with column vectors U_i and V_j representing user-specific and movie-specific latent feature vectors respectively. We define the conditional distribution over the observed ratings as

$$p(C|U, V, \sigma_R^2) = \prod_{i=1}^{m} \prod_{j=1}^{n} \mathcal{N}\left[\left(r_{ij}|g(U_i^T V_j), \sigma_R^2\right)\right]^{I_{ij}^R}, \qquad (5.5)$$

where I_{ij}^R is the indicator function that is equal to 1 if user i rated movie j and equal to 0 otherwise. We also place zero-mean spherical Gaussian priors on user and movie feature vectors:

$$p(U|\sigma_U^2) = \prod_{i=1}^{m} \mathcal{N}(U_i|0, \sigma_U^2 \mathbf{I}),$$
$$p(V|\sigma_V^2) = \prod_{j=1}^{n} \mathcal{N}(V_j|0, \sigma_V^2 \mathbf{I}). \tag{5.6}$$

Hence, similar to Eq. (5.3), through a Bayesian inference, we have

$$\begin{aligned} & p(U, V|R, \sigma_R^2, \sigma_U^2, \sigma_V^2) \\ & \quad \propto \; p(R|U, V, \sigma_R^2) p(U|\sigma_U^2) p(Z|\sigma_V^2) \\ & \quad = \; \prod_{i=1}^{m} \prod_{j=1}^{n} \mathcal{N}\left[\left(r_{ij}|g(U_i^T V_j), \sigma_R^2\right)\right]^{I_{ij}^R} \\ & \quad \times \; \prod_{i=1}^{m} \mathcal{N}(U_i|0, \sigma_U^2 \mathbf{I}) \times \prod_{j=1}^{n} \mathcal{N}(V_j|0, \sigma_V^2 \mathbf{I}). \end{aligned} \tag{5.7}$$

Matrix Factorization for Social Trust Recommendation

In order to reflect the phenomenon that a user's social connections will affect this user's judgement of interest in items, we model the problem of social recommendation using the graphical model described in Fig. 5.2, which fuses both the social network graph and the user-item rating matrix into a consistent and compact feature representation.

Based on Fig. 5.2, we have

$$\begin{aligned} & p(U, V, Z|C, R, \sigma_C^2, \sigma_R^2, \sigma_U^2, \sigma_V^2, \sigma_Z^2) \\ & \quad \propto \; p(R|U, V, \sigma_R^2) p(C|U, Z, \sigma_C^2) \\ & \quad \times \; p(U|\sigma_U^2) p(V|\sigma_V^2) p(Z|\sigma_Z^2). \end{aligned} \tag{5.8}$$

The log of the posterior distribution for the above equation is given by

$$\begin{aligned}
&\ln p(U,V,Z|C,R,\sigma_C^2,\sigma_R^2,\sigma_U^2,\sigma_V^2,\sigma_Z^2) = \\
&-\frac{1}{2\sigma_R^2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^R (r_{ij} - g(U_i^T V_j))^2 \\
&-\frac{1}{2\sigma_C^2}\sum_{i=1}^{m}\sum_{k=1}^{m} I_{ik}^C (c_{ik}^* - g(U_i^T Z_k))^2 \\
&-\frac{1}{2\sigma_U^2}\sum_{i=1}^{m} U_i^T U_i - \frac{1}{2\sigma_V^2}\sum_{j=1}^{n} V_j^T V_j - \frac{1}{2\sigma_Z^2}\sum_{k=1}^{m} Z_k^T Z_k \\
&-\frac{1}{2}\left(\left(\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^R\right)\ln\sigma_R^2 + \left(\sum_{i=1}^{m}\sum_{k=1}^{m} I_{ik}^C\right)\ln\sigma_C^2\right) \\
&-\frac{1}{2}\left(ml\ln\sigma_U^2 + nl\ln\sigma_V^2 + ml\ln\sigma_Z^2\right) + \mathcal{C}, \qquad (5.9)
\end{aligned}$$

where $\mathcal{C}$ is a constant that does not depend on the parameters. Maximizing the log-posterior over three latent features with hyperparameters (i.e., the observation noise variance and prior variances) kept fixed is equivalent to minimizing the following sum-of-squared-errors objective functions with quadratic regularization terms:

$$\begin{aligned}
\mathcal{L}(R,C,U,V,Z) \;=\; & \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^R (r_{ij} - g(U_i^T V_j))^2 \\
+\; & \frac{\lambda_C}{2}\sum_{i=1}^{m}\sum_{k=1}^{m} I_{ik}^C (c_{ik}^* - g(U_i^T Z_k))^2 \\
+\; & \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2 + \frac{\lambda_Z}{2}\|Z\|_F^2, \qquad (5.10)
\end{aligned}$$

where $\lambda_C = \sigma_R^2/\sigma_C^2$, $\lambda_U = \sigma_R^2/\sigma_U^2$, $\lambda_V = \sigma_R^2/\sigma_V^2$, $\lambda_Z = \sigma_R^2/\sigma_Z^2$, and $\|\cdot\|_F^2$ denotes the Frobenius norm. A local minimum of the

objective function given by Eq.(5.10) can be found by performing gradient descent in U_i, V_j and Z_k,

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial U_i} &= \sum_{j=1}^{n} I_{ij}^R g'(U_i^T V_j)(g(U_i^T V_j) - r_{ij})V_j \\
&+ \lambda_C \sum_{j=1}^{m} I_{ik}^C g'(U_i^T Z_k)(g(U_i^T Z_k) - c_{ik}^*)Z_k + \lambda_U U_i, \\
\frac{\partial \mathcal{L}}{\partial V_j} &= \sum_{i=1}^{m} I_{ij}^R g'(U_i^T V_j)(g(U_i^T V_j) - r_{ij})U_i + \lambda_V V_j, \\
\frac{\partial \mathcal{L}}{\partial Z_k} &= \lambda_C \sum_{i=1}^{m} I_{ik}^C g'(U_i^T Z_k)(g(U_i^T Z_k) - c_{ik}^*)U_i + \lambda_Z Z_k, \qquad (5.11)
\end{aligned}$$

where $g'(x)$ is the derivative of logistic function $g'(x) = \exp(x)/(1+\exp(x))^2$. In order to reduce the model complexity, in all of the experiments we conduct in Section 5.2, we set $\lambda_U = \lambda_V = \lambda_Z$.

Complexity Analysis

The main computation of gradient methods is evaluating the object function $\mathcal{L}$ and its gradients against variables. Because of the sparsity of matrices R and C, the computational complexity of evaluating the object function $\mathcal{L}$ is $O(\rho_R l + \rho_C l)$, where ρ_R and ρ_C are the numbers of nonzero entries in matrices R and C, respectively. The computational complexities for gradients $\frac{\partial \mathcal{L}}{\partial U}$, $\frac{\partial \mathcal{L}}{\partial V}$ and $\frac{\partial \mathcal{L}}{\partial Z}$ in Eq. (5.11) are $O(\rho_R l + \rho_C l)$, $O(\rho_R l)$ and $O(\rho_C l)$, respectively. Therefore, the total computational complexity in one iteration is $O(\rho_R l + \rho_C l)$, which indicates that the computational time of our method is linear with respect to the number of observations in the two sparse matrices. This complexity analysis shows that our proposed approach is very efficient and can scale up with respect to very large datasets.

5.1.2 Recommendation with Social Tags

In the above section, we demonstrate how to recommend by incorporating users' social trust information. Actually, this general framework can also be easily extended to fuse the user-item rating matrix with social tags information. We can use similar factor analysis approach by utilizing both users' rating information and tagging information at the same time in light of the facts that both users' rating information and users' tagging information can reflect their opinions about Web content. Specifically, on the one hand, we connect users' rating information with users' tagging information through the shared user latent feature space. The graphical model of this case is shown in Fig. 5.3, where the matrix T represents the latent feature of each tag, and F_{ik} indicates how many times that user u_i used tag t_k. We can also have the similar object function as shown in Eq. (5.10) with the parameter λ_T^U controlling how many users' tag information should be used. On the other hand, we connect items' received rating information with items' received tagging information through the shared item latent feature space. The related graphical model is shown in Fig. 5.4, where F_{jk} represents how many times that item v_j is tagged by tag t_k. In the objective function, we employ λ_T^V to control how many items' tag information should be incorporated.

The user latent feature space affects users' behaviors on both rating and tagging activities, while the item latent feature space determines both the received rating information and received tagging information.

5.2 Experimental Analysis

In this section, we conduct several experiments to compare the recommendation quality of our social recommendation approach

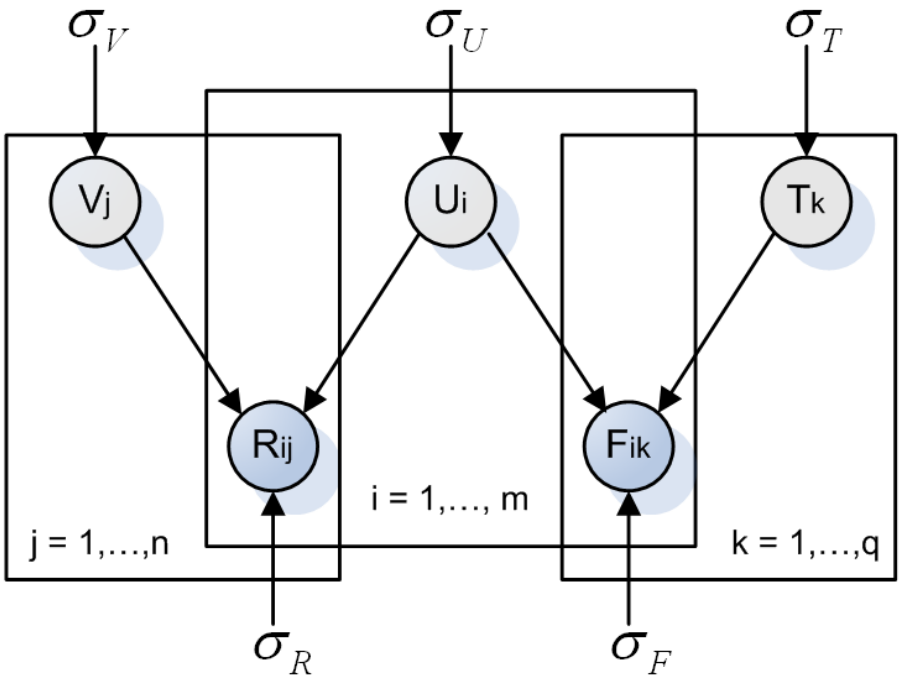


Figure 5.3: Graphical Model for Recommendation with User Tags

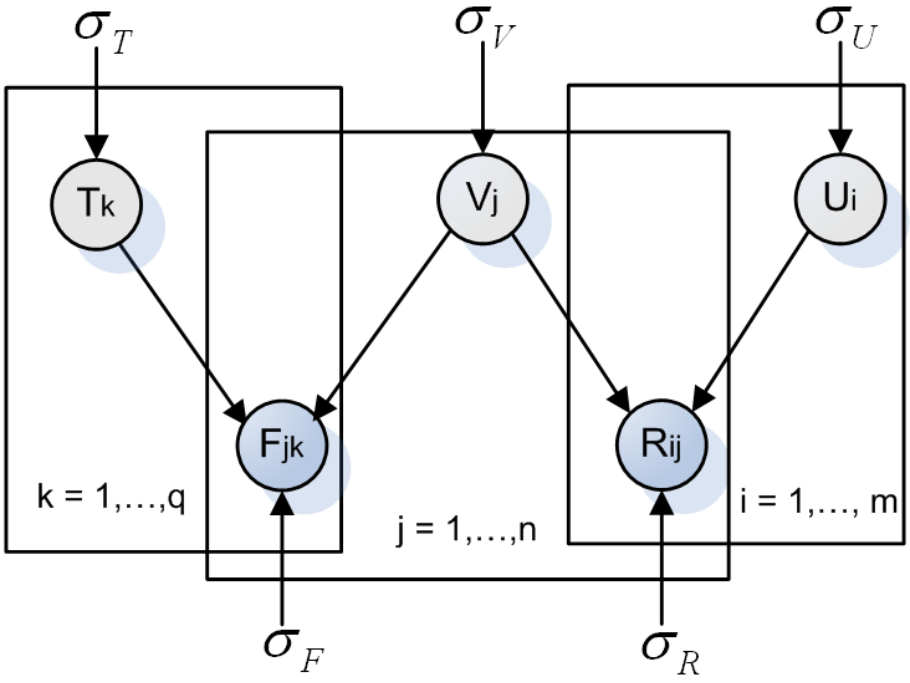


Figure 5.4: Graphical Model for Recommendation with Item Tags

with other state-of-the-art collaborative filtering methods. We conduct the experiments on two different datasets, one is Epinions which is associated with a social trust network, another is Movielens which has tag information that is issued by different users.

Our experiments are intended to address the following questions:

1. How does our approach compare with the published state-of-the-art collaborative filtering algorithms?
2. How does the model parameter λ_C affect the accuracy of prediction?
3. What is the performance comparison on users with different observed ratings?
4. Can our algorithm achieve good performance even if users have no observed ratings?
5. Is our algorithm efficient for large datasets?

5.2.1 Metrics

We use two metrics, the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE), to measure the prediction quality of our proposed approach in comparison with other collaborative filtering and trust-aware recommendation methods.

The metrics MAE is defined as:

$$MAE = \frac{\sum_{i,j} |r_{i,j} - \widehat{r}_{i,j}|}{N}, \tag{5.12}$$

where $r_{i,j}$ denotes the rating user i gave to item j, $\widehat{r}_{i,j}$ denotes the rating user i gave to item j as predicted by a method, and N denotes the number of tested ratings. The metrics RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum_{i,j}(r_{i,j} - \widehat{r}_{i,j})^2}{N}}. \tag{5.13}$$

5.2.2 Compared Methods

In this section, in order to show the performance improvement of our Recommendation Algorithm with Social Contextual Information (SoRec), we compare our algorithm with two baseline methods User Mean and Item Mean, as well as two state-of-the-art algorithms SVD [63] and Probabilistic Matrix Factorization (PMF) [99].

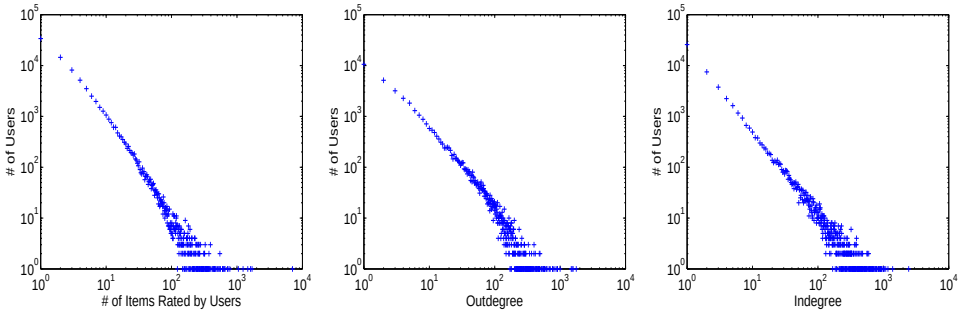
5.2.3 Epinions Dataset

Description of the Epinions Dataset

A tremendous amount of data has been produced on the Internet every day over the past decade. Millions of people influence each other implicitly or explicitly through online social network services, such as Facebook[1]. As a result, there are many online opportunities to mine social networks for the purposes of social recommendations.

We choose Epinions as the data source for our experiments on social recommendation. Epinions.com is a well known knowledge sharing and review site that was established in 1999. In order to add reviews, users (contributors) need to register for free and they begin submitting their own personal opinions on topics such as products, companies, movies, or reviews issued by other users. Users can also assign products or reviews integer ratings from 1 to 5. These ratings and reviews will influence future customers when they are deciding whether a product is worth buying or a movie is worth watching. Every member of

[1]http://www.facebook.com



(a) (b) (c)

Figure 5.5: Power-Law Distributions of the Epinions Dataset. (a) Items per User Distribution. (b) Trust Graph Outdegree Distribution. (c) Trust Graph Indegree Distribution.

Epinions maintains a "trust" list which presents a network of trust relationships between users, and a "block (distrust)" list which presents a network of distrust relationships. This network is called the "Web of trust", and is used by Epinions to re-order the product reviews such that a user first sees reviews by users that they trust. Epinions is thus an ideal source for experiments on social recommendation. Note that in this chapter, we only employ trust statements between users while ignoring the distrust statements, for the following two reasons: (1) The distrust list of each user is kept private in Epinions.com in order to protect the privacies of users, hence it is not available in our dataset. (2) As presented in [42], the understanding of distrust is more complicated than trust, which indicates that the user trust latent feature space may not be the same as the user distrust latent feature space. The study of distrust-based social recommendation will be conducted as future work.

The dataset used in our experiments is collected by crawling the Epinions.com site on Jan 2009. It consists of 51,670 users who have rated a total of 83,509 different items. The total number of ratings is 631,064. The density of the user-item rating matrix is less than 0.015%. We can observe that

Table 5.1: Statistics of User-Item Rating Matrix of Epinions

Statistics	User	Item
Max. Num. of Ratings	1,960	7,082
Avg. Num. of Ratings	12.21	7.56

Table 5.2: Statistics of Social Trust Network of Epinions

Statistics	Trust per User	Be Trusted per User
Max. Num.	1,763	2,443
Avg. Num.	9.91	9.91

the user-item rating matrix of Epinions is very sparse, since the densities for the two most famous collaborative filtering datasets Movielens (6,040 users, 3,900 movies and 1,000,209 ratings) and Eachmovie (74,424 users, 1,648 movies and 2,811,983 ratings) are 4.25% and 2.29%, respectively. Moreover, an important factor that we choose the Epinions dataset is that user social trust network information is not included in the Movielens and Eachmovie datasets. The statistics of the Epinions user-item rating matrix is summarized in Table 5.1. As to the user social trust network, the total number of issued trust statements is 511,799. The statistics of this data source is summarized in Table 5.2.

We also observe a number of power law distributions in our dataset, including items per user distribution, social trust network outdegree and indegree distributions. The distributions are shown in Fig. 5.5.

Comparison

We use different amounts of training data (90%, 80%, 70%, 60%) to test all the algorithms. Training data 90%, for example, means we randomly select 90% of the ratings from Epinions dataset as the training data to predict the remaining 10% of ratings. The random selection was carried out 5 times indepen-

Table 5.3: MAE comparison with other approaches on Epinions dataset (A smaller MAE value means a better performance)

Methods		90% Training	80% Training	70% Training	60% Training
User Mean		0.9294	0.9319	0.9353	0.9384
Item Mean		0.8936	0.9115	0.9316	0.9528
5D	SVD	0.8739	0.8946	0.9214	0.9421
	PMF	0.8678	0.8946	0.9127	0.9350
	SoRec	**0.8442**	**0.8638**	**0.8751**	**0.8948**
10D	SVD	0.8702	0.8921	0.9189	0.9382
	PMF	0.8651	0.8886	0.9092	0.9328
	SoRec	**0.8404**	**0.8580**	**0.8722**	**0.8921**

Table 5.4: RMSE comparison with other approaches on Epinions dataset (A smaller RMSE value means a better performance)

Methods		90% Training	80% Training	70% Training	60% Training
User Mean		1.1927	1.1968	1.2014	1.2082
Item Mean		1.1678	1.1973	1.2276	1.2505
5D	SVD	1.1635	1.1845	1.2067	1.2298
	PMF	1.1583	1.1773	1.1943	1.2163
	SoRec	**1.1333**	**1.1530**	**1.1690**	**1.1892**
10D	SVD	1.1600	1.1812	1.2011	1.2268
	PMF	1.1544	1.1760	1.1968	1.2230
	SoRec	**1.1293**	**1.1492**	**1.1660**	**1.1852**

dently. The experimental results are shown in Table 5.3. The parameter settings of our approach are $\lambda_C = 20$, $\lambda_U = \lambda_V = \lambda_Z = 0.001$, and in all the experiments conducted in the following sections, we set all of the parameters λ_U, λ_V and λ_Z equal to 0.001. From Table 5.3 and Table 5.4, we can observe that our approach outperforms the other methods. The improvements are significant, which shows the promising future of our recommendation approach.

Impact of Parameter λ_C

The main advantage of our recommendation approach is that it incorporates the social trust network information, which helps predict users' preferences. In our model, parameter λ_C balances the information from the user-item rating matrix and the user social trust network. If $\lambda_C = 0$, we only mine the user-item rating matrix for matrix factorization, and if $\lambda_C = \infty$, we only extract information from the social network to predict users' preferences. In other cases, we fuse information from the user-item rating matrix and the user social network for probabilistic matrix factorization and, furthermore, to predict ratings for active users.

Fig. 5.6 shows the impacts of λ_C on MAE and RMSE. We observe that the value of λ_C impacts the recommendation results significantly, which demonstrates that fusing the user-item rating matrix with the user social trust network greatly improves the recommendation accuracy. As λ_C increases, the prediction accuracy also increases at first, but when λ_C surpasses a certain threshold, the prediction accuracy decrease with further increase of the value of λ_C. This phenomenon confirms the intuition that fusing the user-item rating matrix and the user social trust network can generate better performance than only purely using each of these two resources separately. From Fig. 5.6, we observe that for this Epinions dataset, our social recommendation

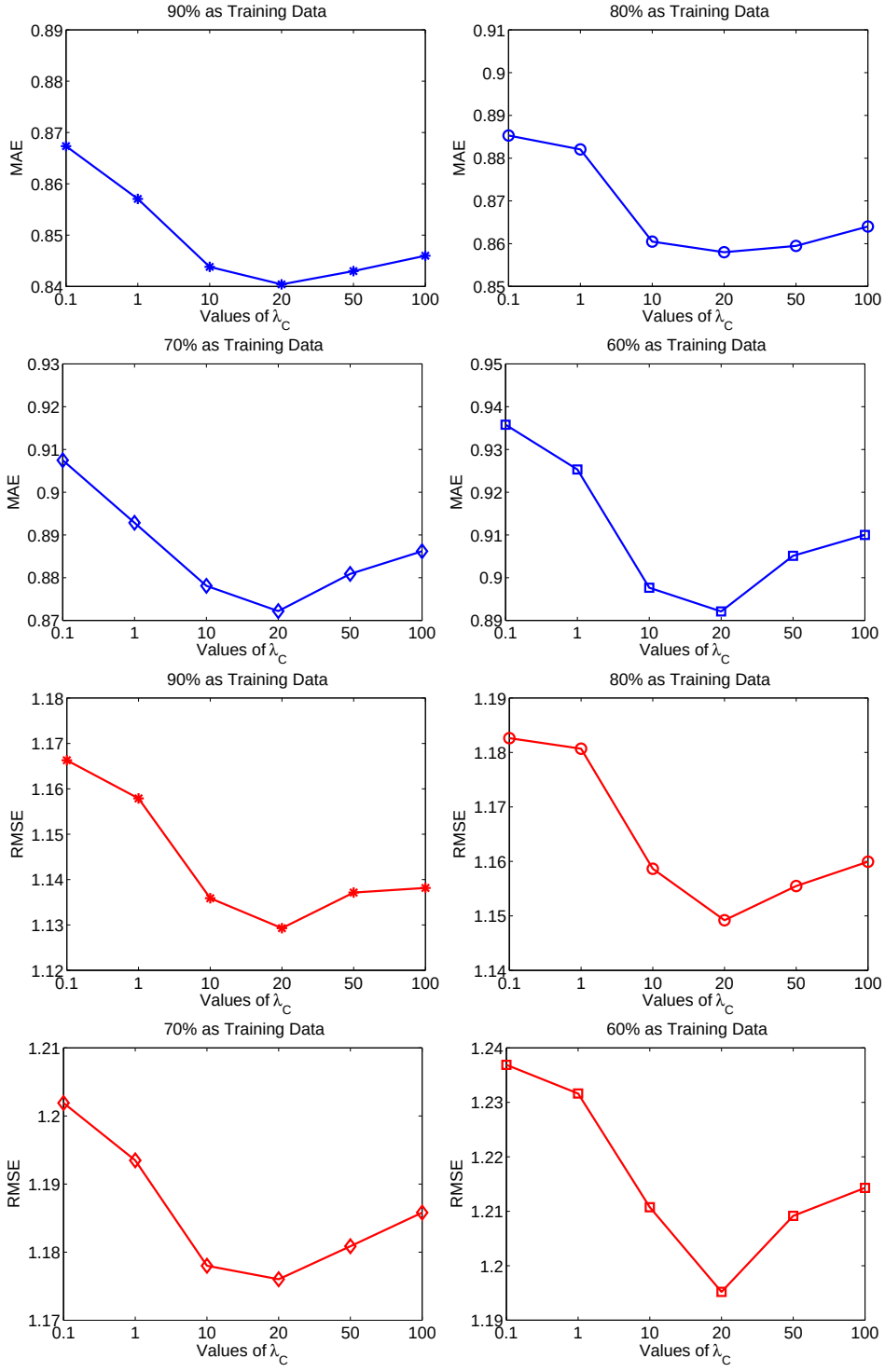
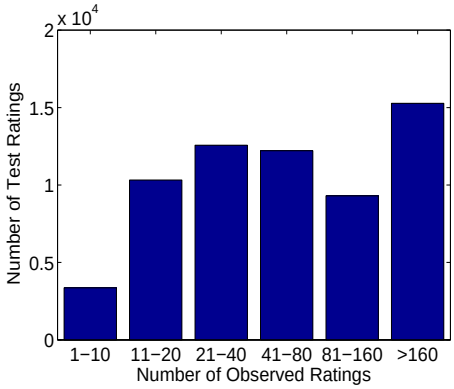
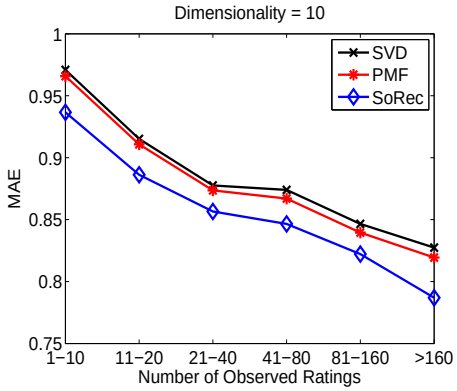


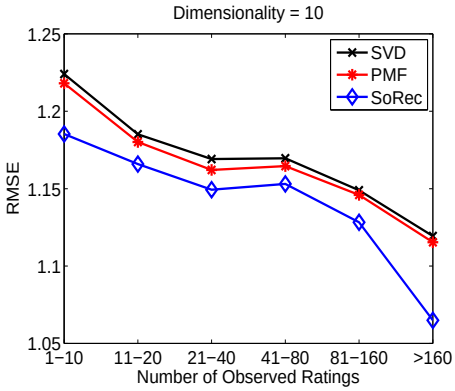
Figure 5.6: Impact of Parameter λ_C (Dimensionality = 10)



(a) Distribution of Testing Data (90% as Training Data)



(b) MAE Comparison on Different User Rating Scales (90% as Training Data)



(c) RMSE Comparison on Different User Rating Scales (90% as Training Data)

Figure 5.7: Performance Comparison on Different Users

method achieves the best performance when λ_C is around 20, while smaller values like $\lambda_C = 0.1$ or larger values $\lambda_C = 100$ can potentially degrade the model performance.

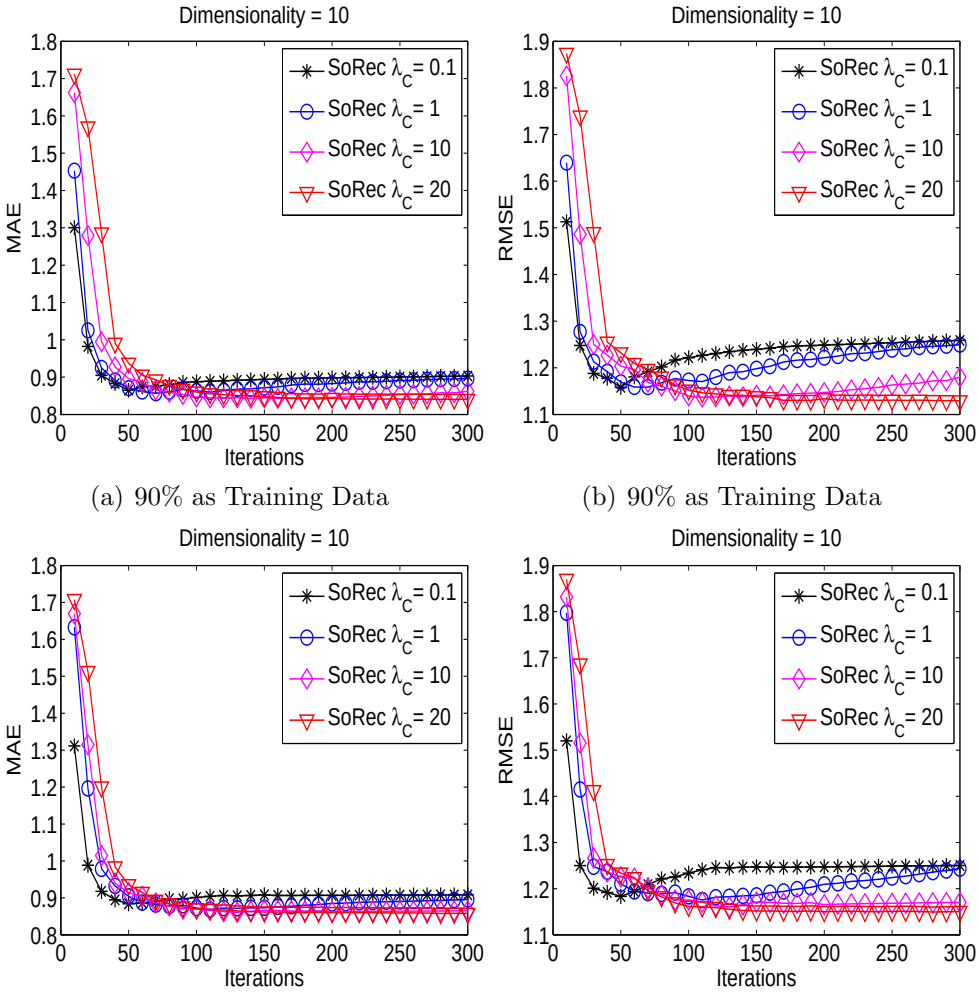
Performance on Different Users

One main task we target in this chapter is to provide accurate recommendations when users only supply a few ratings or even have no rating records. Although previous work has noticed this critical problem, few approaches perform well when few user ratings are given. Hence, in order to compare our approach with the other methods thoroughly, we first group all the users based on the number of observed ratings in the training data, and then evaluate prediction accuracies of different user groups. The experimental results are shown in Fig. 5.7. Users are grouped into 10 classes: "$= 0$", "$1 - 5$", "$6 - 10$", "$11 - 20$", "$21 - 40$", "$41 - 80$", "$81 - 160$", "$160 - 320$", "$320 - 640$", and "> 640", denoting how many ratings users have rated.

Fig. 5.7(a) summarizes the distributions of testing data according to groups in the training data (90% as training data). For example, there are a total of 3,360 user-item pairs to be predicted in the testing dataset in which the related users in the training dataset have rating numbers from 1 to 10. In Fig. 5.7(b) and Fig. 5.7(c), we observe that our SoRec algorithm consistently outperforms other methods even when users only rated very few ratings.

Efficiency Analysis

The complexity analysis in Section 5.1.1 states that the computational complexity of our approach is linear with respect to the number of ratings, which proves that our approach is scalable to very large datasets. Actually, our approach is very efficient even when using a very simple gradient descent method. In the



(a) 90% as Training Data (b) 90% as Training Data

(c) 80% as Training Data (d) 80% as Training Data

Figure 5.8: Efficiency Analysis

experiments using 90% of the data as training data, each iteration only needs less than 2 seconds. Also, as shown in Fig. 5.8, when using 90% of the data as training data, our method needs less than 300 iterations to converge, which only needs approximately 10 minutes. When using 60% of the data as training data, we only need less than 5 minutes to train the model. All the experiments are conducted on a normal personal computer containing an *Intel Pentium D* CPU (3.0 GHz, *Dual Core*) and 1 *Giga* byte memory.

From Fig. 5.8, we also observe that when using a small value of λ_C, such as $\lambda_C = 0.1$ or $\lambda_C = 1$, after 50 or 100 iterations, the model begins to overfit, while a larger λ_C, such as $\lambda_C = 20$, does not have the overfitting problem. These experiments clearly demonstrate that in this Epinion dataset, using little social network information can cause overfiting problem, and that the predictive accuracy can be improved by incorporating more social network information.

5.2.4 MovieLens Dataset

Description of the MovieLens Dataset

MovieLens is a famous recommender system. The dataset we employ in this chapter is the 10M/100K dataset. This data set contains 10,000,054 ratings and 95,580 tags added to 10,681 movies by 71,567 users of the online movie recommender service MovieLens.

Comparison

In the comparison, we employ different amounts of training data, including 80%, 50%, 30%, 10%. 80% training data means we randomly select 80% of the ratings from the MovieLens 10M/100K data set as the training data, and leave the remaining 20% as prediction performance testing. The procedure is carried out 5 times independently, and we report the average values in this chapter.

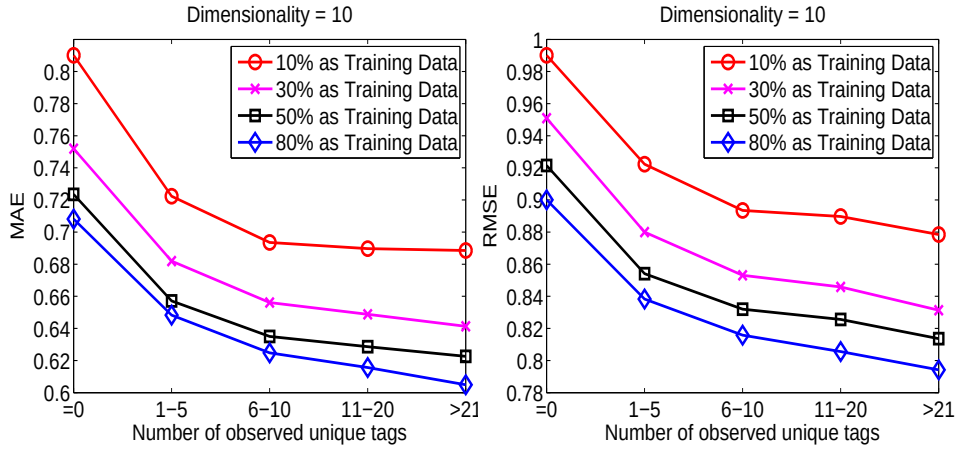
As introduced in Section 5.1.2, we can incorporate social tag information in two ways: (1) the first method is to treat the tags as the favors of users (we call this method SoRecUser, and it is related to the graphical model shown in Fig. 5.3 with the parameter λ_T^U); (2) the second method is to interpret the tags as the properties of items (we call this method SoRecItem, and

Table 5.5: MAE comparison with other approaches on MovieLens dataset (A smaller MAE value means a better performance)

Methods		80% Training	50% Training	30% Training	10% Training
User Mean		0.7686	0.7710	0.7742	0.8234
Item Mean		0.7379	0.7389	0.7399	0.7484
5D	SVD	0.6390	0.6547	0.6707	0.7448
	PMF	0.6325	0.6542	0.6698	0.7430
	SoRecUser	0.6209	0.6419	0.6607	0.7040
	SoRecItem	**0.6199**	**0.6407**	**0.6395**	**0.7026**
10D	SVD	0.6386	0.6534	0.6693	0.7431
	PMF	0.6312	0.6530	0.6683	0.7417
	SoRecUser	0.6197	0.6408	0.6595	0.7028
	SoRecItem	**0.6187**	**0.6395**	**0.6584**	**0.7016**

Table 5.6: RMSE comparison with other approaches on MovieLens dataset (A smaller RMSE value means a better performance)

Methods		80% Training	50% Training	30% Training	10% Training
User Mean		0.9779	0.9816	0.9869	1.1587
Item Mean		0.9440	0.9463	0.9505	0.9851
5D	SVD	0.8327	0.8524	0.8743	0.9892
	PMF	0.8310	0.8582	0.8758	0.9698
	SoRecUser	0.8121	0.8384	0.8604	0.9042
	SoRecItem	**0.8112**	**0.8370**	**0.8591**	**0.9033**
10D	SVD	0.8312	0.8509	0.8728	0.9878
	PMF	0.8295	0.8569	0.8743	0.9681
	SoRecUser	0.8110	0.8372	0.8593	0.9034
	SoRecItem	**0.8097**	**0.8359**	**0.8578**	**0.9019**



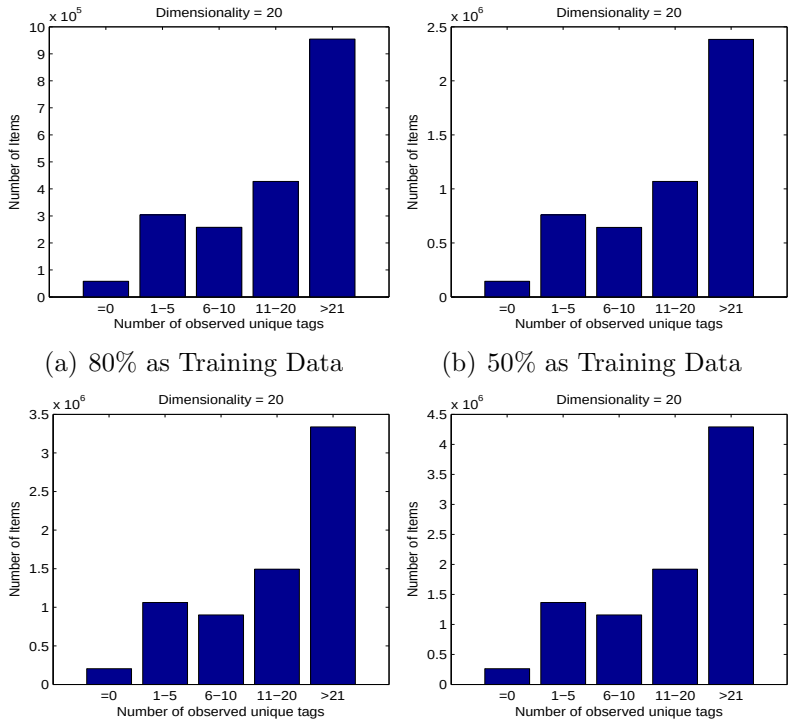
(a) MAE (b) RMSE

Figure 5.9: Performance Comparison on Items with Different # of Tags

it is associated with the graphical model shown in Fig. 5.4 with the parameter λ_T^V).

In the comparison, we set $\lambda_T^U = 1$ and $\lambda_T^V = 10$. The MAE results and RMSE results are reported in Table 5.5 and Table 5.6, respectively. From the results, we can see that our SoRecUser and SoRecItem approaches consistently outperform the baseline methods and the state-of-the-art recommendation algorithms, especially when there is a small amount of training data, which is equivalent to data sparsity in reality. In addition, it is necessary to notice that in the MovieLens 10M/100K data set, all the selected users have rated at least 20 movies, but in reality, according to the famous power law distribution phenomenon, in almost all kinds of Web activities, most users only rated very few items. Thus, we can see the improvement of our method is significant, and this again shows the promising future of our approach.

As to the parameters λ_T^U and λ_T^V basically, they share the similar trends with Fig. 5.6, hence we do not show the detailed results here.



(a) 80% as Training Data (b) 50% as Training Data

(c) 30% as Training Data (d) 10% as Training Data

Figure 5.10: Tag Distributions of Testing Data on Different Amount of Training Data

Performance on Items with Different Number of Tags

One major contribution of this chapter is incorporating social tagging information with traditional rating information to improve prediction quality. In order to further investigate how the number of tags attached to one item affects the prediction accuracies, we first group all the items based on the number of unique tags they have been annotated, then evaluate the prediction accuracies on different groups. We divide the items into 5 groups based on the number of unique tags that have been annotated: "= 0", "1-5", "6-10", "11-20", and "≥21".

Experimental results are presented in Fig. 5.9. This figure shows the prediction accuracies (measured with MAE and RMSE) of groups of items annotated with different number of unique tags, and the results of different amount of training data are all presented. We only report the results on dimensionality = 10. From Fig. 5.9, we can see that incorporating tags information can improve prediction quality significantly. In addition, as the number of annotated unique tags increases, the prediction quality first improves drastically, then gradually stabilizes after the number of tags surpasses some threshold value (around 20 in this data set). This phenomenon is reasonable, because with more tags' information, the concept of an item can be represented more accurately, but too many tags result in redundancy in representing the concepts of the items. Fig. 5.10 shows the tag distributions of testing data on different amount of training data.

5.3 Summary

In this chapter, in order to alleviate the data sparsity problem in the traditional recommender systems, we present a novel, efficient and general recommendation framework fusing a user-item

rating matrix with social contextual information using probabilistic matrix factorization. The experimental results show that our approach outperforms the other state-of-the-art collaborative filtering algorithms, and the complexity analysis indicates it is scalable to very large datasets. Moreover, the data fusion method using probabilistic matrix factorization we introduce in this chapter is not only applicable to recommendation with social contextual information, but also extensible to other popular research topics, such as social search.

For future work, we employ the inner product of two vectors to fit the observed data in this chapter; this approach assumes that the observed data is a linear combination of several latent factors. Although we use the logistic function to constrain the inner product, a more natural and accurate improvement over this assumption is to use a kernel representation for the two low-dimensional vectors, such as a Gaussian Kernel or a Polynomial Kernel, which map the relations of the two vectors into a nonlinear space, and thus leading to an increase in the model's performance.

Moreover, we only employ inter-user trust information in this chapter, but in many online social networks, the distrust information is also stated by many users. Because a user trust feature space may not be consistent with the corresponding user distrust feature space, we cannot simply incorporate the distrust information into our model. In the future, we need to investigate the following two problems: whether the distrust information is useful to increase the prediction quality, and how to incorporate this distrust information to obtain better quality results.

Furthermore, when fusing the social trust network information, we ignore the information diffusion or propagation between users. A more accurate approach is to consider the diffusion process between users. Hence, we need to replace the social network matrix factorization with the social network diffusion processes.

This consideration will help alleviate the data sparsity problem and will potentially increase the prediction accuracy.

Lastly, we either associate tags with users or associate tags with items. Actually, we can design a more general framework to incorporate tags with users and items simultaneously. This consideration will provide more information than either of the proposed methods, hence can further improve the recommendation quality.

□ **End of chapter.**

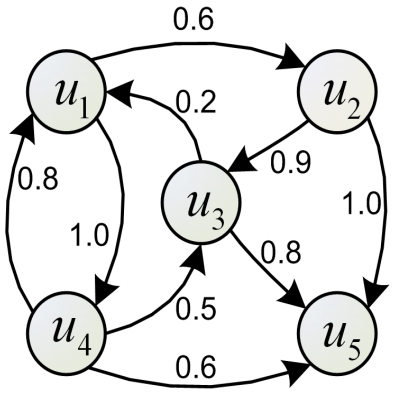
Chapter 6

Recommend with Social Trust Ensemble

In last chapter, we developed a factor analysis method based on the probabilistic graphical model which fuses the user-item matrix with the users' social trust networks by sharing a common latent low-dimensional user feature matrix. The experimental analysis shows that this method generates better recommendations than the non-social collaborative filtering algorithms. However, the disadvantage of this work is that although the users' social trust network is integrated into the recommender systems by factorizing the social trust graph, the real world recommendation processes are not reflected in the model. This drawback not only causes lack of interpretability in the model, but also affects the recommendation qualities. A more novel and realistic approach is needed to model the trust-aware recommendation problem.

6.1 Recommendation with Social Trust Ensemble

Traditional recommender system techniques, like collaborative filtering, only utilize the information of the user-item rating matrix for recommendations while ignore the social trust relations



(a) Social Trust Graph

	v_1	v_2	v_3	v_4	v_5	v_6
u_1		5	2		3	
u_2	4			3		4
u_3			2			2
u_4	5			3		
u_5		5	5			3

(b) User-Item Rating Matrix

Figure 6.1: Example for Trust based Recommendation

among users. As the exponential growth of online social networks, incorporating social trust information into recommender systems is becoming more and more important. In this section, we first describe the trust-aware recommendation problem in Section 6.1.1, and then provide the solution in Sections 6.1.2, 6.1.3 and 6.1.4.

6.1.1 Problem Description

In the real world, the process of recommendation scenario includes two central elements: the trust network and the favors of these friends, which can essentially be modeled by the examples of the trust graph in Fig. 6.1(a) and the user-item rating matrix in Fig. 6.1(b), respectively. In the trust graph illustrated in Fig. 6.1(a), totally, 5 users (nodes, from u_1 to u_5) are connected with 9 relations (edges) between users, and each relation is associated with a weight S_{ij} in the range $(0, 1]$ to specify how much user u_i knows or trusts user u_j. Normally, the trust relations in the online trust network are explicitly stated by online users. As illustrated in Fig. 6.1(b), each user also rated some items (from v_1 to v_6) on a 5-point integer scale to express the extent of the favor of each item (normally, 1, 2, 3, 4 and 5 represent "hate",

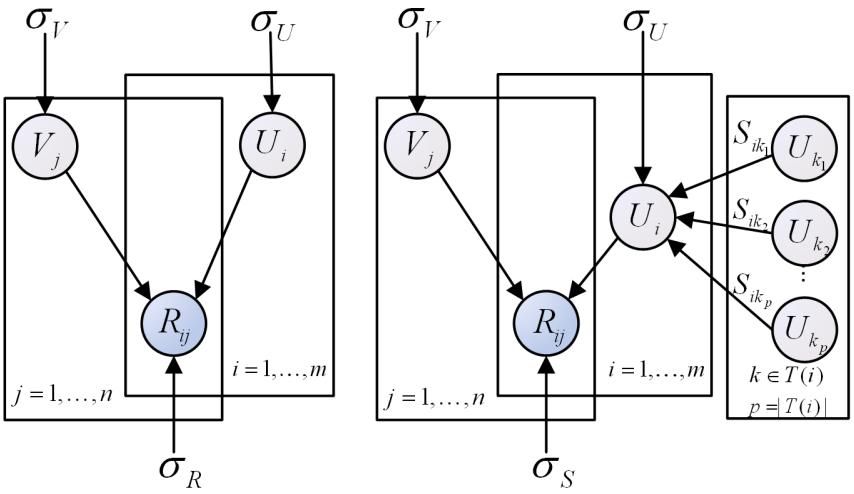
"don't like", "neutral", "like" and "love", respectively). The problem we study in this chapter is how to predict the missing values for the users effectively and efficiently by employing the trust graph and the user-item rating matrix.

6.1.2 User Features Learning

In order to learn the characteristics or features of the users, we employ matrix factorization to factorize the user-item matrix. The idea of user-item matrix factorization is to derive a high-quality l-dimensional feature representation U of users and V of items based on analyzing the user-item matrix R. Suppose in a user-item rating matrix, we have m users, n items, and rating values within the range $[0, 1]$. Actually, most recommender systems use integer rating values from 1 to R_{max} to represent the users' judgements on items. In this chapter, without loss of generality, we map the ratings $1, ..., R_{max}$ to the interval $[0, 1]$ using the function $f(x) = x/R_{max}$. Let R_{ij} represent the rating of user u_i for item v_j, and $U \in R^{l \times m}$ and $V \in R^{l \times n}$ be latent user and item feature matrices, with column vectors U_i and V_j representing the l-dimensional user-specific and item-specific latent feature vectors of user u_i and item v_j, respectively. Note that the solutions of U and V are not unique. In [99], the conditional distribution over the observed ratings is defined as:

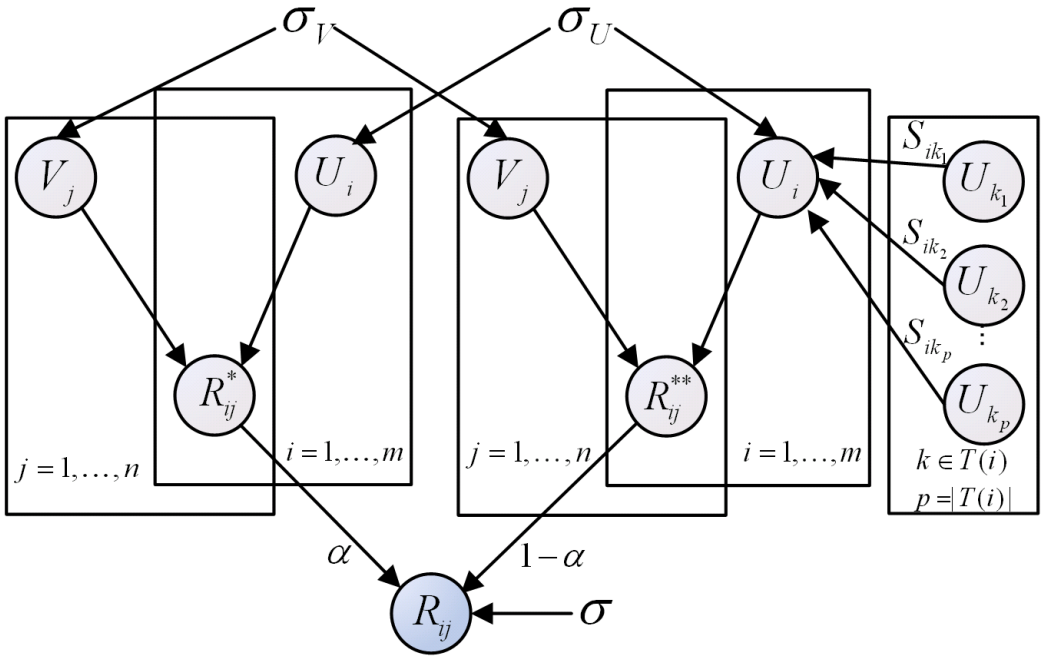
$$p(R|U, V, \sigma_R^2) = \prod_{i=1}^{m} \prod_{j=1}^{n} \left[\mathcal{N}\left(R_{ij}|g(U_i^T V_j), \sigma_R^2\right) \right]^{I_{ij}^R}, \qquad (6.1)$$

where $\mathcal{N}(x|\mu, \sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and variance σ^2, and I_{ij}^R is the indicator function that is equal to 1 if user u_i rated item v_j and equal to 0 otherwise. The function $g(x)$ is the logistic function $g(x) = 1/(1 + \exp(-x))$, which makes it possible to bound the range of $U_i^T V_j$ within the range $[0, 1]$. The zero-mean spherical



(a) Factorization of User-Item Matrix

(b) Recommendations by Trusted Friends



(c) Recommendations with Social Trust Ensemble

Figure 6.2: Graphical Models

Gaussian priors are also placed on user and item feature vectors:

$$p(U|\sigma_U^2) = \prod_{i=1}^{m} \mathcal{N}(U_i|0, \sigma_U^2 \mathbf{I}),\ p(V|\sigma_V^2) = \prod_{j=1}^{n} \mathcal{N}(V_j|0, \sigma_V^2 \mathbf{I}). \quad (6.2)$$

Hence, through a Bayesian inference, we have

$$\begin{aligned} p(U, V|R, \sigma_R^2, \sigma_U^2, \sigma_V^2) &\propto p(R|U, V, \sigma_R^2) p(U|\sigma_U^2) p(V|\sigma_V^2) \\ &= \prod_{i=1}^{m} \prod_{j=1}^{n} \left[\mathcal{N}\left(R_{ij}|g(U_i^T V_j), \sigma_R^2\right)\right]^{I_{ij}^R} \\ &\times \prod_{i=1}^{m} \mathcal{N}(U_i|0, \sigma_U^2 \mathbf{I}) \times \prod_{j=1}^{n} \mathcal{N}(V_j|0, \sigma_V^2 \mathbf{I}). \end{aligned} \quad (6.3)$$

The graphical model of Eq. (6.3) is shown in Fig. 6.2(a). This equation represents the method on how to derive the users' latent feature space or users' characteristics purely based on the user-item rating matrix without considering the favors of users' trusted friends. In the next section, we will systematically illustrate how to recommend based on the tastes of trusted friends.

6.1.3 Recommendations by Trusted Friends

In this section, we analyze how our social trust networks affect our decisions or behaviors, and propose a method to recommend only by using the tastes of trusted friends.

Suppose we have a directed social trust graph $\mathcal{G} = (\mathcal{U}, \mathcal{E})$, where the vertex set $\mathcal{U} = \{u_i\}_{i=1}^m$ represents all the users in a social trust network and the edge set $\mathcal{E}$ represents the trust relations between users. Let $S = \{S_{ij}\}$ denote the $m \times m$ matrix of $\mathcal{G}$, which is also called the social trust matrix in this chapter. For a pair of vertices, u_i and u_j, let $S_{ij} \in (0, 1]$ denote the weight associated with an edge from u_i to u_j, and $S_{ij} = 0$, otherwise. The physical meaning of the weight S_{ij} can be interpreted as how much a user u_i trusts or knows user u_j in a social network.

Note that social trust matrix S is an asymmetric matrix, since in a trust-based social network, user u_i trusting u_j does not necessary indicate user u_j trusts u_i.

In reality, we always turn to our friends for recommendations since we trust our friends. We also believe that most probably we will like the items (books, music, movies, etc.) that our trusted friends recommend. Even if the recommended items are not the types we like, we still have a high probability to be influenced by our trusted friends. In the real world, suppose a user wants to see the movie "The Dark Knight" (suppose it is the item v_1 in Fig. 6.1(b)), which is now playing at the theaters, but he/she knows nothing about the movie, like user u_1 in Fig. 6.1(b). What this user normally do is to take into account his/her trusted friends' recommendations. Among all of his/her trusted friends in Fig. 6.1(a), u_2 and u_4 rated this movie as 4 and 5, and u_1 trusts u_4 (weight 1.0) more than u_2 (weight 0.6). Based on the information, there is a very high probability that u_1 will draw the conclusion that "The Dark Knight" is a very good movie worth of watching.

From the above analysis, we can generalize the above social process as

$$\widehat{R}_{ik} = \frac{\sum\limits_{j \in \mathcal{T}(i)} R_{jk} S_{ij}}{|\mathcal{T}(i)|}, \tag{6.4}$$

where $\widehat{R}_{ik}$ is the prediction of the rating that user u_i would give item v_j, R_{jk} is the score that user u_j gave item v_k, $\mathcal{T}(i)$ is the friends set that user u_i trusts and $|\mathcal{T}(i)|$ is the number of trusted friends of user u_i in the set $\mathcal{T}(i)$. $|\mathcal{T}(i)|$ can be merged into S_{ij} since it is the normalization term of trust scores. Hence, Eq. (6.4) can be simplified as

$$\widehat{R}_{ik} = \sum_{j \in \mathcal{T}(i)} R_{jk} S_{ij}. \tag{6.5}$$

Then the prediction of the ratings that user u_i gives to all the items can be inferred as

$$\begin{pmatrix} \widehat{R}_{i1} \\ \widehat{R}_{i2} \\ ... \\ \widehat{R}_{in} \end{pmatrix} = \begin{pmatrix} R_{11} & R_{21} & ... & R_{m1} \\ R_{12} & R_{22} & ... & R_{m2} \\ ... & ... & ... & ... \\ R_{1n} & R_{2n} & ... & R_{mn} \end{pmatrix} \begin{pmatrix} S_{i1} \\ S_{i2} \\ ... \\ S_{im} \end{pmatrix}. \tag{6.6}$$

We can then infer that for all the users to obtain

$$\widehat{R} = SR, \tag{6.7}$$

where SR can be interpreted as the recommendations purely based on the trusted friends' tastes.

From the social trust network aspect, we define the conditional distribution over the observed ratings as

$$p(R|S,U,V,\sigma_R^2) = \prod_{i=1}^{m}\prod_{j=1}^{n}\left[\mathcal{N}\left(R_{ij}|g(\sum_{k\in\mathcal{T}(i)} S_{ik}U_k^T V_j), \sigma_S^2\right)\right]^{I_{ij}^R}, \tag{6.8}$$

where S_{ik} is normalized by $|\mathcal{T}(i)|$, which is the number of trusted friends of user u_i in the set $\mathcal{T}(i)$. I_{ij}^R is the indicator function that is equal to 1 if user i rated item j and equal to 0 otherwise.

Hence, similar to Eq. (6.3), through a Bayesian inference, we have

$$\begin{aligned} &p(U,V|R,S,\sigma_S^2,\sigma_U^2,\sigma_V^2) \\ &\quad\propto\ p(R|S,U,V,\sigma_S^2)p(U|S,\sigma_U^2)p(V|S,\sigma_V^2). \end{aligned} \tag{6.9}$$

In Eq. (6.9), we can assume that S is independent with the low-dimensional matrices U and V, then this equation can be changed to

$$
\begin{aligned}
p(U,V|R,S,\sigma_S^2,\sigma_U^2,\sigma_V^2) &\propto p(R|S,U,V,\sigma_S^2)p(U|\sigma_U^2)p(V|\sigma_V^2), \\
&= \prod_{i=1}^{m}\prod_{j=1}^{n}\left[\mathcal{N}\left(R_{ij}|g(\sum_{k\in\mathcal{T}(i)} S_{ik}U_k^T V_j),\sigma_S^2\right)\right]^{I_{ij}^R} \\
&\times \prod_{i=1}^{m}\mathcal{N}(U_i|0,\sigma_U^2\mathbf{I}) \times \prod_{j=1}^{n}\mathcal{N}(V_j|0,\sigma_V^2\mathbf{I}). \qquad (6.10)
\end{aligned}
$$

where $p(U|\sigma_U^2)$ and $p(V|\sigma_V^2)$ are zero-mean spherical Gaussian priors on user and item feature vectors. This equation specifies the method to recommend purely based on users' trusted friends' tastes. The graphical model is shown in Fig. 6.2(b).

6.1.4 Social Trust Ensemble

In Section 6.1.2, given the user-item rating matrix, the observed rating R_{ij} is interpreted by the user u_i's favor on item v_j, while in Section 6.1.3, given the user-item rating matrix and users' social trust network, the observed rating R_{ij} is realized as the favors on item v_j of user u_i's trusted friends. Actually, both of the above assumptions are partially right since in the real world situation, every user has his/her own taste and at the same time, every user may be influenced by his/her friends he/she trusts. Hence, in order to define the model more realistically, every observed rating in the user-item matrix should reflect both of these two factors. Based on this motivation, we model the conditional

distribution over the observed ratings as:

$$
\begin{aligned}
&p(U, V|R, S, \sigma^2, \sigma_U^2, \sigma_V^2) \\
&= \prod_{i=1}^{m}\prod_{j=1}^{n}\left[\mathcal{N}\left(R_{ij}|g(\alpha U_i^T V_j + (1-\alpha)\sum_{k\in\mathcal{T}(i)} S_{ik}U_k^T V_j), \sigma^2\right)\right]^{I_{ij}^R} \\
&\times \prod_{i=1}^{m}\mathcal{N}(U_i|0, \sigma_U^2\mathbf{I}) \times \prod_{j=1}^{n}\mathcal{N}(V_j|0, \sigma_V^2\mathbf{I}). \qquad (6.11)
\end{aligned}
$$

In Eq. (6.11), the users' favors and the trusted friends' favors are smoothed by the parameter α, which naturally fuses appropriate amount of real world recommendation processes into the recommender systems. The parameter α controls how much do users trust themselves or their trusted friends. It is also the reason we call our approach Recommendation with Social Trust Ensemble (RSTE). The graphical model of RSTE is shown in Fig. 6.2(c).

The log of the posterior distribution for the recommendations is given by

$$
\begin{aligned}
&\ln p(U, V|R, S, \sigma^2, \sigma_U^2, \sigma_V^2) = \\
&-\frac{1}{2\sigma^2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^R(R_{ij} - g(\alpha U_i^T V_j + (1-\alpha)\sum_{k\in\mathcal{T}(i)} S_{ik}U_k^T V_j))^2 \\
&-\frac{1}{2\sigma_U^2}\sum_{i=1}^{m} U_i^T U_i - \frac{1}{2\sigma_V^2}\sum_{j=1}^{n} V_j^T V_j \\
&-\frac{1}{2}(\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^R)\ln\sigma^2 - \frac{1}{2}(ml\ln\sigma_U^2 + nl\ln\sigma_V^2) + \mathcal{C}, \qquad (6.12)
\end{aligned}
$$

where $\mathcal{C}$ is a constant that does not depend on the parameters. Maximizing the log-posterior over two latent features with hyperparameters (i.e., the observation noise variance and prior variances) kept fixed is equivalent to minimizing the following

sum-of-squared-errors objective functions with quadratic regularization terms:

$$\begin{aligned}
&\mathcal{L}(R,S,U,V)\\
&=\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{R}(R_{ij}-g(\alpha U_i^T V_j+(1-\alpha)\sum_{k\in\mathcal{T}(i)} S_{ik}U_k^T V_j))^2\\
&+\frac{\lambda_U}{2}\|U\|_F^2+\frac{\lambda_V}{2}\|V\|_F^2, \qquad (6.13)
\end{aligned}$$

where $\lambda_U = \sigma^2/\sigma_U^2$, $\lambda_V = \sigma^2/\sigma_V^2$, and $\|\cdot\|_F^2$ denotes the Frobenius norm.

A local minimum of the objective function given by Eq. (6.13) can be found by performing gradient descent in U_i, V_j,

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial U_i} &= \alpha\sum_{j=1}^{n} I_{ij}^{R} g'(\alpha U_i^T V_j+(1-\alpha)\sum_{k\in\mathcal{T}(i)} S_{ik}U_k^T V_j)V_j\\
&\times(g(\alpha U_i^T V_j+(1-\alpha)\sum_{k\in\mathcal{T}(i)} S_{ik}U_k^T V_j)-R_{ij})\\
&+(1-\alpha)\sum_{p\in\mathcal{B}(i)}\sum_{j=1}^{n} I_{pj}^{R} g'(\alpha U_p^T V_j+(1-\alpha)\sum_{k\in\mathcal{T}(p)} S_{pk}U_k^T V_j)\\
&\times(g(\alpha U_p^T V_j+(1-\alpha)\sum_{k\in\mathcal{T}(p)} S_{pk}U_k^T V_j)-R_{pj})S_{pi}V_j+\lambda_U U_i,\\
\frac{\partial\mathcal{L}}{\partial V_j} &= \sum_{i=1}^{m} I_{ij}^{R} g'(\alpha U_i^T V_j+(1-\alpha)\sum_{k\in\mathcal{T}(i)} S_{ik}U_k^T V_j)\\
&\times(g(\alpha U_i^T V_j+(1-\alpha)\sum_{k\in\mathcal{T}(i)} S_{ik}U_k^T V_j)-R_{ij})\\
&\times(\alpha U_i+(1-\alpha)\sum_{k\in\mathcal{T}(i)} S_{ik}U_k^T)+\lambda_V V_j, \qquad (6.14)
\end{aligned}$$

where $g'(x)$ is the derivative of logistic function $g'(x) = \exp(x)/(1+\exp(x))^2$ and $\mathcal{B}(i)$ is the set that includes all the users who trust

user u_i. In order to reduce the model complexity, in all of the experiments we conduct in Section 6.2, we set $\lambda_U = \lambda_V$.

6.1.5 Complexity Analysis

The main computation of gradient methods is evaluating the object function $\mathcal{L}$ and its gradients against variables. Because of the sparsity of matrices R and S, the computational complexity of evaluating the object function $\mathcal{L}$ is $O(\rho_R l + \rho_R \overline{k} l)$, where ρ_R is the number of nonzero entries in the matrix R, and $\overline{k}$ is the average number of friends that a user trusts. Since almost all of the online social networks fit the power-law distribution, a large long tail of users only have few trusted friends. This indicates that the value of $\overline{k}$ is relatively small. The computational complexities for the gradients $\frac{\partial \mathcal{L}}{\partial U}$ and $\frac{\partial \mathcal{L}}{\partial V}$ in Eq. (6.14) are $O(\rho_R \overline{p}\ l + \rho_R \overline{p}\ \overline{k} l)$ and $O(\rho_R l + \rho_R \overline{k} l)$, respectively, where $\overline{p}$ is the average number of friends who trust a user, which is also a small value. Actually, in a social trust graph, the value of $\overline{k}$ is always equal to the value of $\overline{p}$, which is 9.91 in the dataset we employ in the Section 6.2. Therefore, the total computational complexity in one iteration is $O(\rho_R \overline{p}\ l + \rho_R \overline{p}\ \overline{k} l)$, which indicates that theoretically, the computational time of our method is linear with respect to the number of observations in the user-item matrix R. This complexity analysis shows that our proposed approach is very efficient and can scale to very large datasets.

6.2 Empirical Analysis

In this section, we conduct several experiments to compare the recommendation qualities of our RSTE approach with other state-of-the-art collaborative filtering and trust-aware recommendation methods. Our experiments are intended to address the following questions: (1) How does our approach compare

Table 6.1: Statistics of User-Item Rating Matrix of Epinions

Statistics	User	Item
Max. Num. of Ratings	1960	7082
Avg. Num. of Ratings	12.21	7.56

Table 6.2: Statistics of Social Trust Network of Epinions

Statistics	Trust per User	Be Trusted per User
Max. Num.	1763	2443
Avg. Num.	9.91	9.91

with the published state-of-the-art collaborative filtering and trust-aware recommendation algorithms? (2) How does the model parameter α affect the accuracy of prediction? (3) What is the performance comparison on users with different observed ratings? (4) Can our algorithm achieve good performance even if users have few observed rating records? (5) Is our algorithm efficient when training the model?

6.2.1 Dataset Description

We choose Epinions as the data source for our experiments on recommendation with social trust ensemble. Epinions.com is a well known knowledge sharing site and review site, which was established in 1999. In order to add reviews, users (contributors) need to register for free and begin submitting their own personal opinions on topics such as products, companies, movies, or reviews issued by other users. Users can also assign products or reviews integer ratings from 1 to 5. These ratings and reviews will influence future customers when they are about to decide whether a product is worth buying or a movie is worth watching. Every member of Epinions maintains a "trust" list which presents a social network of trust relationships between users. Epinions is thus an ideal source for experiments on social trust recommendation.

The dataset used in our experiments is collected by crawling the Epinions.com site on Jan 2009. It consists of 51,670 users who have rated a total of 83,509 different items. The total number of ratings is 631,064. The density of the user-item rating matrix is less than 0.015%. We can observe that the user-item rating matrix of Epinions is very sparse, since the densities for the two most famous collaborative filtering datasets Movielens (6,040 users, 3,900 movies and 1,000,209 ratings) and Eachmovie (74,424 users, 1,648 movies and 2,811,983 ratings) are 4.25% and 2.29%, respectively. Moreover, an important factor that we choose the Epinions dataset is that user social trust network information is not included in the Movielens and Eachmovie datasets. The statistics of the Epinions user-item rating matrix is summarized in Table 6.1. As to the user social trust network, the total number of issued trust statements is 511,799. The statistics of this data source is summarized in Table 6.2.

6.2.2 Metrics

We use two metrics, the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE), to measure the prediction quality of our proposed approach in comparison with other collaborative filtering and trust-aware recommendation methods.

The metrics MAE is defined as:

$$MAE = \frac{\sum_{i,j} |r_{i,j} - \widehat{r}_{i,j}|}{N}, \tag{6.15}$$

where $r_{i,j}$ denotes the rating user i gave to item j, $\widehat{r}_{i,j}$ denotes the rating user i gave to item j as predicted by a method, and N denotes the number of tested ratings. The metrics RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum_{i,j}(r_{i,j} - \widehat{r}_{i,j})^2}{N}}. \tag{6.16}$$

6.2.3 Comparison

In this section, in order to show the performance improvement of our RSTE approach, we compare our method with the following approaches.

1. PMF: this method is proposed by Salakhutdinov and Minh in [99]. It only uses user-item matrix for the recommendations, and it is based on probabilistic matrix factorization.

2. Trust: this is the method purely uses trusted friends' tastes making recommendations. It is proposed in Section 6.1.3 in this chapter. It is also a special case of RSTE when $\alpha = 0$.

3. SoRec: this is the method proposed in [74]. It is a social trust-aware recommendation method that factorizes the user-item rating matrix and users' social trust network by sharing the same user latent space.

We use different amounts of training data (90%, 80%) to test the algorithms. Training data 90%, for example, means we randomly select 90% of the ratings from Epinions dataset as the training data to predict the remaining 10% of ratings. The random selection was carried out 5 times independently. The experimental results using 5 and 10 dimensions to represent the latent features are shown in Table 6.3.

The parameter settings of our approach are $\alpha = 0.4$ for both 90% training data and 80% training data, $\lambda_U = \lambda_V = 0.001$, and in all the experiments conducted in the following sections, we set all of the parameters λ_U, λ_V equal to 0.001. From Table 6.3, we can observe that our approach RSTE outperforms the other methods. In general, two social trust recommendation approaches SoRec and RSTE all perform better than the PMF method (only uses the user-item matrix for recommendations). However, the Trust method performs worse than the

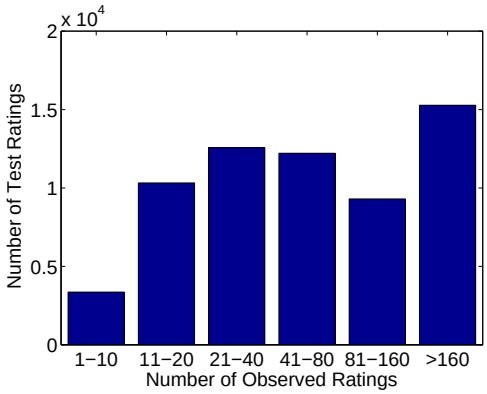
Table 6.3: Performance Comparisons (A Smaller MAE or RMSE Value Means a Better Performance)

Training Data	Metrics	Dimensionality = 5			
		Trust	PMF	SoRec	RSTE
90%	MAE	0.9054	0.8676	0.8442	**0.8377**
	RMSE	1.1959	1.1575	1.1333	**1.1109**
80%	MAE	0.9221	0.8951	0.8638	**0.8594**
	RMSE	1.2140	1.1826	1.1530	**1.1346**
Training Data	Metrics	Dimensionality = 10			
		Trust	PMF	SoRec	RSTE
90%	MAE	0.9039	0.8651	0.8404	**0.8367**
	RMSE	1.1917	1.1544	1.1293	**1.1094**
80%	MAE	0.9215	0.8886	0.8580	**0.8537**
	RMSE	1.2132	1.1760	1.1492	**1.1256**

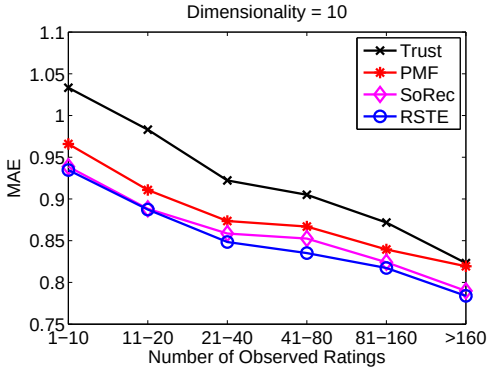
PMF method, which indicates purely utilizing trusted friends' tastes to recommend is not applicable. Among these three trust-aware recommendation methods, our RSTE method generally achieves better performance than the SoRec and Trust methods on both MAE and RMSE. This demonstrates that our interpretation on the formation of the ratings is realistic and reasonable.

6.2.4 Performance on Different Users

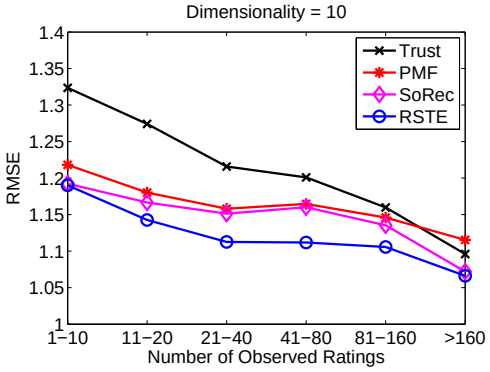
One challenge of the recommender systems is that it is difficult to recommend items to users who have very few ratings. Hence, in order to compare our approach with the other methods thoroughly, we first group all the users based on the number of observed ratings in the training data, and then evaluate prediction accuracies of different user groups. The experimental results are shown in Fig. 6.3. Users are grouped into 6 classes: "$1-10$", "$11-20$", "$21-40$", "$41-80$", "$81-160$" and "> 160", denoting how many ratings users have rated.



(a) Distribution of Testing Data (90% as Training Data)



(b) MAE Comparison on Different User Rating Scales (90% as Training Data)



(c) RMSE Comparison on Different User Rating Scales (90% as Training Data)

Figure 6.3: Performance Comparison on Different Users

Fig. 6.3(a) summarizes the distributions of testing data according to groups in the training data (90% as training data). For example, there are a total 3,360 user-item pairs to be predicted in the testing dataset in which the related users in the training dataset have rating numbers from 1 to 10. In Fig. 6.3(b) and Fig. 6.3(c), we observe that our RSTE algorithm consistently performs better than other methods, especially when few user ratings are given. When users' rating records are ranging from 1 to 80, our RSTE method performs much better than the Trust, PMF and SoRec approaches.

6.2.5 Impact of Parameter α

In our method proposed in this chapter, the parameter α balances the information from the users' own characteristics and their friends' favors. It controls how much our method should trust users themselves and their friends. If $\alpha = 1$, we only mine the user-item rating matrix for matrix factorization, and simply employ users' own tastes in making recommendations. If $\alpha = 0$, we only extract information from the social trust graph to predict users' preferences purely from the friends they trust. In other cases, we fuse information from the user-item rating matrix and the user social trust network for probabilistic matrix factorization and, furthermore, to predict ratings for the users.

Fig. 6.4 shows the impacts of α on MAE and RMSE. We observe that the value of α impacts the recommendation results significantly, which demonstrates that fusing the users' own tastes with their friends' favors greatly improves the recommendation accuracy. No matter using 90% training data or 80% training data, as α increases, the MAE and RMSE decrease (prediction accuracy increases) at first, but when α surpasses a certain threshold, the MAE and RMSE increase (prediction accuracy decreases) with further increase of the value

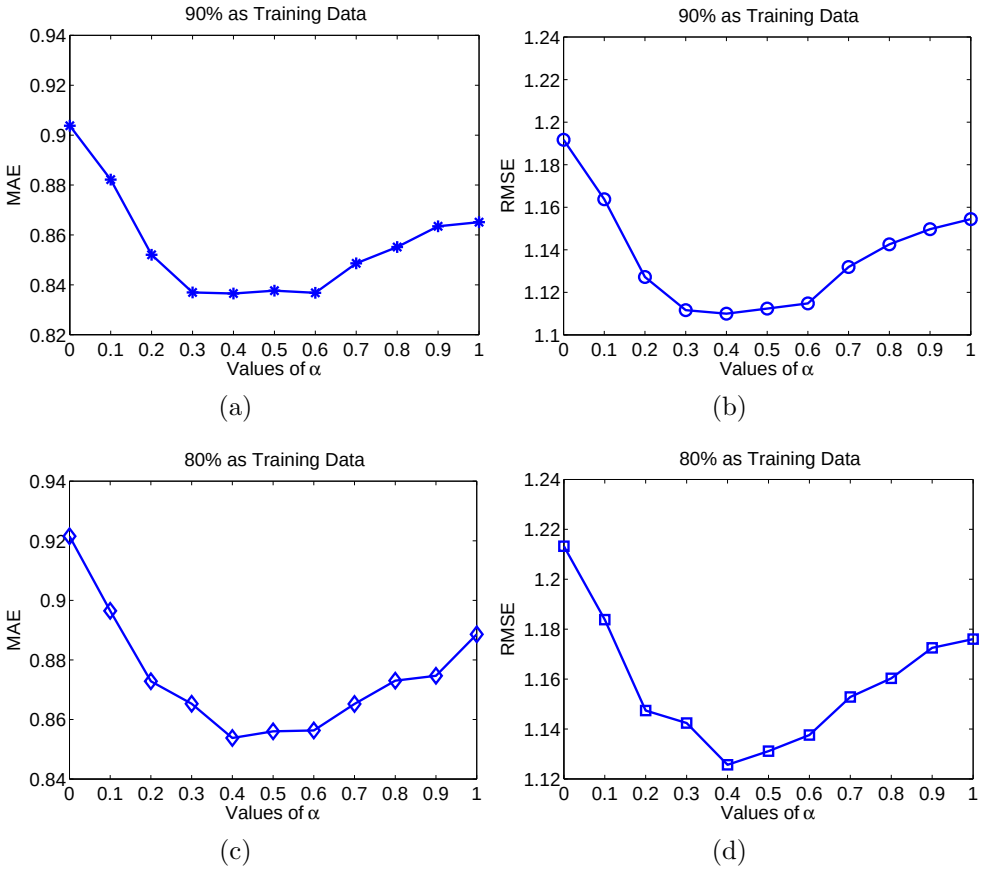
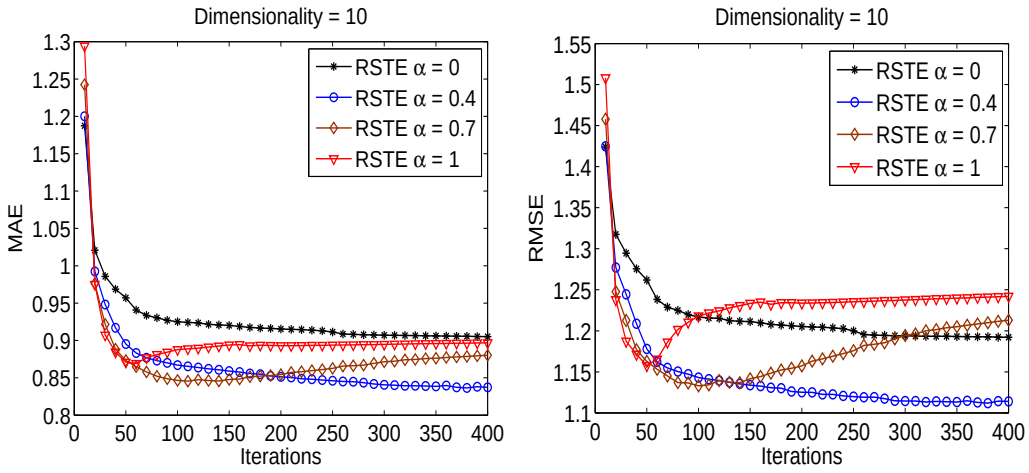


Figure 6.4: Impact of Parameter α (Dimensionality = 10)

of α. This phenomenon confirms with the intuition that purely using the user-item rating matrix or purely using the users' social trust network for recommendations cannot generate better performance than fusing these two favors together.

From Fig. 6.4(a) and Fig. 6.4(b), when using 90% ratings as training data, we observe that, our RSTE method achieves the best performance when α is around 0.4, while smaller values like $\alpha = 0.1$ or larger values like $\alpha = 0.7$ can potentially degrade the model performance. This indicates that we need to trust more about the tastes of users' trusted friends than their own tastes, since the training data of user-item matrix is very sparse, which can hardly learn the accurate characteristics of users. In Fig. 6.4(c) and Fig. 6.4(d), when using 80% ratings as train-



(a) (b)

Figure 6.5: Efficiency Analysis (90% as Training Data)

ing data, the optimal value of α is also around 0.4. However, less ratings for users will lead to an overall degradation of the recommendation results.

6.2.6 Training Efficiency Analysis

The complexity analysis in Section 6.1.5 states that the computational complexity of our approach is linear with respect to the number of ratings, which shows that our approach is scalable to very large datasets. Actually, our approach is very efficient even when using a very simple gradient descent method. In the experiments using 90% of the data as training data, our method only needs less than 400 iterations for training, and each iteration only requires less than 20 seconds. All the experiments are conducted on a normal personal computer containing an *Intel Pentium D* CPU (3.0 GHz, *Dual Core*) and 1G memory.

Fig. 6.5(a) and Fig. 6.5(b) show the performance (MAE and RMSE) changes with the iterations. We observe that when using a large value of α, such as $\alpha = 1$ or $\alpha = 0.7$, at the end of the training, the model begins to overfit (especially for the RMSE), while a relatively smaller α, such as $\alpha = 0$ or $\alpha = 0.4$, does not

have the overfitting problem. These experiments clearly demonstrate that in this dataset, an approach ignoring the social trust information can cause the overfitting problem, and that the predictive accuracy can be improved by incorporating appropriate amount of social trust information.

6.3 Summary

This chapter is motivated by the fact that a user's trusted friends on the Web will affect this user's online behavior. Based on the intuition that every user's decisions on the Web should include both the user's characteristics and the user's trusted friends' recommendations, we propose a novel, effective and efficient probabilistic matrix factorization framework for the recommender systems. Experimental analysis on the Epinions dataset shows the promising future of our proposed method. Moreover, the method introduced in this chapter by using probabilistic matrix factorization is not only working in trust-aware recommender systems, but also applicable to other popular research topics, such as social search, collaborative information retrieval, and social data mining.

In this chapter, although we employ the trusted friends' opinions in the social trust network to make recommendations for the users, we do not consider the possible diffusions of trusts between various users. Under the circumstance that both the user-item rating matrix and the trust relations of a social network are very sparse, the diffusions of trust relations become inevitable since this consideration will help to alleviate the data sparsity problem and will potentially increase the prediction accuracy. We plan to employ the diffusion processes in our future work.

In many popular applications on the Web, users not only can keep a list of trust relationships, but also have the rights to

establish a list of distrust or block relationships. If a user u_j is in the distrust list of a user u_i, most probably, it is because the user u_i thinks the user u_j's taste is totally different from him/her. Actually, this information is very useful on the recommender systems. Unfortunately, to the best of our knowledge, no previous work can employ this information well into recommender systems. The understanding of distrust relations is still unclear to the researchers: We cannot use diffusion methods to model it due to the reason that one person's enemy's enemy is not necessarily the enemy of this person. In the future, we plan to study the formation and nature of the distrust relations, and explicitly model them in the recommender systems.

□ **End of chapter.**

Chapter 7

Recommend with Social Distrust

Although we developed two trust-based recommendation approaches in this thesis, we ignored a very important information, i.e., distrust relations among users.

In this chapter, we propose a factor analysis framework with the constraints of distrust and trust relations among users. Our work is based on the following intuitions:

- Users' latent features can be extracted by factorizing the user-item rating matrix.
- Users' distrust relations can be interpreted as the "dissimilar" relations since user u_i distrusts user u_d indicates that user u_i disagrees with most of the opinions issued by user u_d.
- Users' trust relations can be modeled as the "similar" relations due to the reason that user u_i trusts user u_t means that user u_i agrees with most of the opinions issued by u_t.

7.1 Recommendation Framework

Previous recommender system techniques only utilize the information of the user-item rating matrix for recommendations

while ignoring the trust and distrust relationships among users. However, the fact is, trust and distrust information is very helpful in making the recommendations since to some extent, they represent the "similar" and "dissimilar" relationships. With the exponential growth of Web 2.0 Web sites, providing personalized recommendations and incorporating trust and distrust into traditional recommender systems are becoming more and more important.

In this section, we first describe the problem we study in Section 7.1.1, and then brief the matrix factorization technique for recommendation in Section 7.1.2. We provide solutions on how to incorporate the distrust and trust into recommendations in Section 7.1.3, Section 7.1.4 and Section 7.1.5. Finally, the complexity analysis is conducted in Section 7.1.6.

7.1.1 Problem Definition

Fig. 7.1(a) illustrates a typical Web user we will study in this chapter. In this figure, user u_1 rated three items v_1, v_3 and v_5. In addition to the rating data, this user also maintains two lists: trust list and distrust list. The trust list stores all the users that user u_1 trusts while the distrust list includes all the users that user u_1 distrusts.

By integrating all the information from all the users, we summarize three different data sources: the user-item rating matrix shown in Fig. 7.1(b), the user trust graph shown in Fig. 7.1(c) and the user distrust graph shown in Fig. 7.1(d). In this example, totally, there are 5 users (from u_1 to u_5) and 5 items (from v_1 to v_5) with 6 trust relations (edges) and 5 distrust relations between users. Each relation is associated with a weight w_{ij} in the range $(0, 1]$ to specify how much user u_i trusts or distrusts user u_j. In an online social network Web site, the weight w_{ij} is often explicitly stated by user u_i. Typically, each user also

(a) A Web User

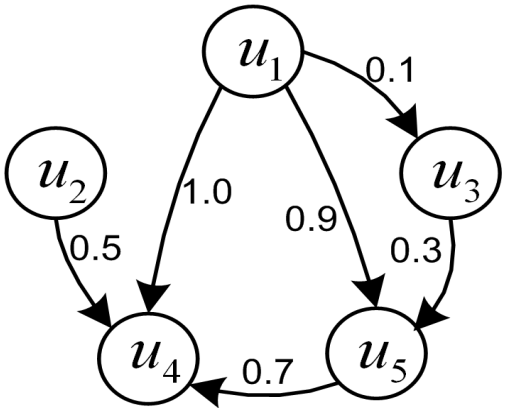
Rated Items: v_1, v_3, v_5

Trust List: u_3, u_4, u_5

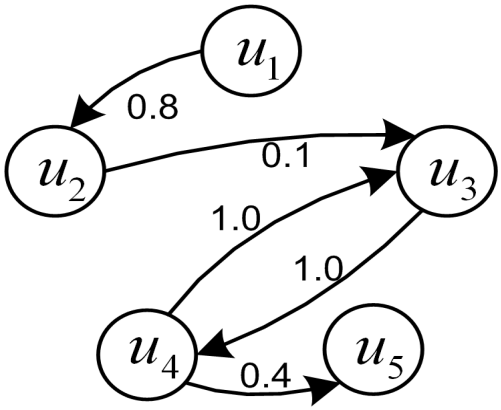
Distrust List: u_2

	v_1	v_2	v_3	v_4	v_5
u_1	5		3		5
u_2				1	
u_3		4			
u_4	3		4	2	
u_5		5			4

(b) User-Item Rating Matrix



(c) User Trust Graph



(d) User Distrust Graph

Figure 7.1: A Toy Example

rates some items on a 5-point integer scale to express the extent of the favor of each item (normally, 1, 2, 3, 4 and 5 represent "hate", "don't like", "neutral", "like" and "love", respectively).

The problem we study in this chapter is how to effectively and efficiently predict the missing values of the user-item matrix by employing these different data sources.

7.1.2 Matrix Factorization for Recommendation

A common and popular approach to recommender systems is to fit a factor model to the user-item rating matrix, and use it in order to make further predictions [48, 76, 94, 99]. The premise

behind a low-dimensional 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 [94].

Consider an $m \times n$ user-item rating matrix R, the matrix factorization method employs a rank-l matrix $X = U^T V$ to fit it, where $U \in \mathbb{R}^{l \times m}$ and $V \in \mathbb{R}^{l \times n}$. From the above definition, we can see that the low-dimensional matrices U and V are unknown, and need to be estimated. Moreover, this feature representations have clear physical meanings. In this linear factor model, each factor is a preference vector, and a user's preferences correspond to a linear combination of these factor vectors, with user-specific coefficients. More specifically, each row of U performs as a "feature vector", and each row of V is a linear predictor, predicting the entries in the corresponding column of R based on the "features" in U.

Actually, most recommender systems use integer rating values from 1 to R_{max} to represent the users' judgements on items. In this chapter, without loss of generality, we map the ratings $1, ..., R_{max}$ to the interval $[0, 1]$ using the function $f(x) = x/R_{max}$. However, simply employing $U_i^T V_j$ to predict the missing value $R_{i,j}$ can make the prediction outside of the range of valid rating values. Hence, instead of using a simple linear factor model, in this chapter, the inner product between user-specific and movie-specific feature vectors is mapped through a nonlinear logistic function $g(x) = 1/(1 + \exp(-x))$, which bounds the range of the predictions into [0, 1].

Hence, by adding the constraints of the norms of U and V, we have the following optimization problem:

$$\min_{U,V} \mathcal{L}(R,U,V) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{R}(R_{ij} - g(U_i^T V_j))^2 + \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2, \quad (7.1)$$

where I_{ij}^R is the indicator function that is equal to 1 if user u_i rated item v_j and equal to 0 otherwise, and $\|\cdot\|_F^2$ denotes the Frobenius norm.

The optimization problem in Eq. (7.1) minimizes the sum-of-squared-errors objective function with quadratic regularization terms. It also has a probabilistic interpretation with Gaussian observation noise, which is detailed in [99]. However, the same as many other collaborative filtering methods, this approach only utilizes the user-item rating matrix for the recommendations. In the following sections, we will introduce how to incorporate the distrust and trust information into the matrix factorization method.

7.1.3 Recommendation with Distrust Relations

In this section, we analyze how the distrust relationships can affect the recommendation processes.

Distrust is one of the most controversial topics and issues to cope with, especially when considering trust metrics and trust propagation [135]. Although many researchers have already conducted comprehensive studies on the trust related applications, the understanding of distrust relations is still unclear to the researchers. Distrust is totally different with trust, hence the method employed in the trust-aware recommender systems cannot be simply transplanted to distrust-aware recommender systems. For example, the most popular method in trust-aware recommender systems is to improve the recommendation quality by the propagation of trust; however, we cannot simply use

propagation methods to model distrust due to the reason that one person's enemy's enemy is not necessarily the enemy of this person.

However, we cannot ignore the distrust information since as reported in [42], experience with real-world implemented trust systems such as Epinions and eBay suggests that distrust is at least as important as trust.

In this chapter, we employ a simple intuition to make positive influence using distrust information. If a user u_d is in the distrust list of a user u_i, most probably, it is because the user u_i thinks the user u_d's taste is totally different from him/her. Actually, this information is very useful on the recommender systems. We could interpret this problem using the following intuition: if user u_i distrusts user u_d, then we could assume that the features U_i and U_d will have a large distance in the feature space. Based on this assumption, for all the users in the user space, we summarize the following optimization function:

$$\max_{U} \frac{1}{2} \sum_{i=1}^{m} \sum_{d \in \mathcal{D}^+(i)} S_{id}^{\mathcal{D}} \|U_i - U_d\|_F^2, \tag{7.2}$$

where $\mathcal{D}^+(i)$ is the set of users that user u_i distrusts, and $S_{id}^{\mathcal{D}} \in (0, 1]$ is the weight of distrust score that user u_i gives to user u_d. The larger the value of $S_{id}^{\mathcal{D}}$ is, the more the user u_i distrusts the user u_d.

Based on Eq. (7.1) and Eq. (7.2), we define the recommendation with distrust relations as the following optimization problem:

$$\begin{aligned}\min_{U,V} \mathcal{L}_{\mathcal{D}}(R, S^{\mathcal{D}}, U, V) &= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{R}(R_{ij} - g(U_i^T V_j))^2 \\ &+ \frac{\beta}{2}\sum_{i=1}^{m}\sum_{d\in\mathcal{D}^+(i)} (-S_{id}^{\mathcal{D}}\|U_i - U_d\|_F^2) \\ &+ \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2. \end{aligned} \tag{7.3}$$

In the online opinion sharing or recommender systems, the distrust value $S_{id}^{\mathcal{D}}$ is typically issued by user u_i explicitly with respect to user u_d, and it cannot accurately describe the relations between users since it contains noises and ignores the graph structure information of distrust network. For instance, similar to the Web link adjacency graph in [130], in a distrust graph, the confidence of distrust value $S_{id}^{\mathcal{D}}$ should be decreased if user u_i distrusts lots of users; however, the confidence of distrust value $S_{id}^{\mathcal{D}}$ should be increased if user u_d is trusted by lots of users. Hence, we propose to smooth the term $S_{id}^{\mathcal{D}}$ by incorporating local authority and local hub values in Eq. (7.3),

$$S_{id}^{\mathcal{D}} = \frac{\nabla^-(u_d)}{\nabla^+(u_i) + \nabla^-(u_d)} \times S_{id}^{\mathcal{D}}, \tag{7.4}$$

where $\nabla^+(u_i)$ represents the outdegree of user u_i in the distrust graph, while $\nabla^-(u_d)$ indicates the indegree of user u_d in the distrust graph.

A local minimum of the objective function given by Eq. (7.3) can be found by performing gradient descent in U_i, V_j,

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\mathcal{D}}}{\partial U_i} &= \sum_{j=1}^{n} I_{ij}^{R} g'(U_i^T V_j)(g(U_i^T V_j) - R_{ij}) V_j \\
&+ \beta \sum_{d \in \mathcal{D}^+(i)} S_{id}^{\mathcal{D}}(U_d - U_i) + \beta \sum_{p \in \mathcal{D}^-(i)} S_{pi}^{\mathcal{D}}(U_p - U_i) \\
&+ \lambda_U U_i, \\
\frac{\partial \mathcal{L}_{\mathcal{D}}}{\partial V_j} &= \sum_{i=1}^{m} I_{ij}^{R} g'(U_i^T V_j)(g(U_i^T V_j) - R_{ij}) U_i + \lambda_V V_j, \quad (7.5)
\end{aligned}$$

where $\mathcal{D}^-(i)$ is the set of users that distrust user u_i.

7.1.4 Recommendation with Trust Relations

In this section, we discuss how to incorporate the trust relationships into recommender systems. In order to model the trust relationships between users realistically, we first need to understand where the "trust" comes from. Actually, on the Web, it is not difficult to interpret the generation of trust relations. For example, in an opinion sharing Web site, if a user u_t is in the trust list of a user u_i, most probably, the underlying cause is that user u_i agrees with most of user u_t's opinions. Moreover, how much user u_i trusts user u_t depends on how much user u_i agrees with user u_t.

Based on the above interpretation, if user u_i trusts user u_t, we can assume that the feature representations U_i and U_d of these two users are close in the feature space. Following this intuition, we minimize the objective function

$$\min_{U} \frac{1}{2} \sum_{i=1}^{m} \sum_{t \in \mathcal{T}^+(i)} S_{it}^{\mathcal{T}} \|U_i - U_t\|_F^2, \quad (7.6)$$

where $\mathcal{T}^+(i)$ is the set of users that user u_i trusts, and $S_{it}^{\mathcal{T}} \in (0, 1]$ is the degree indicates how much user u_i trusts user u_t. The

larger the value of $S_{it}^{\mathcal{T}}$ is, the more the user u_i trusts the user u_t.

By employing Eq. (7.1) and Eq. (7.6), we define the recommendation problem with trust relations as the following optimization problems:

$$\begin{aligned}\min_{U,V} \mathcal{L}_{\mathcal{T}}(R, S^{\mathcal{T}}, U, V) &= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{R}(R_{ij} - g(U_i^T V_j))^2 \\ &+ \frac{\alpha}{2}\sum_{i=1}^{m}\sum_{t\in\mathcal{T}^+(i)} (S_{it}^{\mathcal{T}}\|U_i - U_t\|_F^2) \\ &+ \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2. \end{aligned} \tag{7.7}$$

Similar to Eq. (7.4), we also smooth the trust value $S_{it}^{\mathcal{T}}$ in Eq. (7.7) based on the following equation:

$$S_{it}^{\mathcal{T}} = \frac{\Delta^-(u_t)}{\Delta^+(u_i) + \Delta^-(u_t)} \times S_{it}^{\mathcal{T}}, \tag{7.8}$$

where $\Delta^+(u_i)$ represents the outdegree of user u_i in the trust graph, while $\Delta^-(u_t)$ indicates the indegree of user u_t in the trust graph.

In Eq. (7.7), by performing gradient descent in U_i, V_j, we have

$$\begin{aligned}\frac{\partial \mathcal{L}_{\mathcal{T}}}{\partial U_i} &= \sum_{j=1}^{n} I_{ij}^{R} g'(U_i^T V_j)(g(U_i^T V_j) - R_{ij})V_j \\ &+ \alpha \sum_{t\in\mathcal{T}^+(i)} S_{it}^{\mathcal{T}}(U_i - U_t) + \alpha \sum_{q\in\mathcal{T}^-(i)} S_{qi}^{\mathcal{T}}(U_i - U_q) \\ &+ \lambda_U U_i, \\ \frac{\partial \mathcal{L}_{\mathcal{T}}}{\partial V_j} &= \sum_{i=1}^{m} I_{ij}^{R} g'(U_i^T V_j)(g(U_i^T V_j) - R_{ij})U_i + \lambda_V V_j, \end{aligned} \tag{7.9}$$

where $\mathcal{T}^-(i)$ is the set of users that trust user u_i.

7.1.5 Prediction

After the low-dimensional latent feature spaces U and V are learned, the next step is to predict the ratings for the active users. For the given missing data R_{ij}, the value predicted by our method is defined as

$$\hat{R}_{ij} = g(U_i^T V_j). \tag{7.10}$$

We will evaluate the prediction quality in Section 7.2.

7.1.6 Complexity Analysis

The main computation of gradient methods is evaluating the object functions $\mathcal{L}_\mathcal{D}$, $\mathcal{L}_\mathcal{T}$ and their gradients against variables.

Because of the sparsity of matrices R, $S^\mathcal{D}$ and $S^\mathcal{T}$, the computational complexities of evaluating the objective functions $\mathcal{L}_\mathcal{D}$ are $\mathcal{L}_\mathcal{T}$ are $O(\rho_R l + m\overline{r}l)$ and $O(\rho_R l + m\overline{s}l)$, respectively, where ρ_R is the number of nonzero entries in the matrix R, l is the dimensions of the user feature, m is the number of users, $\overline{r}$ is the average number of users that a user distrusts, and $\overline{s}$ is the average number of friends that a user trusts. Since almost all of the online social network graphs fit the power-law distribution, a large long tail of users only have few trusted or distrusted users. This indicates that the values of $\overline{r}$ and $\overline{s}$ are relatively small. Generally, $m\overline{r} << \rho_R$ and $m\overline{s} << \rho_R$.

The computational complexities for the gradients $\frac{\partial \mathcal{L}_\mathcal{D}}{\partial U}$ and $\frac{\partial \mathcal{L}_\mathcal{D}}{\partial V}$ in Eq. (7.5) are $O(\rho_R l^2 + m(\overline{r} + \overline{r'})l)$ and $O(\rho_R l^2)$, respectively, where $\overline{r'}$ is the average number of users who distrust a user, which is also a small value. Actually, in a distrust network graph, the value of $\overline{r}$ is always equal to the value of $\overline{r'}$, which is 0.94 in the dataset we employ in the Section 7.2.

The computational complexities for the gradients $\frac{\partial \mathcal{L}_\mathcal{T}}{\partial U}$ and $\frac{\partial \mathcal{L}_\mathcal{T}}{\partial V}$ in Eq. (7.9) are $O(\rho_R l^2 + m(\overline{s} + \overline{s'})l)$ and $O(\rho_R l^2)$, respectively, where $\overline{s'}$ is the average number of friends who trust a

user. In a trust network graph, the value of $\overline{s}$ is also equal to the value of $\overline{s'}$, which is 5.45 in the dataset we employ in the experiments.

Therefore, the total computational complexity in one iteration is $O(\rho_R l + \rho_R l^2)$, which indicates that theoretically, the computational time of our method is linear with respect to the number of observations in the user-item matrix R. This complexity analysis shows that our proposed approach is very efficient and can scale to very large datasets.

7.2 Experimental Analysis

In this section, we conduct several experiments to compare the recommendation qualities of our approaches with other state-of-the-art collaborative filtering and trust-aware recommendation methods. Our experiments are intended to address the following questions:

1. How does our approach compare with the published state-of-the-art collaborative filtering and trust-aware recommendation algorithms?

2. How do the model parameter α and β affect the accuracy of prediction?

7.2.1 Dataset Description

We choose Epinions as the data source for our experiments on trust and distrust-aware recommendations. Epinions.com is a well known knowledge sharing site and review site, which was established in 1999. In order to add reviews, users (contributors) need to register for free and begin submitting their own personal opinions on topics such as products, companies, movies, or reviews issued by other users. Users can also assign products or

Table 7.1: Statistics of User-Item Rating Matrix of Epinions

Statistics	User	Item
Min. Num. of Ratings	1	1
Max. Num. of Ratings	162169	1179
Avg. Num. of Ratings	102.07	17.79

Table 7.2: Statistics of Trust Network of Epinions

Statistics	Trust per User	Be Trusted per User
Max. Num.	2070	3338
Avg. Num.	5.45	5.45

reviews integer ratings from 1 to 5. These ratings and reviews will influence future customers when they are about to decide whether a product is worth buying or a movie is worth watching. Every member of Epinions maintains a "trust" list which presents a network of trust relationships between users, and a "block (distrust)" list which presents a network of distrust relationships. This network is called the "Web of trust", and is used by Epinions to re-order the product reviews such that a user first sees reviews by users that they trust. Epinions is thus an ideal source for experiments on social recommendation.

The dataset used in our experiments consists of 131,580 users who have rated at least one of a total of 755,137 different items. The total number of ratings is 13,430,209. The density of the user-item matrix is 0.014%. We can observe that the user-item matrix of Epinions is very sparse, since the densities for the two most famous collaborative filtering datasets Movielens (6,040 users, 3,900 movies and 1,000,209 ratings) and Eachmovie (74,424 users, 1,648 movies and 2,811,983 ratings) are 4.25% and 2.29%, respectively. Moreover, an important reason that we choose the Epinions dataset is that user trust and distrust information is not included in the Movielens and Eachmovie datasets. The statistics of the Epinions user-item rating matrix is summarized

Table 7.3: Statistics of Distrust Network of Epinions

Statistics	Distrust per User	Be Distrusted per User
Max. Num.	1562	540
Avg. Num.	0.94	0.94

in Table 7.1.

As to the user trust network, the total number of issued trust statements is 717,129. The statistics of the this data source is summarized in Table 7.2. In the user distrust network, the total number of issued distrust statements is 123,670, and the statistics of the distrust data is summarized in Table 7.3.

We also observe a number of power-law distributions in these data sources, including items per user, trust relations per user (outdegree in the trust graph) and distrust relations per user (outdegree in the distrust graph). The distributions are shown in Fig. 7.2.

7.2.2 Metrics

We employ the Root Mean Square Error (RMSE) to measure the prediction quality of our proposed approaches in comparison with other collaborative filtering and trust-aware recommendation methods.

The metrics RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum_{i,j}(r_{i,j} - \widehat{r}_{i,j})^2}{N}}. \tag{7.11}$$

where $r_{i,j}$ denotes the rating user i gave to item j, $\widehat{r}_{i,j}$ denotes the rating user i gave to item j as predicted by a method, and N denotes the number of tested ratings.

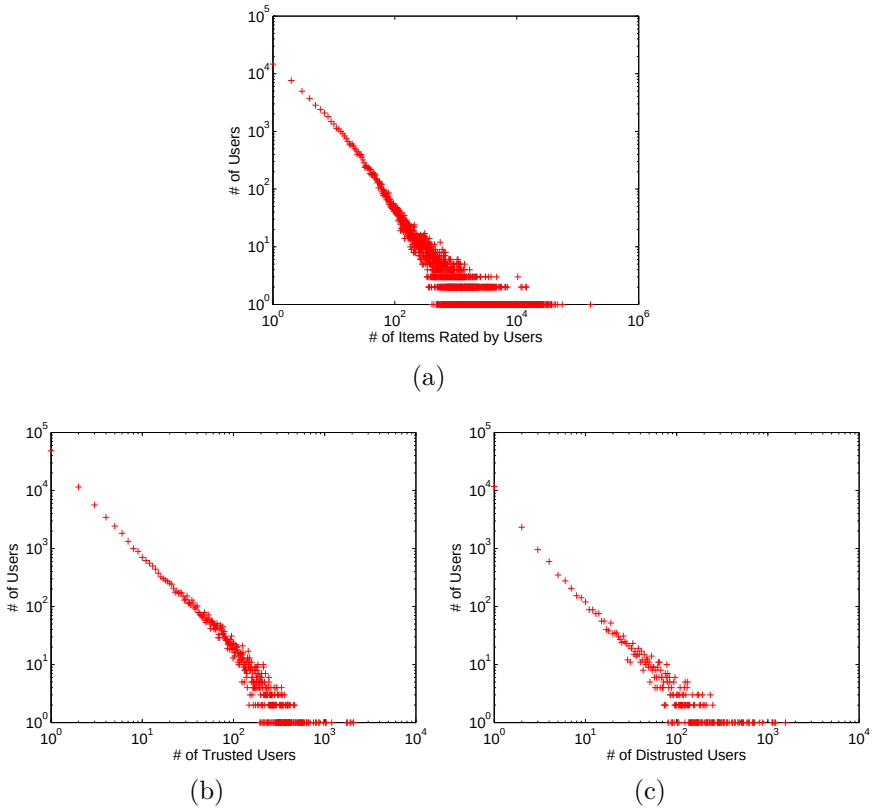


Figure 7.2: Power-Law Distributions of the Epinions Dataset. (a) Items per User Distribution. (b) Trust Graph Outdegree Distribution. (c) Distrust Graph Outdegree Distribution.

7.2.3 Comparison

In this section, in order to show the effectiveness of our proposed recommendation approaches, we compare the recommendation results of the following methods:

1. PMF (Probabilistic Matrix Factorization): this method is proposed by Salakhutdinov and Minh in [99]. It only uses user-item matrix for the recommendations.

2. SoRec (Social Recommendation): this is the method proposed in [74]. It is a trust-aware recommendation method that factorizes the user-item rating matrix and users' trust network by sharing the same user latent space.

3. RWD (Recommendation With Distrust): this is a matrix factorization-based recommendation method with distrust constraints. It is proposed in Section 7.1.3 in this chapter.

4. RWT (Recommendation With Trust): this is a matrix factorization-based recommendation method with trust constraints. It is proposed in Section 7.1.4 in this chapter.

As to the training data, we employ three settings: 5%, 10% and 20% for training, where 20% means we randomly select 20% ratings as training data to predict the remaining 80% ratings.

In our RWD and RWT methods, there are totally four parameters need to be set, including α, β, λ_U and λ_V. Without loss of generality, in order to reduce the model complexity, we set $\lambda_U = \lambda_V = 0.001$ in all the experiments we conduct in this chapter. We will discuss the influence of the parameters α and β in the experiments conducted in Section 7.2.4.

The prediction accuracies evaluated by RMSE are shown in Table 7.4. In our proposed distrust-aware recommendation method RWD, the parameter β is set to be 0.00001 while in our

Table 7.4: RMSE Comparison with other popular algorithms. The reported values are the RMSE on the Epinions Dataset achieved from dividing the data into 5%, 10%, and 20% for training data, respectively.

Dataset	Traning Data	Dimensionality	PMF	SoRec	RWD	RWT
Epinions	5%	5D	1.228	1.199	1.186	**1.177**
		10D	1.214	1.198	1.185	**1.176**
	10%	5D	0.990	0.944	0.932	**0.924**
		10D	0.977	0.941	0.931	**0.923**
	20%	5D	0.819	0.788	0.723	**0.721**
		10D	0.818	0.787	0.723	**0.720**

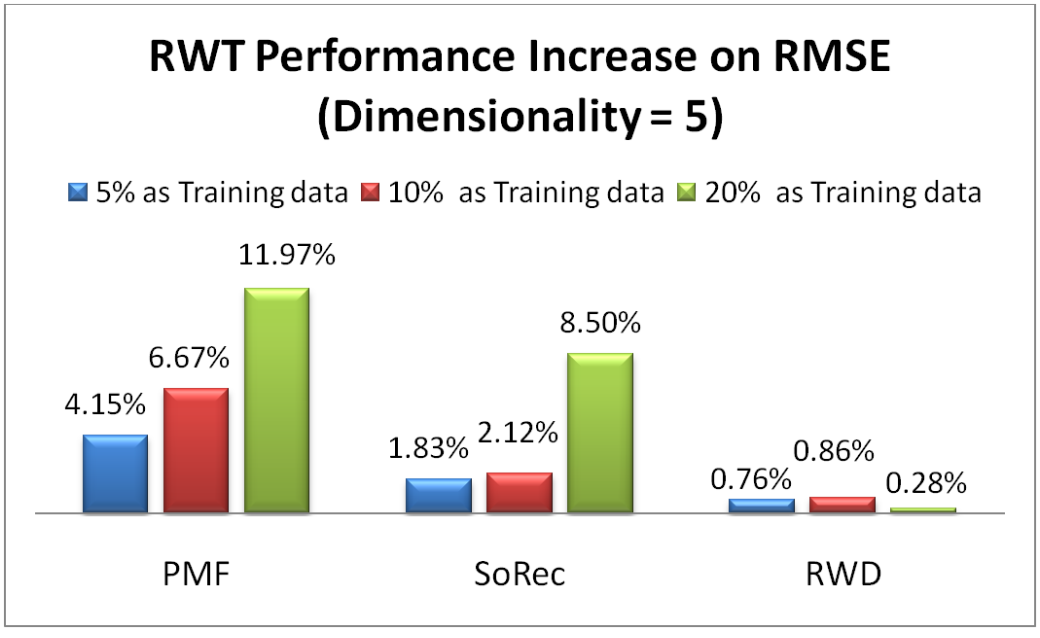


Figure 7.3: RWT Performance Increase (5D)

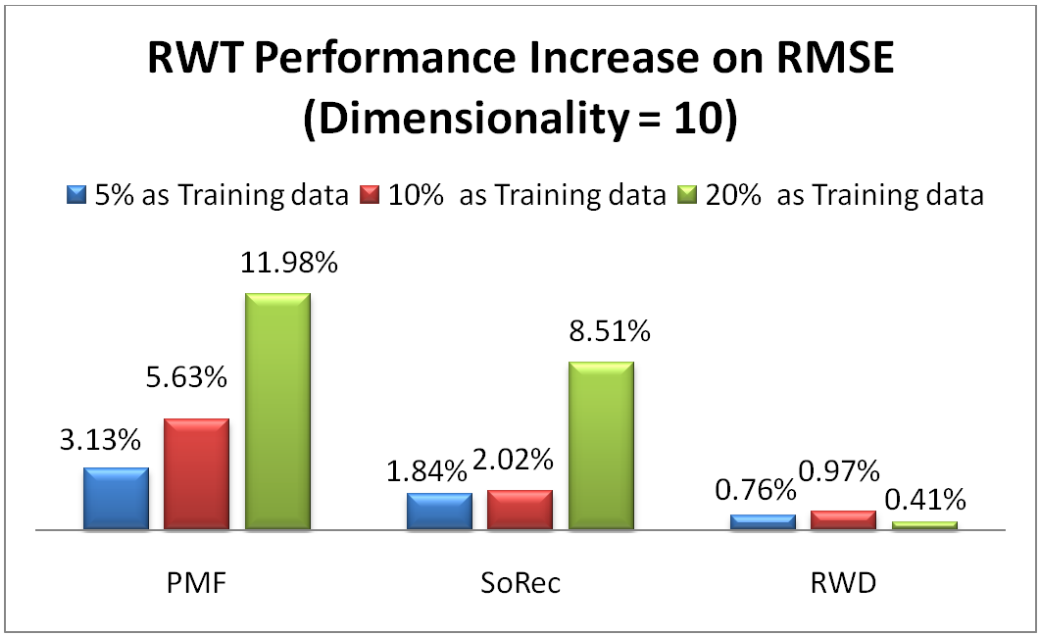
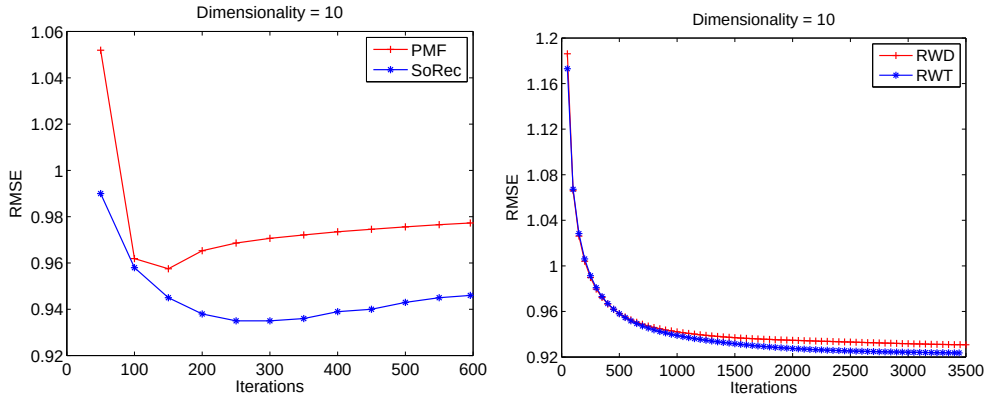


Figure 7.4: RWT Performance Increase (10D)

trust-aware recommendation method RWT, the parameter α is set to be 0.001.

From Table 7.4, we can observe that our RWD and RWT approaches constantly performs better than the other methods in all the settings. When we use 20% as training data, we find that our method generates much better performance than PMF and SoRec. This demonstrates the advantages of trust and distrust-aware recommendation algorithms.

In Fig. 7.3 and Fig. 7.4, we also plot the percentages of performance increase of our RWT algorithm against PMF, SoRec as well as our RWD algorithms in terms of RMSE. From these figures, we observe an interesting phenomenon: as the sparsity of the data decreases, the percentages of performance increase against PMF and SoRec keep increasing. This observation is reasonable since in the very spare training settings like 5% and 10%, the user features cannot be accurately learned since the training sample is very sparse. Hence our optimization methods cannot maximize the influences of the trust and distrust con-



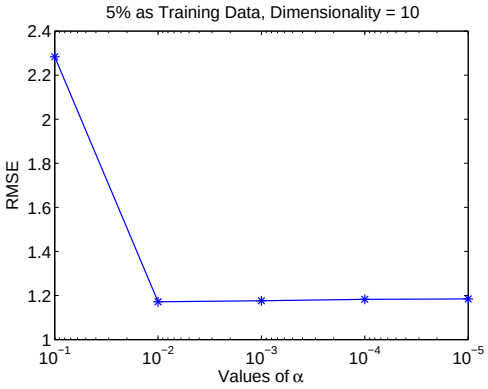
(a) (b)

Figure 7.5: Efficiency Analysis (10% as Training Data). (a) RMSEs of PMF and SoRec Change with Iterations. (b) RMSEs of RWD and RWT Change with Iterations ($\alpha = 0.001$, $\beta = 0.00001$).

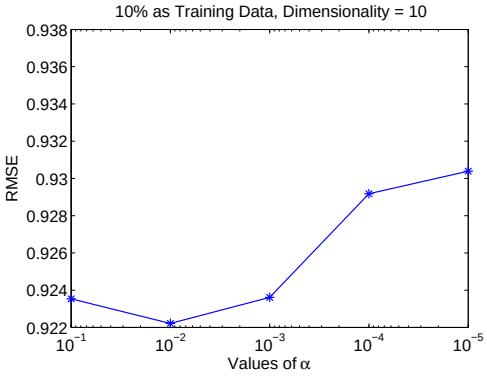
straints. But as the increase of the training data, RWD and RWT performs better and better.

We also observe another phenomenon worthy of studying. We find that the distrust-based method RWD performs almost as good as the trust-based method RWT (Please notice that in Table 7.2 and Table 7.3, in average, every user only has 0.94 distrusted users while has 5.45 trusted users). This observation proves that the distrust information among users is as important as the trust information in the recommender systems.

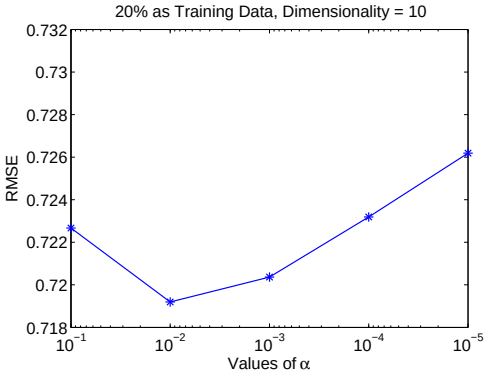
In Fig. 7.5, we plot the performance (RMSE) changes with the iterations. We observe that in the PMF and SoRec methods, at the end of the training, the models begin to overfit, as shown in Fig. 7.5(a), while our RWD and RWT methods do not have the overfitting problem, as illustrated in Fig. 7.5(b). These experiments clearly demonstrate that in this dataset, the employ of our trust and distrust regularization terms not only generates better performance than other methods, but also avoids the overfitting problem.



(a) 5% as Training Data



(b) 10% as Training Data



(c) 20% as Training Data

Figure 7.6: Impact of Parameter α

7.2.4 Impact of Parameters α and β

In our method proposed in this chapter, the parameters α and β play very important roles. They control how much our method should use the information of trusted or distrusted users. In the extreme case, if we use a very small value of α or β, we only mine the user-item rating matrix for matrix factorization, and simply employ users' own tastes in making recommendations. On the other side, if we employ a very large value of α or β, the trust or distrust information will dominate the learning processes. In normal cases, we integrate information from the user-item rating matrix and the users' trust or distrust network for matrix factorization and, furthermore, to predict ratings for the users.

Fig. 7.6 shows the impacts of α on RMSE. We observe that the value of α impacts the recommendation results significantly, which demonstrates that incorporating the trust information greatly improves the recommendation accuracy. No matter using 5% training data, 10% training data or 20% training data, as α increases, the RMSE decrease (prediction accuracy increases) at first, but when α surpasses a certain threshold like 0.01, the RMSE increase (prediction accuracy decreases) with further increase of the value of α. The existence of the yielding point confirms with the intuition that purely using the user-item rating matrix or purely using the users' trust information for recommendations cannot generate better performance than appropriately integrating these two sources together.

The impact of β generally shares the same trend as the impact of α. The difference is that we should choose a relatively small value of β, since if we choose a large value, the optimization problem in Eq. (7.3) will become unbounded, hence we cannot find the solutions.

7.3 Summary

In this chapter, we systematically study how to effectively and efficiently incorporate the trust and distrust information into the recommender systems. Our proposed framework is based on matrix factorization with regularization terms constraining the trust and distrust relations between users. The complexity of our proposed optimization framework is linear with the observations of the ratings, and the experimental analysis on a large Epinions dataset shows that our RWD and RWT methods outperforms other state-of-the-arts algorithms. Based on the experimental analysis, we also draw the conclusion that the distrust information is at least as important as the trust information. This observation brings a major contribution to the research of trust and distrust-aware applications since it proves that the distrust information can also be utilized to influence online applications in a positive fashion.

In this chapter, the trust and distrust constraints are regularized separately. In order to generate better prediction quality, a possible improvement is to fuse these two data sources into the same objective function. The most direct method is simply attaching the constraints in Eq. (7.2) and Eq. (7.6) to the objective function in Eq. (7.1). However, this will increase the model complexity, hence a more flexible and efficient method needs to be designed in the future.

□ **End of chapter.**

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In this chapter, we provide a summary of the thesis. The thesis consists of two parts: the first part deals with traditional recommender systems while the second part focuses on social-based recommender systems. All of the approaches proposed in this thesis are aiming at alleviating the data sparsity problems in recommender systems.

In the first part, we first present an effective missing data prediction method for collaborative filtering, which is a memory-based method. In order to improve the recommendation performance, we predict some of the missing data, and utilize the enriched data for further prediction. The second method in this part is a model-based method which utilizes matrix factorization technique to constrain the mean of the testing data with the mean of the training data. The experimental results show that this method can generate better results.

In the second part of this thesis, thanks to the many Web 2.0 Web sites, we propose three methods to incorporate social contextual information into traditional recommender systems. For trust relations, the underlying assumption we make is that online users can be very easily influenced by the friends they trust, and prefer their friends' recommendations. For distrust

relations, we model them as "dissimilar" relations since user u_i distrusts user u_d indicates that user u_i disagrees with most of the opinions issued by user u_d.

In general, the goal of our work is to model the real world recommendation as realistic as possible. Our proposed social recommendation framework opens a new direction for other researchers.

8.2 Future Work

There has several research directions we can follow in both traditional and social-based recommender systems in the future.

For traditional methods, we plan to conduct more research on the relationship between user information and item information since our simulations show the algorithm combining these two kinds of information generates better performance. Another direction worth of investigation is how design a method to take advantages of both memory-based and model-based methods.

For the social based methods, currently, we only use social trust and distrust information to improve recommendations. However these two types of relations are still different with the "Friend" relation, such as friend relations in Facebook [1]. To achieve the final goal of social recommendation, we need to utilize social friend data instead of social trust data for recommendation.

As the exponential growth of online social network sites continues, the research of social search is becoming more and more important. We also plan to develop similar techniques to allow users' trusted friends to influence the users' search results or query suggestions. The intuition behind this is that if a large number of our friends are searching for something, it's likely

[1] http://www.facebook.com

that we may be interested in that topic too. This would be an interesting search phenomenon to explore in social networks.

The Web is now leaving the era of search and entering one of discovery. Search is what you do when you are looking for something. Discovery is when something wonderful that you do not know existed, or do not know how to ask for, finds you. Hence, in the future, another promising research topic would be how to actually extend recommendations techniques to search problems. If we can accurately model users' search behaviors, we believe we can also design accurate personalized results for all the online users.

□ **End of chapter.**

Bibliography

[1] A. Abdul-Rahman and S. Hailes. A distributed trust model. In *NSPW '97: Proceedings of the 1997 workshop on New security paradigms*, pages 48–60, New York, NY, USA, 1997. ACM.

[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 Trans. Knowl. Data Eng.*, 17(6):734–749, 2005.

[3] R. Andersen, C. Borgs, J. Chayes, U. Feige, A. Flaxman, A. Kalai, V. Mirrokni, and M. Tennenholtz. Trust-based recommendation systems: an axiomatic approach. In *Proc. of WWW '08*, pages 199–208, New York, NY, USA, 2008. ACM.

[4] J. S. Armstrong. *Principles of Forecasting - A Handbook for Researchers and Practitioners*. Kluwer Academic, 2001.

[5] R. A. Baeza-Yates, C. A. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. In *EDBT Workshops*, pages 588–596, 2004.

[6] A. Banerjee, C. Krumpelman, J. Ghosh, S. Basu, and R. J. Mooney. Model-based overlapping clustering. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international*

conference on Knowledge discovery in data mining, pages 532–537, New York, NY, USA, 2005. ACM.

[7] S. Banerjee and K. Ramanathan. Collaborative filtering on skewed datasets. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 1135–1136, New York, NY, USA, 2008. ACM.

[8] P. Bedi, H. Kaur, and S. Marwaha. Trust based recommender system for semantic web. In *IJCAI'07: Proceedings of International Joint Conferences on Artificial Intelligence*, pages 2677–2682, 2007.

[9] R. Bell, Y. Koren, and C. Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proc. of KDD '07*, pages 95–104, New York, NY, USA, 2007. ACM.

[10] R. Bell, Y. Koren, and C. Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 95–104, New York, NY, USA, 2007. ACM.

[11] R. M. Bell and Y. Koren. Lessons from the netflix prize challenge. *SIGKDD Explor. Newsl.*, 9(2):75–79, 2007.

[12] R. M. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, pages 43–52, Washington, DC, USA, 2007. IEEE Computer Society.

[13] D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *ICML*, pages 46–54, 1998.

[14] D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 46–54, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[15] P. Bonhard, M. A. Sasse, and C. Harries. "the devil you know knows best": how online recommendations can benefit from social networking. In *BCS-HCI '07: Proceedings of the 21st British CHI Group Annual Conference on HCI 2007*, pages 77–86, Swinton, UK, UK, 2007. British Computer Society.

[16] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 43–52, San Francisco, 1998. Morgan Kaufmann.

[17] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI'98: Proceedings of Uncertainty in Artificial Intelligence*, 1998.

[18] C. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. In *NIPS*, pages 193–200, 2006.

[19] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 89–96, New York, NY, USA, 2005. ACM.

[20] J. Canny. Collaborative filtering with privacy via factor analysis. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and de-*

velopment in information retrieval, pages 238–245, New York, NY, USA, 2002. ACM.

[21] B. Cao, J.-T. Sun, J. Wu, Q. Yang, and Z. Chen. Learning bidirectional similarity for collaborative filtering. In *ECML PKDD '08: Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*, pages 178–194, Berlin, Heidelberg, 2008. Springer-Verlag.

[22] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 129–136, New York, NY, USA, 2007. ACM.

[23] M. Carbone, M. Nielsen, and V. Sassone. A formal model for trust in dynamic networks. In *IN PROC. OF INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND FORMAL METHODS (SEFM03*, pages 54–63. Society Press, 2003.

[24] Y. H. Chien and E. I. George. A bayesian model for collaborative filtering. In *Proceedings of the Seventh International Workshop Artificial Intelligence and Statistics*, 1999.

[25] P. A. Chirita, C. S. Firan, and W. Nejdl. Personalized query expansion for the web. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 7–14, New York, NY, USA, 2007. ACM.

[26] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *WWW '07: Proceedings of the 16th international*

conference on World Wide Web, pages 271–280, New York, NY, USA, 2007. ACM.

[27] J. Delgado and N. Ishii. Memory-based weighted-majority prediction for recommender systems. In *SIGIR '99: Proceedings of the SIGIR Workshop Recommender Systems: Algorithms and Evaluations*, 1999.

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

[29] P. Domingos. Mining social networks for viral marketing. *IEEE Intelligent Systems*, 20(1):80–82, 2005.

[30] P. Domingos and M. Richardson. Mining the network value of customers. In *Proc. of the ACM SIGKDD Conf.*, pages 57–66, 2001.

[31] D. Dueck and B. Frey. Probabilistic sparse matrix factorization. In *Technical Report PSI TR 2004-023*, Dept. of Computer Science, University of Toronto, 2004.

[32] G. Dupret and M. Mendoza. Automatic query recommendation using click-through data. In *IFIP PPAI*, pages 303–312, 2006.

[33] D. Gambetta. Can we trust trust? In *Trust: Making and Breaking Cooperative Relations*, pages 213–237. Basil Blackwell, 1988.

[34] W. Gao, C. Niu, J.-Y. Nie, M. Zhou, J. Hu, K.-F. Wong, and H.-W. Hon. Cross-lingual query suggestion using query logs of different languages. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 463–470, New York, NY, USA, 2007. ACM.

[35] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 625–628, Washington, DC, USA, 2005. IEEE Computer Society.

[36] L. Getoor and M. Sahami. Using probabilistic relational models for collaborative filtering. In *In Workshop on Web Usage Analysis and User Profiling (WEBKDD'99*, 1999.

[37] D. Gleich and L. Zhukov. Svd subspace projections for term suggestion ranking and clustering. In *Technical Report of Yahoo! Research Labs*, 2004.

[38] J. Golbeck. Generating predictive movie recommendations from trust in social networks. In *iTrust*, pages 93–104, 2006.

[39] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, 1992.

[40] K. Y. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Inf. Retr.*, 4(2):133–151, 2001.

[41] Z. Guan, J. Bu, Q. Mei, C. Chen, and C. Wang. Personalized tag recommendation using graph-based ranking on multi-type interrelated objects. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 540–547, New York, NY, USA, 2009. ACM.

[42] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 403–412, New York, NY, USA, 2004. ACM.

[43] J. Herlocker, J. A. Konstan, and J. Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information Retrieval*, 5:287–310, 2002.

[44] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, New York, NY, USA, 1999. ACM.

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

[46] W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[47] T. Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 259–266, New York, NY, USA, 2003. ACM.

[48] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22(1):89–115, 2004.

[49] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intel-*

ligence, pages 688–693, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[50] Z. Huang, H. Chen, and D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):116–142, 2004.

[51] S.-Y. Hwang and L.-S. Chen. Using trust for collaborative filtering in ecommerce. In *ICEC '09: Proceedings of the 11th International Conference on Electronic Commerce*, pages 240–248, New York, NY, USA, 2009. ACM.

[52] J. Jeon, W. B. Croft, and J. H. Lee. Finding similar questions in large question and answer archives. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 84–90, New York, NY, USA, 2005. ACM.

[53] R. Jin, J. Y. Chai, and L. Si. An automatic weighting scheme for collaborative filtering. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 337–344, New York, NY, USA, 2004. ACM.

[54] R. Jin, L. Si, and C. Zhai. A study of mixture models for collaborative filtering. *Inf. Retr.*, 9(3):357–382, 2006.

[55] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, 43(2):618–644, 2007.

[56] A. Kohrs and B. Merialdo. Clustering for collaborative filtering applications. In *Proceedings of CIMCA*, 1999.

[57] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: applying collabo-

rative filtering to usenet news. *Commun. ACM*, 40(3):77–87, 1997.

[58] J. Koren, Y. Zhang, and X. Liu. Personalized interactive faceted search. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 477–486, New York, NY, USA, 2008. ACM.

[59] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proc. of KDD '08*, pages 426–434, New York, NY, USA, 2008. ACM.

[60] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, New York, NY, USA, 2008. ACM.

[61] Y. Koren. Collaborative filtering with temporal dynamics. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456, New York, NY, USA, 2009. ACM.

[62] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Recommendation systems: A probabilistic analysis. *J. Comput. Syst. Sci.*, 63(1):42–61, 2001.

[63] M. Kurucz, A. A. Benczur, and K. Csalogany. Methods for large scale svd with missing values. In *Proceedings of KDD Cup and Workshop*, 2007.

[64] G. L. Lilien, P. Kotler, and K. S. Moorthy. *Marketing Models*. Prentice Hall, 1992.

[65] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, pages 76–80, Jan/Feb 2003.

[66] N. N. Liu and Q. Yang. Eigenrank: a ranking-oriented approach to collaborative filtering. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 83–90, 2008.

[67] Y. Liu, B. Gao, T.-Y. Liu, Y. Zhang, Z. Ma, S. He, and H. Li. Browserank: letting web users vote for page importance. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 451–458, New York, NY, USA, 2008. ACM.

[68] H. Ma, I. King, and M. R. Lyu. Effective missing data prediction for collaborative filtering. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 39–46, New York, NY, USA, 2007. ACM.

[69] H. Ma, I. King, and M. R. Lyu. Learning to recommend with social trust ensemble. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 203–210, New York, NY, USA, 2009. ACM.

[70] H. Ma, M. R. Lyu, and I. King. Learning to recommend with trust and distrust relationships. In *RecSys '09: Proceeding of the 3rd ACM conference on Recommender Systems*, New York, NY, USA, 2009. ACM.

[71] H. Ma, H. Yang, I. King, and M. R. Lyu. Learning latent semantic relations from clickthrough data for query suggestion. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 709–718, New York, NY, USA, 2008. ACM.

[72] H. Ma, H. Yang, I. King, and M. R. Lyu. Semi-nonnegative matrix factorization with global statistical consistency in collaborative filtering. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, New York, NY, USA, 2009. ACM.

[73] H. Ma, H. Yang, M. R. Lyu, and I. King. Mining social networks using heat diffusion processes for marketing candidates selection. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 233–242, New York, NY, USA, 2008. ACM.

[74] H. Ma, H. Yang, M. R. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 931–940, New York, NY, USA, 2008. ACM.

[75] B. Marlin. Modeling user rating profiles for collaborative filtering. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[76] B. Marlin and R. S. Zemel. The multiple multiplicative factor model for collaborative filtering. In *Proc. of ICML '04*, page 73, Banff, Alberta, Canada, 2004.

[77] P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. In *Proceedings of CoopIS/DOA/ODBASE*, pages 492–508, 2004.

[78] P. Massa and B. Bhattacharjee. Using trust in recommender systems: An experimental analysis. In *iTrust*, pages 221–235, 2004.

[79] M. R. McLaughlin and J. L. Herlocker. A collaborative filtering algorithm and evaluation metric that accurately

model the user experience. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 329–336, New York, NY, USA, 2004. ACM.

[80] B. P. S. Murthi and S. Sarkar. The role of the management sciences in research on personalization. *Management Science*, 49(10):1344–1362, 2003.

[81] A. Nakamura and N. Abe. Collaborative filtering using weighted majority prediction algorithms. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 395–403, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[82] J. O'Donovan and B. Smyth. Eliciting trust values from recommendation errors. In *FLAIRS Conference*, pages 289–294, 2005.

[83] J. O'Donovan and B. Smyth. Trust in recommender systems. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 167–174, New York, NY, USA, 2005. ACM.

[84] J. O'Donovan and B. Smyth. Is trust robust?: an analysis of trust-based recommendation. In *IUI*, pages 101–108, 2006.

[85] M. Papagelis, D. Plexousakis, and T. Kutsuras. Alleviating the sparsity problem of collaborative filtering using trust inferences. In *iTrust*, pages 224–239, 2005.

[86] D. Pavlov, E. Manavoglu, D. M. Pennock, and C. L. Giles. Collaborative filtering with maximum entropy. *IEEE Intelligent Systems*, 19(6):40–48, 2004.

[87] D. Pavlov and D. M. Pennock. A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains. In *NIPS*, pages 1441–1448, 2002.

[88] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *Proceedings of of UAI*, 2000.

[89] G. Pitsilis and L. Marshall. A model of trust derivation from evidence for use in recommendation systems. In *Technical Report Series, CS-TR-874*. University of Newcastle Upon Tyne, 2004.

[90] G. Pitsilis and L. Marshall. Trust as a key to improving recommendation systems. In *iTrust*, pages 210–223, 2005.

[91] M. J. D. Powell. *Approximation Theory and Methods*. Cambridge Univ. Press, 1981.

[92] T. Qin, T.-Y. Liu, X.-D. Zhang, D.-S. Wang, W.-Y. Xiong, and H. Li. Learning to rank relational objects and its application to web search. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 407–416, New York, NY, USA, 2008. ACM.

[93] D. Quercia, S. Hailes, and L. Capra. B-trust: Bayesian trust framework for pervasive computing. In *iTrust*, pages 298–312, 2006.

[94] J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *ICML '05: Proceedings of the 22th International Conference on Machine Learning*, 2005.

[95] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collabora-

tive filtering of netnews. In *Proceedings of ACM Conference on Computer Supported Cooperative Work*, 1994.

[96] E. Rich. User modeling via stereotypes. pages 329–342, 1998.

[97] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proc. of the ACM SIGKDD Conf.*, pages 61–70, 2002.

[98] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML '08: Proceedings of the 25th International Conference on Machine Learning*, 2008.

[99] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

[100] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 791–798, New York, NY, USA, 2007. ACM.

[101] G. Salton. *Automatic Text Processing*. Addison-Wesley, 1989.

[102] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM.

[103] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender

system - a case study. In *In ACM WebKDD Workshop*, 2000.

[104] E. Savia, K. Puolam?ki, J. Sinkkonen, and S. Kaski. Two-way latent grouping model for user preference prediction. In *In Proceedings of the UAI'05*, pages 518–525. AUAI Press, 2005.

[105] G. Shani, R. I. Brafman, and D. Heckerman. An mdp-based recommender system. In *UAI*, pages 453–460, 2002.

[106] U. Shardanand and P. Maes. Social information filtering: algorithms for automating "word of mouth". In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[107] L. Si and R. Jin. Flexible mixture model for collaborative filtering. In *ICML '03: Proceedings of the 20th International Conference on Machine Learning*, 2003.

[108] B. Sigurbjörnsson and R. van Zwol. Flickr tag recommendation based on collective knowledge. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 327–336, New York, NY, USA, 2008. ACM.

[109] P. Singla and M. Richardson. Yes, there is a correlation - from social networks to personal behavior on the web. In *WWW '08: Proceedings of the 17th international conference on World Wide Web*, pages 655–664, New York, NY, USA, 2008. ACM.

[110] R. R. Sinha and K. Swearingen. Comparing recommendations made by online systems and friends. In *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, 2001.

[111] X. Song, B. L. Tseng, C.-Y. Lin, and M.-T. Sun. Personalized recommendation driven by information flow. In *Proc. of the ACM SIGIR Conf.*, pages 509–516, 2006.

[112] Y. Song, Z. Zhuang, H. Li, Q. Zhao, J. Li, W.-C. Lee, and C. L. Giles. Real-time automatic tag recommendation. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 515–522, New York, NY, USA, 2008. ACM.

[113] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *ICML '03: Proceedings of the 20th International Conference on Machine Learning*, pages 720–727, 2003.

[114] N. Srebro, J. D. M. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. In *NIPS*, 2004.

[115] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[116] L. Ungar, D. Foster, E. Andre, S. Wars, F. S. Wars, D. S. Wars, and J. H. Whispers. Clustering methods for collaborative filtering. AAAI Press, 1998.

[117] L. Ungar and D. P. Foster. A formal statistical approach to collaborative filtering. In *In CONALD98*, 1998.

[118] F. E. Walter, S. Battiston, and F. Schweitzer. A model of a trust-based recommendation system on a social network. *Autonomous Agents and Multi-Agent Systems*, 16(1):57–74, 2008.

[119] J. Wang, A. P. de Vries, and M. J. T. Reinders. Unifying user-based and item-based collaborative filtering ap-

proaches by similarity fusion. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508, New York, NY, USA, 2006. ACM.

[120] J. Wang, A. P. de Vries, and M. J. T. Reinders. A user-item relevance model for log-based collaborative filtering. In *ECIR*, pages 37–48, 2006.

[121] J. Wang, A. P. de Vries, and M. J. T. Reinders. Unified relevance models for rating prediction in collaborative filtering. *ACM Trans. Inf. Syst.*, 26(3):1–42, 2008.

[122] J. Weng, C. Miao, and A. Goh. Improving collaborative filtering with trust-based metrics. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 1860–1864, New York, NY, USA, 2006. ACM.

[123] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 4–11, New York, NY, USA, 1996. ACM.

[124] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 114–121, New York, NY, USA, 2005. ACM.

[125] K. Yu, J. D. Lafferty, S. Zhu, and Y. Gong. Large-scale collaborative prediction using a nonparametric random effects model. In *ICML*, page 149, 2009.

[126] K. Yu, S. Zhu, J. Lafferty, and Y. Gong. Fast nonparametric matrix factorization for large-scale collaborative fil-

tering. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 211–218, New York, NY, USA, 2009. ACM.

[127] S. Zhang, W. Wang, J. Ford, and F. Makedon. Learning from incomplete ratings using non-negative matrix factorization. In *SDM*, 2006.

[128] Y. Zhang and J. Koren. Efficient bayesian hierarchical user modeling for recommendation system. In *Proc. of SIGIR '07*, pages 47–54, New York, NY, USA, 2007. ACM.

[129] Z. Zheng, H. Ma, M. R. Lyu, and I. King. Wsrec: A collaborative filtering based web service recommender system. In *ICWS '09: Proceedings of the 2009 IEEE International Conference on Web Services*, pages 437–444, Washington, DC, USA, 2009. IEEE Computer Society.

[130] D. Zhou, B. Scholkopf, and T. Hofmann. Semi-supervised learning on directed graphs. In *Advances in Neural Information Processing Systems*, volume 17, 2005.

[131] D. Zhou, S. Zhu, K. Yu, X. Song, B. L. Tseng, H. Zha, and C. L. Giles. Learning multiple graphs for document recommendations. In *WWW '08: Proceedings of the 17th international conference on World Wide Web*, pages 141–150, New York, NY, USA, 2008. ACM.

[132] T. C. Zhou, H. Ma, I. King, and M. R. Lyu. Tagrec: Leveraging tagging wisdom for recommendation. In *Proceedings of IEEE International Symposium on Social Intelligence and Networking*, 2009.

[133] S. Zhu, K. Yu, and Y. Gong. Predictive matrix-variate t models. In J. Platt, D. Koller, Y. Singer, and S. Roweis,

editors, *Advances in Neural Information Processing Systems 20*, pages 1721–1728. MIT Press, Cambridge, MA, 2008.

[134] S. Zhu, K. Yu, and Y. Gong. Stochastic relational models for large-scale dyadic data using mcmc. In *NIPS*, pages 1993–2000, 2008.

[135] C.-N. Ziegler and G. Lausen. Propagation models for trust and distrust in social networks. *Information Systems Frontiers*, 7(4-5):337–358, 2005.