Evaluating Recommendation Algorithms Based on U-I Matrix Property Analysis

Chongze Jiao

Evaluating Recommendation Algorithms Based on U-I Matrix Property Analysis

by

Chongze Jiao

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Monday August 28, 2017 at 10:00 AM.

Student number:	4512820	
Project duration:	October 15, 2016 – August 27, 2017	
Thesis committee:	Prof. dr. ir. Alan Hanjalic,	TU Delft
	Dr. Huijuan Wang,	TU Delft, supervisor
	Dr. JuanJuan Cai,	TU Delft

Abstract

Recent years, recommender systems are more and more important for solving information overload problem. They sort through massive data to provide users with personalized content and services. Most researchers focus on designing new algorithms to increase the performance of recommender systems. However, some open challenges stand: Why the performance of an algorithm on different data sets can vary quite a lot? Which property of the data set influences the accuracy of the algorithms? In this thesis, we introduce methodologies to investigate the impact of user-item interactions properties on the accuracy of classical collaborative filtering recommendation algorithms. Firstly, we propose to characterize U-I matrix properties from three domains: network topology, spectrum and information domains. Furthermore, we design several network modification algorithms to systematically modify basic topology properties of a given U-I matrix to create more U-I matrices. Meanwhile, the properties of the spectrum and information domains are also changed as topological features are modified. We finally evaluated several classical collaborative filtering algorithms on a large number of U-I matrices and explore which properties in the three domains can influence or better explain the accuracy of the algorithms. We find that the effect of U-I matrix properties on the accuracy of recommendation algorithms is approximately consistent across various data sets. We identify two properties from the network topology and information domain respectively that could better explain the accuracy of algorithms. Understanding how U-I matrix properties affect the accuracy of algorithms has practical significance. The recommender system designers can estimate and explain the accuracy of their recommender systems and are inspired in the design of policies to orient the user-item interactions such that the accuracy of their recommendation algorithms could be improved.

Keywords: Recommender System, Collaborative Filtering, Weighted Bipartite Graph, U-I Matrix Property

Acknowledgements

I would like to thank all of them who contribute to my master thesis.

First of all, I would like to thank my supervisor dr. Huijuan Wang, who gives me the opportunity to do this project. I really appreciate for your time, support, valuable guidance, critical suggestions and encouragement. You are such a pleasant supervisor. I have learned a lot from you. Thanks, Huijuan.

I would like to express my gratitude for Cunquan Qu and Xiuxiu Zhan. Thank you for all your help during my master thesis. You are so kind to give responses to all the inquiries. I have learned very much from you.

Moreover, I would like to thank my committee members, Prof. dr. ir. Alan Hanjalic and dr. JuanJuan Cai,. Thank you for your time.

Finally, I am so grateful for my wife, Wangyang Yu for the encouragement, support and company.

Chongze Jiao

Delft, Netherlands

August 17, 2017

Contents

1 Introduction

The World Wide Web(WWW) has grown considerably in the last several decades. The explosive growth in the amount of available online information and the number of Internet visitors have created a so-called information overload challenge that users will spend more time on locating their interested information on the Internet. As a consequence, many information retrieval systems, like Google, Baidu, try to solve this information overload problem by adding filters to filter the unimportant information for users. However, all the solutions are inadequate, since personalization and prioritization (systems rank the information base on the user's interest and preference) challenges still exist[21]. The urgent need to solve information overload problem drives the development of recommender systems. A recommender system is a subclass of the information filter system that predicts the rating or preference a user would give an item. For example, Amazon recommender system can recommend a range of products that you are likely to click and buy. Recommender systems deal with the information overload problem by filtering vital information from the huge amount of dynamically generated information according to user profiles, interest, or observed item features[42].

Based on the existing ratings in the a user-item matrix, a representative of a real-world data set for recommender systems, recommender systems guess the users' behavior, e.g., which music to listen, which movie to see and which item to buy[42]. User-item matrix's two dimensions are user, item ID and values are users' interest on items (e.g., using a discrete [1 - 5] rating scale to indicate the level of interest). Recommender systems have a tremendous expansion in the recent decade, regarding the employed techniques and practical applications[6]. More and more companies implement these recommender systems in their websites or systems, for example, Amazon, Yahoo, YouTube, Netflix, Last.fm and Alibaba. In business, recommender systems become important tools to achieve a higher return on investment and quality of service by providing users consumption suggestions on the items which the users are high possibly interested in. E-commerce websites with recommender systems, therefore, show increasing product sales[7]. Recommender systems are beneficial to both online suppliers and users. For users, recommender systems reduce costs in finding and selecting products from millions of online items. Recommender systems are helpful to improve the quality of decision-making process[21]. On the other hand, they enhance profits as effective tools to sell more products. To achieve broader business goal of increasing revenue, researchers often focus on recommendation relevance, recommendation novelty, increasing recommendation diversity in technical field[3].

Recommender systems are typically grouped into collaborative filtering approach and content-based approach. However, different service providers build their systems according to diverse demands. For example, the recommendation ratings on Amazon.com are the [1-5] rating scale that we mentioned above, which is based on the explicitly provided ratings, buying behavior, and browsing behavior. On the other hand, Facebook system predicts the links between recommending friends and increase the advertising revenues other than recommend products, which is also considered as the link prediction problem in the social network analysis field[3]. Based on prediction objectives, the recommendation tasks can be commonly divided into two scenarios,

- **Rating Prediction:** Recommender systems predict a potential rating that user u will rate item i, for example, in MovieLens website, a user can rate any movies from 1 star to 5 stars.
- **Top-N Recommendation:** Recommender systems provide user u a n-item list that he/she is most possibly interested in. Like on the Amazon website, you often see a list that you may like or you may buy.

1.1. Project Motivation and Goals

Recommender systems are one of the most powerful tools in the present digital world. Since the first recommender system, Tapestry, was designed to recommend documents from newsgroups in 1994[41], various approaches have been developed and implemented to improve the performance of recommender systems[21]. The main focus of recommender systems' development is designing new algorithms to improve performance by using more information, such as improving machine learning algorithms and extending the user's and item's features and their interactions[1, 18, 24, 48]. However, a recommender algorithm could perform well in some platforms with its specific user item interactions captured by its user-item data/matrix but badly in other platforms. For example, Item K-Nearest-Neighborhood algorithm (Item KNN) and Biased Matrix Factorization (BMF) algorithm have similar accuracy on Filmtrust, but Item KNN performs much better than BMF on Movielens 100K. An open challenge stands: how do the properties of user-item interactions affect or possibly explain the performance of recommender system algorithms? Some hypotheses have been proposed. Jonathan L. Herlocker hypothesized that the prediction accuracy of recommender systems may be lower on a data set with a more uniform ratings distribution[20]. Yoon-Joo Park and Alexander Tuzhilin hypothesized that the long tail distribution of item degrees (the number of ratings an item received) may degrade algorithm performance since it is hard to make prediction based on many items with only few ratings[34]. To investigate the impact of user-item matrix characteristics on the performance of recommender system algorithms, Gediminas Adomavicius proposed a "window sampling" method to extract a subset from the original dataset. He built a linear regression model to uncover the relationships between several data set characteristics and the accuracy of algorithms[2]. However, few researches have been dedicated to providing a systematic, in-depth exploration and analysis of how the data set structure affects different recommender system algorithms. Understanding this relationship also has practical significance. For example, it would enable system designers to estimate the expected performance of their systems based on simple user-item matrix properties in advance and to guide the user-item interactions to maximize performance of a recommendation algorithm.

In this thesis, we want to explore which properties of the user-item interactions may better explain the accuracy of classical collaborative filtering based recommender system algorithms. Our thesis project's contribution towards that goal includes seeking the answers to the following questions:

- Can we design algorithms to systematically modify user-item matrix topology properties?

- Can we propose reasonable metrics that might affect or explain the accuracy of algorithms?
- How do user-item matrix properties affect recommendation algorithms' accuracy and which property could better explain the accuracy of algorithms?

In order to address these questions, we propose to characterize the user-item interactions/matrix properties from three domains: the network topology, the information domain and the spectrum. In the area of network topologies, a user-item matrix of recommender systems can be seen as a weighted bipartite graph. The users and items are two disjoint sets of nodes. The ratings are weighted links between these two sets. We consider basic topology properties from graph theory, e.g., standard deviation of item ratings, which is the average of standard deviation of ratings received by each item. In the information domain, we proposed an entropy of item ratings based on Shannon entropy to measure the uncertainty of item ratings. Moreover, we use singular values as charactorizers of the spectrum. Real-world user-item matrices are large and differ dramatically in e.g., data sparsity, average rating, the number of users and items. It is difficult to compare their properties. We therefore proposed several network modification strategies to systematically modify some topology properties to create more user-item matrices while keeping the number of users/items, the sparsity and average rating unchanged. When the topology properties are changed, the properties in other domains will be correspondingly changed as well. We finally evaluated several classical collaborative filtering algorithms on the user-item matrices and explore which properties in the three domains we mentioned above can influence and better explain the accuracy of the algorithms.

1.2. Thesis Outline

The thesis is structured as follows:

- **Chapter 2 Background of Recommender Sytems:** In Chapter 2, we give the definition of challenges recommender systems faced, show an overview of existing recommender systems and evaluation metrics. Moreover, we present detailed description of the classic collaborative filtering recommender system algorithms that we considered in the thesis.
- **Chapter 3 Properties of U-I Matrix:** We propose metrics from three domains to characterize user-item matrices : the network topology, the information domain and the spectrum.
- **Chapter 4 Network Modification Strategies:** In this chapter, we introduce the motivation and the methods to modify topological properties of a given user-item matrix.
- **Chapter 5 Experiments Setup:** Chapter 5 summarizes of the various data sets and software tools we used.
- **Chapter 6 Experiments Result:** In this chapter, we show all the experiments under different network modification algorithms and analyze the corresponding results from three domains mentioned in Chapter 3.
- **Chapter 7 Conclusion and Future Work:** In the last chapter, we give the conclusions drawn from the project and offer a suggestion on possible future research.

2

Background of Recommender Systems

In this chapter, we firstly introduce some basic information of recommender systems. Afterwards, we discuss two classes of recommender system algorithms that have been widely used, content-based filtering and collaborative filtering[24]. The collaborative filtering based algorithms will be considered in later chapters when we explore how their performance will be influenced by user-item interaction properties. Furthermore, we present the measures that have been used to evaluation the performance, especially the accuracy of a recommendation algorithm.

2.1. Basic Information of Recommender Systems

In this section, we show some basic information that is widely used in recommender systems, e.g., U-I matrix, user profiles, item attributes.

- The User-Item matrix: The input data of recommender systems can be placed as a user-item (U-I) matrix. An example is shown in Table 2.1. The two dimensions of the user-item matrix are user ID and item ID. The values in this matrix are the ratings that users rated items. There are always massive users and items in the data sets for recommender systems. Since a user in the system is only likely to rate an extremely small percentage of items in a data set, the U-I matrices are sparse.

- User Profiles and Item Attributes: Apart from U-I matrices, many recommender systems consider additional information to increase their accuracy. User profiles information and item attributes are commonly used by recommender systems. User profiles contain age, gender, location and so on. Examples of item attributes are genre, directors and actors for movies.

- Social Network Information: Recent years, more and more researchers add social network analysis into their recommender systems. Social networks provide important information regarding users and their interactions[17], which can be extracted as implicit information for recom-

Table 2.1: An example of user-item matrix with rating 1-5.

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5
Andy	5	1		2	2
Lily	1	5	2	5	5
Lee	2		3	5	4
John	4	3	5	3	

mender systems. The intuition behind this method is that users, who are friends, are likely to have similar interest on an item and users, who are enemies, show different interest on items.

2.2. Content-Based Filtering

As introduced above, there are two commonly used approaches in recommender systems. In this section, we introduce the content-based filtering (CBF) approach first. It recommends items based on a comparison between the description of items and profiles of users[24]. Content-based strategy does not use ratings from others. It can provide explanations on why specific recommendations are produced for users[21]. A CBF recommender system creates a profile to characterize features of a user or an item. For example, a user profile can include the demographic information, like gender and age. An item profile can include actors, directors, genres. The recommendation process is matching up the attributes of a user profile with the features of content objects (items) which are rated by users[30]. As a consequence, if a user profile can reflect users' interest accurately, it is of great advantage in predicting user's preference. A shortcoming of CBF is that it requires rich external information which might be unavailable or difficult to gather[24].

A well-know successful case of CBF approaches is the Music Genome Project, which is used by Pandora.com. The project uses over 450 attributes to characterize songs. Given the songs which a user like or dislike, a list of recommended songs can be constructed by the system[57].

2.3. Collaborative Filtering

Collaborative filtering (CF) is another widely used approach in recommender systems. Many algorithms based on this approach have been developed. It makes recommendation based on ratings or behaviors of other users in a system[10]. The fundamental assumption is that other users' behaviors can be collected to predict the active user's preference. That is, if the users who have the same opinion of some items previously, they will likely have an agreement on other items[10]. This section introduces the different types of CF recommender systems and describes the CF algorithms we used in detail. CF systems are grouped into memory-based CF and model-based CF. In the thesis, we mainly focuses on determining how a U-I matrix property affects the accuracy of several classic or popular collaborative filtering algorithms. The algorithms that are used in our experiments are introduced below, including matrix factorization, bias matrix factorization, SVD++, user k-nearest neighbors, item k-nearest neighbors and factorization machine.

2.3.1. Memory-Based Collaborative Filtering

The Memory-based approaches use user-item ratings data to predict ratings for a new item directly. The most popular memory-based approaches are neighborhood-based algorithms. These algorithms are based on the fact that similar users have similar preference and similar items always receive similar ratings. This situation can be illustrated by the following example,

Example:

As shown in Table 2.2, the recommender system predicts the rating of the movie 2 which Lee has not watched yet. Lily has the similar preference with him, since both of them like Movie 4, 5 and dislike Movie 1. The system can provide recommendation based on Lily's opinion. On the other hand, Lee has different tastes with John, the system could discard John's opinion or consider John's opposite preference.

Table 2.2: An example of memory-based collaborative filtering recommender system task.

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5
Andy	5	1		2	2
Lily	1	5	2	5	5
Lee	2	?	3	5	4
John	4	3	5	3	

The neighborhood-based algorithms can be characterized into user-based and item-based models. User-based models predict the queried user's rating on a item by aggregating previous ratings given by similar users[48]. Item-based models identify items that are similar to the active item and predict the rating as the weighted average ratings of these similar items[27]. Neighborhood-based approaches use similarity matrix to compute the weighted average ratings. Popular functions to compute similarity matirx include cosine similarity, pearson correlation and mean-squared-difference (MSD)[10]. The simplest formulation form of neighborhood-based CF is introduced by Shi. Y [48]

$$\hat{R}_{ij} = \frac{1}{C} \sum_{k \in Z_j} sim(j,k) R_{jk}, \tag{2.1}$$

where Z_j is the set of k-users(items) neighborhood of user(item) j, C is a normalizing constant, $sim(j,k)$ designates the similarity (in terms of a predefined similarity measure) between user(item) j and user(item) k.

The memory-based CF can integrate rich side information of users and items to advance similarity and improve the accuracy of recommendation[48]. However, there is a typical shortcoming in memory-based CF. The computation of similarity between user or item pairs is expensive since the time complexity is very high. It is noteworthy that the similarity metrics here can also be used in model-based approaches.

Some popular methods used for neighborhood-based recommendation are: k-nearest neighbors(KNN), k-means, k-d trees and locality sensitive hashing[9]. In our experiment, we used k-nearest neighbors recommendation algorithms, which compute similarity matrix by calculating the distance between every interaction vector in the query set against the vectors in the reference set. Next, we introduce two KNN methods, the user KNN and item KNN.

User K-Nearest Neighbors

User-based k-nearest neighbors approach was first proposed in GroupLens usenet article recommendation[41]. It is a straightforward algorithmic achieved collaborative filtering method. It predicts the rating that user u rates item i by using the ratings given to i by the k most similar users to u. Suppose ω_{uv} represents the similarity between user u and user $v (u \neq v)$. Let $\mathcal{N}(u)$ denote the set of k-nearest neighbors of user u rated item i. The predicted rating $\hat{r}_{ui}$ is the average rating given to item i by these neighbors:

$$\hat{r}_{ui} = \frac{1}{|\mathcal{N}_i(u)|} \sum_{v \in \mathcal{N}_i(u)} r_{ui}, \tag{2.2}$$

where $|x|$ denotes the number of x. However, different user pairs have different levels of similarity. If the sum of these weights do not equal to 1, the predicted ratings will be outside the rating range. A widely used solution to solve this problem is normalizing these weights[42]. The predicted rating becomes

$$\hat{r}_{ui} = \frac{\sum_{v \in \mathcal{N}_i(u)} \omega_{uv} r_{ui}}{\sum_{v \in \mathcal{N}_i(u)} |\omega_{uv}|}. \tag{2.3}$$

There are many ways to compute the similarity between user u and user v, among which commonly used metrics are pearson correlation coefficient and cosine similarity. Pearson correlation coefficient measures how much two users relate to each other linearly. It computes the statistical correlation between user pair's common ratings to obtain the similarity[10]. It is defined as

$$sim(u,v) = Pearson(u,v) = \frac{\sum_{k \in I_u \bigcap I_v} (r_{uk} - \mu_u)(r_{vk} - \mu_v)}{\sqrt{\sum_{k \in I_u \bigcap I_v} (r_{uk} - \mu_u)^2} \sqrt{\sum_{k \in I_u \bigcap I_v} (r_{vk} - \mu_v)^2}}, \tag{2.4}$$

where μ is the average rating of a user and I_u is the set of items rated by user u. Another popular metric, cosine similarity, is a vector-space method based on linear algebra. In this approach, the unknown ratings are considered as 0. The formula is defined as

$$sim(u,v) = Cosine(u,v) = \frac{\sum_{k \in I_u \bigcap I_v} r_{uk} r_{vk}}{\sqrt{\sum_{k \in I_u} r_{uk}^2} \sqrt{\sum_{k \in I_v} r_{vk}^2}}. \tag{2.5}$$

Item K-Nearest Neighbors

While user k-nearest neighbors approach bases on the similar preference of like-minded users, item k-nearest neighbors method relies on the ratings of similar items. The overall structure is similar as the structure of content-based recommendation method. However, item similarity is determined by user preference rather than extracted information from item data[10]. Similar to user KNN approach, it can use average weighted similarity scores to predict ratings. Denote the most similar items to item i rated by user u by $\mathcal{N}(i)$. The idea behind item KNN can be formalized as

$$\hat{r}_{ui} = \frac{\sum_{j \in \mathcal{N}_u(i)} \omega_{ij} r_{uj}}{\sum_{j \in \mathcal{N}_u(i)} |\omega_{ij}|}. \tag{2.6}$$

The methods to compute the similarity between items are similar as those of user KNN approach.

2.3.2. Model-Based Collaborative Filtering

Model-based CF trains the prediction model with the training data set, which is later used to predict missing ratings and item recommendation. Examples of common model-based CF methods include latent factor models[24], Bayesian classifiers[31]. Latent factor models were very popular several years ago. Its basic assumption is that there exist some low-dimensional feature factors of users and items which can model user-item affinity accurately[44]. Some successful realizations of latent factor models are based on matrix factorization (MF), which are introduced in Section 2.3.4

Although the model-based approaches potentially offer the improvement on both prediction speed and scalability, they still face problems. Model-based systems are inflexible. Since building a model is a time- and resource-consuming process, it is difficult to add new data into the existing systems.

2.3.3. Hybrid Method

Both content-based approaches and collaborative-based approaches have several limitations. Hybrid recommender systems attempt to aggregate different methods to mutually eliminate their drawbacks[49]. There are different ways to combine several recommender methods into a hybrid method[1]:

- Combining separate recommender systems: In this method, we implement collaborative and content-based systems separately. We combine the ratings obtained from each recommender system into one final recommendation or we can use one recommender system with the best performance in a given situation.
- Integrating content-based characteristics into collaborative-based approaches: Most methods in this category are based on traditional collaborative techniques and utilize the content-based profiles. These profiles are commonly used to compute the similarity between users. With this strategy, these hybird recommender systems can overcome sparsity problem which is due to only few user pairs have a considerable number of commonly rated items. Moreover, users can be recommended an item not only highly rated by similar users, but also directly based on his own preference[25].
- Integrating collaborative-based characteristics into content-based approaches: The most widely used approach in this type is to use some dimensionality reduction technique on a group of content-based profiles. For example, we can use latent semantic indexing (LSI) to create collaborative user profiles, which are term vectors[32]. This method provides better performance than pure content-based methods.

2.3.4. Matrix Factorization

Many successful model-based CF recommender systems are based on matrix factorization (MF). Basically, MF characterizes both items and users by latent factor vectors. The idea behind this method is that user's preferences are determined by a few unobserved factors[43]. MF provides recommendations based on the high correspondence between item and user factors. It maps both users and items to a low-dimensionality joint latent factor space[24]. MF reduces the original U-I matrix to two much smaller matrices that approximate the original one when they are multiplied together. Each user u is associated with vector p_u and each item i is associated with vector q_i. The elements of p_u and q_i represent the extent corresponding to user or item's factors. The dot product ${q_i}^T p_u$ measures a user's overall interest of an item based on its characters. That is, the product approximates user u rates item i, which is denoted by $\hat{r}_{ui}$,

$$\hat{r}_{ui} = {q_i}^T p_u. \tag{2.7}$$

The main challenge of MF is how to obtain the accurate user or item latent factor vectors. With these vectors, we can predict the missing ratings accurately[24].

As we have mentioned, MF approaches use dimensionality reduction technologies, which are useful tools to find the hidden information of the data. A popular dimensionality reduction model is singular value decomposition (SVD), which is an effective technique for identifying latent semantic factors in information retrieval[5, 24]. To learn the latent factor vector(p_u, q_i), MF systems minimize the regularized squared error on the available rating set. We use Z_j to denote the set of the (u,i) pairs for where r_{ui} is available in the training set, the formula is shown as

$$\min_{p^*,q^*} \sum_{(u,i)\in Z_j} (r_{ui} - {q_i}^T p_u)^2 + \lambda(\|q_i\| + \|p_u\|)^2, \tag{2.8}$$

where the parameter λ controls the extent of regularization to reduce overfitting.

There are many other popular matrix decomposition models which are used in MF systems, like probabilistic matrix factorization (PMF) and non-negative matrix factorization (NMF).

- Probabilistic Matrix Factorization

  In 2008, Ruslan Salakhutdinov and Andriy Mnih proposed a probabilistic model for regularization. It is a probabilistic linear model with Gaussian observation noise. Moreover, it models the U-I matrix as a product of user and item latent factors[43]. Assume there are M users and N items, R_{ij} is the rating that user i rates item j, and U_i and V_j denote the user latent factor vectors and item latent factor vectors respectively.

  The conditional distribution over the observed ratings is

$$p(R|U,V,\sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{N}[\mathcal{N}(R_{ij}|{U_i}^T V_j, \sigma^2)]^{I_{ij}}, \tag{2.9}$$

  where $\mathcal{N}(x|\mu,\sigma^2)$ is the probability density function of Gaussian distribution with mean μ and variance σ^2. If user i rates item j, I_{ij} equals to 1, otherwise equals to 0. The prior distribution of user and item factor vectors are

$$p(U|{\sigma_U}^2) = \prod_{i=1}^{N}\mathcal{N}(U_i|0, {\sigma_U}^2\boldsymbol{I}), \tag{2.10}$$

$$p(U|{\sigma_V}^2) = \prod_{j=1}^{N}\mathcal{N}(V_j|0, {\sigma_V}^2\boldsymbol{I}). \tag{2.11}$$

  With the fixed observation noise variance σ and the prior variances σ_U and σ_V, maximizing the log posterior distribution over the user and item features is equivalent to minimizing the sum-of-squared-error objective function with quadratic regularization terms. The learning process of the model is minimizing the sum-of-squared-error objective function with quadratic regularization terms. The sum-of-squared-error objective function is

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - {U_i}^T V_j)^2 + \frac{\lambda_U}{2}\|U_i\|^2 + \frac{\lambda_V}{2}\|V_j\|^2, \tag{2.12}$$

  where λ_U is the regularization parameter of user latent vectors and λ_V is the regularization parameter of item latent vectors.

- Non-Negative Matrix Factorization

  Paatero firstly published several papers about positive matrix factorization in 1994. In 1999, new studies about NMF was released by Lee and Seung. After that, NMF became popular[47]. NMF analyzes data matrices with non-negative elements. It uses non-negative constraint to replace conventional orthogonal constraint in matrix factorization models[18, 26]. Suppose the U-I matrix is a $m \times n$non-nagtive matrix ($R^{m\times n}$). NMF method decomposes R into two non-negative factors $W(R^{m\times k})$, $H(R^{k\times n})$ with $k \ll min(m,n)$ and $R \approx WH$. This form is similar as SVD. Since NMF method gives basis and weight vectors with non-negative constraints, comparing to SVD, it offers a better interpretation of the data[47].

As mentioned previously, MF is a popular technique used in many real-world recommender systems. In our experiment, we used three different approaches, MF, bias matrix factorization (BMF), and SVD++. Next, we introduce these three approaches separately.

Matrix Factorization

This algorithm was proposed by Koren in 2009. As introduced above, the basic idea to achieve MF method is finding two low rank matrices whose product best approximates the original U-I matrix[24]. Commonly, MF describes both users and items by factor vectors inferred from item rating patterns. Explicit feedback data (e.g., ratings, votes) can be represented as a sparse U-I matrix containing all the ratings that the users rate the items, the size of which is $M \times N$, where M is the amount of users and N is the items' number. We assume that there are K latent factors applied. Each user is characterized by a k-dimensionality factor vector p_u and each item is featured by a k-dimensionality vector q_i. For a certain item i, the elements of q_i measure the strength of the associations between item i and the latent factors. For a given user u, the elements of p_u weigh the user's preference extent on the corresponding item factors. The elements in both the user and item factor vectors can be positive or negative[24, 36]. The dot product of these two vectors catches the interaction between user u and item i, that is, the user's overall preference on the item's factors. This dot product is the predicted rating $\hat{r}_{ui}$, which is described as Equation 2.7.

With accurate mapping information, a recommender system can provide the rating that user u give item i by the Equation 2.7. The mapping of these factor vectors can be realized in several dimensionality reduction technologies we mentioned above. In our experiments, the MF algorithm that we evaluated uses a quite commonly used method, SVD. It needs full user-item rating matrix. However, it is often impossible beacuse of the large amount of missing values caused by the sparsity of the data set. To solve the problem, a technique, called imputation, is often applied to replace the missing values with estimated values and to make the matrix dense. However, imputation process can be very computing expensive when the data set is extremely large. Moreover, imputation can cause overfitting or be inaccurate[24]. Consequently, more works only model the observed ratings directly and avoid overfitting by regularizing the model. To learn the latent factor vector (p_u, q_i), MF algorithm minimizes the regularized squared error on the available rating set, which is described as Equation 2.8

Bias Matrix Factorization

BMF algorithm in MyMediaLite[13] based on PMF algorithm, we mentioned above, with explicit user and item bias. PMF can be viewed as a probabilistic extension of the SVD model[43]. The objective function in Equation 2.12 can be achieved by performing gradient descent in U and V. In this algorithm, in order to avoid making predicted ratings outside of the range of valid rating values, Ruslan Salakhutdinov and Andriy Mnih used logistic function $g(x) = \frac{1}{1+exp(-x)}$ to obtain the dot product between use- and item-factor vectors[43].

Combining user and item bias on PMF model, the predicted ratings become

$$\hat{r}_{ij} = \mu + b_i + b_j + {q_i}^T p_u, \tag{2.13}$$

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - (\mu + b_i + b_j + {U_i}^T V_j))^2 + \frac{\lambda_U}{2}\|U_i\|^2 + \frac{\lambda_V}{2}\|V_j\|^2 + + \frac{\lambda_{Ub}}{2}\|b_i\|^2 + \frac{\lambda_{Vb}}{2}\|b_j\|^2, \tag{2.14}$$

where μ is the global bias, b_i and b_j are user bias and item bias.

SVD++

As introduced, SVD is a widely used technique in information retrieval. Since SVD can not be used directly on sparse matrices, the popular method to solve the problem is computing SVD approximately or filling the missing values by 0[36]. Koren introduced a new algorithm called SVD++ that integrates the neighborhood model with factor model, that is, adding implicit feedback to latent factor model. The biased latent factor equation is shown in Equation.2.13.

SVD++ algorithm adapts Arkadiusz Paterek's idea on NSVD model, which avoids fitting each user u_i separately but models user u_i as a function based on the items they rated[35]. The representation of a user u in the item-based CF therefore can be written as

$$U = p_u + \frac{1}{\sqrt{|R(u)|}} \sum_{j \in R(u)} y_j, \tag{2.15}$$

where $R(u)$ is the set of items rated by user u, $|R(u)|$ is the number of user and y_j indicates a new set of item factors that characterize implicit ratings. p_u is learned from the explicit rating and $\frac{1}{\sqrt{|R(u)|}} \sum_{j \in R(u)} y_j$ is the implicit feedback (e.g., clicks, purchase actions). The sum of y_j is normalized by $\sqrt{|R(u)|}$ to stabilize the variance across the range of observed values of $|R(u)|$[42].

SVD++ is an extension of SVD algorithm that takes implicit ratings into account. Combining these two equations, we can obtain the exact model of predicted ratings,

$$\hat{r}_{ij} = \mu + b_i + b_j + q_i{}^T (p_u + \frac{1}{\sqrt{|R(u)|}} \sum_{j \in R(u)} y_j). \tag{2.16}$$

2.3.5. Factorization Machine

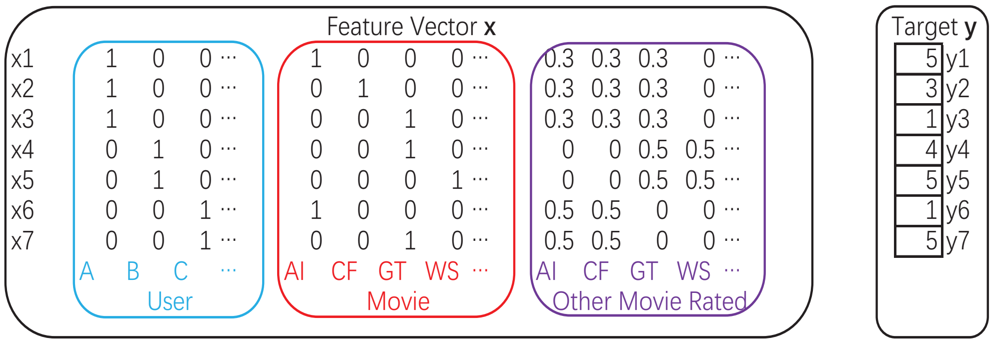
Factorization Machine (FM) is a popular model in recent years, which was first proposed by Rendle in 2010. It combines the flexibility of feature engineering and the strength of factorization models in building interaction between latent factors of large domain[40]. FMs work well in huge sparse settings, such as recommender systems. The high performance because of the internal mode applies factorized interactions between variable information like other factorization models do. Moreover, it can mimic most factorization models by feature engineering, which is a process using data domain knowledge to create features that makes machine learning algorithms work[42]. Similar to other machine learning methods, like linear regression, the input data of FMs consist of real-valued features. Its prediction task also estimates a function from real-value feature vectors to a target domain[11, 39, 42]. The basic task of FM is estimating a function between the target rating value y and a real valued feature vectors x. To achieve the FM task, the input data has been arranged by feature vector in a new format, which is shown in the Figure 2.1.

The model of FM with degree d = 2, which captures all single and pairwise interactions between features, is computed as[39]

$$\hat{y}(\mathbf{x}) := \omega_0 + \sum_{i=1}^{n} \omega_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \omega_{ij} x_i x_j, \tag{2.17}$$

where the model parameters are listed below,

- n is the number of features
- x is feature vector
- $\hat{y}(x)$ is the predicted rating of feature vector x
- ω_0 is the global bias
- ω_i is the strength of i-th variable



Feature Vector **x**															Target **y**	
	User				Movie					Other Movie Rated						
	A	B	C	⋯	AI	CF	GT	WS	⋯	AI	CF	GT	WS	⋯		
x1	1	0	0	⋯	1	0	0	0	⋯	0.3	0.3	0.3	0	⋯	5	y1
x2	1	0	0	⋯	0	1	0	0	⋯	0.3	0.3	0.3	0	⋯	3	y2
x3	1	0	0	⋯	0	0	1	0	⋯	0.3	0.3	0.3	0	⋯	1	y3
x4	0	1	0	⋯	0	0	1	0	⋯	0	0	0.5	0.5	⋯	4	y4
x5	0	1	0	⋯	0	0	0	1	⋯	0	0	0.5	0.5	⋯	5	y5
x6	0	0	1	⋯	1	0	0	0	⋯	0.5	0.5	0	0	⋯	1	y6
x7	0	0	1	⋯	0	0	1	0	⋯	0.5	0.5	0	0	⋯	5	y7

Figure 2.1: Example for matrix used by factorization machine. Every row represents a feature vector x_i with its corresponding target y_i. The first 4 columns (blue) represent indicator variables for users. The next 5 (red) indicator variables for items. The last 5 columns (purple) are additional implicit indicators (i.e. other items the user has rated). The right column is the target (the predicted ratings).

- v_i describes the vector of feature i
- $\omega_{ij} :=< v_i, v_j >= \sum_{f=1}^{k} v_{i,k} \cdot v_{j,k}$ models the factorization interaction of the variable pair $i-j$

These model parameters can be optimized by learning algorithms. Three popular learning algorithms have been proposed for FMs, stochastic gradient descent (SGD), alternating least-squares (ALS), and Markov Chain Monte Carlo (MCMC)[11, 39]. In this thesis, we only used SGD method. SGD is a simple-implemented algorithm. It works well with different loss functions, and its computational and storage complexity is low. The learning idea of SGD is iterating over cases (rows) of the training data and performing small steps to gradually reduce loss [40]. The detailed optimization process of SGD is shown in Rendle's essay[40].

2.3.6. Limitation of Collaborative Filtering

E-commerce recommender systems are always utilized in challenging situations, since they are asked to offer high-speed and accurate recommendation. Although there are many successful cases based on CF systems, challenges of CF algorithms still exist. According to X. Su(2009)[51], the significant CF challenges are listed below,

- Data Sparsity: In reality, many e-commerce recommender systems use quite huge product sets. The U-I matrix for recommendation is extremely sparse. It is hard to locate the target items. Lacking of information challenges the accuracy of recommendation[21, 45].
- Cold-start problem: If recommender systems do not have sufficient information about a certain user or item, it is difficult to provide accurate recommendation for the user or item[21]. The new or inactive users have not rated any item, which leads to inability to know the taste of the users.
- Scalability: With massive users and items in a data set, traditional CF algorithms suffer serious scalability problems, that is, the computational resources go beyond acceptable levels.

Since most online recommender systems make recommendation for all users immediately, it demands a high-scalability CF system. The most practical methods to deal with scalability problem are dimensionality reduction techniques, such as SVD, which can provide efficient recommendation[10, 21, 45].

- Synonymy: Synonymy is the scenario that the same or very similar items have different names or entries. Most recommender systems find it hard to discover the latent association and treat these items as different ones. For example, “children movie” and “children film” are the same meaning, but CF systems usually assume there is no match between them when compute their similarity[10, 21]. Synonyms decrease the recommendation performance of CF systems.

2.4. Evaluation Metrics

In order to compare the performances of different algorithms, several evaluation metrics have been proposed from different aspects, e.g., accuracy, time complexity, novelty and diversity. In this thesis, we focused on the accuracy of algorithms. In this section, we list some commonly used metrics for evaluating the recommendation accuracy in two scenarios: measuring ratings prediction accuracy, measuring top-N ranking prediction accuracy[5, 10, 46].

2.4.1. Measuring Ratings Prediction Accuracy

In such cases, we measure the accuracy of the predicted ratings, that is, comparing the predicted values with the actual observed values. Root-mean-square Error (RMSE) and Mean absolute error (MAE) are commonly used metrics to measure ratings prediction accuracy. The definition of RMSE and MAE are shown as follows.

- Root-Mean-Square Error: RMSE is the square root of the average of squared differences between predictions and real ratings,

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - \hat{r}_i)^2}. \tag{2.18}$$

- Mean Absolute Error: MAE is the average of the absolute differences between prediction and real ratings where all individual differences have the same weight,

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|r_i - \hat{r}_i|, \tag{2.19}$$

where n is the number of ratings in the train set, r_i is the real rating, $\hat{r}_i$ is the predicted rating provided by the system. RMSE squares the error before summing and gives higher weights to large errors[5, 10, 46]. As a result, RMSE is more useful when large errors are particularly undesirable. RMSE and MAE only depend on the error magnitude. Both of them can range from 0 to ∞ and are indifferent to the direction of errors[53]. The larger their values are, the lower accuracy is. In this thesis, we only used RMSE to measure the prediction accuracy performance of different offline recommender system algorithms, because it has the benefit of penalizing large errors more.

2.4.2. Measuring Top-N Ranking Prediction Accuracy

In such scenarios, the recommender systems predict a list of N items that a user will like or not. We can divide the tasks into measuring usage prediction and ranking measures. The former one is whether the user will use the new item in the future and the later one is providing a list ordered by the user's preference[5, 46].

For measuring usage prediction, precision and recall are the commonly used metrics. The definition of them are shown as follows. Let $|x|$ denote the number of x.

- Precision: Precision is defined as the ratio of relevant items to recommended items, that is, the percentage of relevant items in the result

$$\text{Precision} = \frac{|Interesting\ items \cap Recommended\ items|}{|Recommended\ items|}. \tag{2.20}$$

- Recall: Recall is the ratio of relevant items to interesting items, which denotes the percentage of retrieved relevant items.

$$\text{Recall} = \frac{|Interesting\ items \cap Recommended\ items|}{|Interesting\ items|}. \tag{2.21}$$

If we retrieve more documents, we can improve recall but reduce precision, vice versa. That is to say, there is a tradeoff between precision and recall. Consequently, we need to take both of them into account when evaluating different algorithms. A well-known measure that considers both is called mean average precision(MAP). To calculate MAP, we firstly average precision AP for top-N recommendation. It is defined as

$$AP@N = \frac{1}{min(N,m)} \sum_{i=1}^{N} P(i), \tag{2.22}$$

where m is the relevant item of a particular user, $P(i)$ is the precision of item i in the recommendation list and $min(x, y)$ is the minimum value of x and y. MAP calculates AP for each query, MAP for M queries is

$$MAP@N = \frac{1}{M} \sum_{k=1}^{N} P(k)AP@X_k, \tag{2.23}$$

where X is the number of top recommendations being evaluated.

To evaluate the accuracy for ranking measures, a popular top-N ranking quality called normalized cumulative discounted gain (NDCG) is commonly used, which is a measure of algorithm effectiveness from information retrieval. Discounted cumulative gain (DCG) is a weighted sum of relevance for ranked items. It uses the graded relevance to measure the gain, which represents how well an item meets the need. It is a decreasing function of item rank, which is also called discount[55]. The DCG is defined as

$$DCG_N = \sum_{i=1}^{N} \frac{2^{rel_i} - 1}{\log_2(i+1)}, \tag{2.24}$$

where N is the number of items on the recommendation list, rel_i ranges from 0 to 1, which denotes graded relevance of the item at position i on the top-N recommendation list. NDCG is a normalization of the DCG measure. The value of NDCG ranges from 0 to 1, where 1 means ideal ranking and 0 means the worst ranking[23, 46]. NDCG is the normalized form of DCG given by

$$NDCG_N = \frac{DCG_N}{IDCG_N}, \tag{2.25}$$

where $IDCG_N$ is the ideal DCG, which is the recommendation list with the highest relevancy at the top and is sorted in decreasing order of relevance.

2.5. Summary

Generally speaking, recommender systems can be categorized into content-based filtering, that is based on the correlation between user profiles or item attributes, and collaborative-based filtering, that is based on other users' behavior in the system. Collaborative-based filtering algorithms are widely divided into memory-based and model-based methods. The former ones use the similarity between users or items to provide a recommendation, which are broadly divided into user and item based. The model-based approaches use available data to train a model and predict the ratings based on the model. A commonly used model-based approach is MF. The idea behind such method is that we can use few latent feature factors to determine user's preferences. Some extend implementations are added to the basic matrix factorization approach to improve the performance, like bias, additional information. There are many collaborative filtering challenges that decline the algorithm performance: data sparsity, cold-start problem, scalability and synonymy. Therefore, many hybrid algorithms are developed to eliminate their drawbacks. A new popular model partially based on matrix factorization, called factorization machine, arose several years ago. It combines the strength of matrix factorization method and feature engineering. FM uses any real-valued feather vector as input to predict the ratings. The frequently used learning algorithms for recommender systems are stochastic gradient descent (SGD) and alternating least-squares (ALS).

In our experiments, we used several different Collaborative filtering algorithms, including matrix factorization, bias matrix factorization, SVD++, user k-nearest-neighbor, item k-nearest-neighbor and factorization machine.

To compare the performances of different algorithms, researchers proposed many evaluation metrics. In this thesis, we mainly focus on the accuracy. For rating prediction, RMSE and MAE are popular metrics to measure the accuracy of algorithms, for top-N ranking prediction, MAP and NDCG are commonly used.

3
Metrics of U-I Matrix

In the previous chapter, we have already introduced some basic knowledge of recommender systems. With the background information, we want to identity which property of U-I matrix will affect or explain the accuracy of algorithms. In this chapter, we propose several metrics to characterize U-I matrix properties that may influence algorithms' accuracy. We talk about the metrics from three domains, the network topology, the information domain and the spectrum.

3.1. Network Topology

In network science, there are considerable network topological properties that describe a network, such as degree distribution, assortativity coefficient. As we mentioned in Chapter 1, a U-I matrix of recommender systems can be represented as a weighted bipartite graph. The performance of recommendation algorithms on a U-I matrix may be influenced by the topology properties. In this section, we propose three basic network topology metrics, including user-item degree-degree correlation, STD of item ratings and STD of item degrees.

3.1.1. User-item Degree-Degree Distribution

Assortativity, which is also called degree correlation, is an important topological property in real-world networks. It represents the extent that nodes in a network associate with similar nodes, namely, analyzing whether high-degree nodes in a network tends to connect to high-degree nodes, or to low-degree nodes, or to any node[33, 37]. We define the number of ratings provided by a user as the user's degree and the number of ratings received by an item as the item's degree. The input data sets of recommender systems are often stored as a matrix with 3 columns. The data in the first column are user IDs and the data in the second column are item IDs, which are the two columns in the left of Figure 3.1. Values in the third column are the ratings of items provided by users, e.g. a 1-5 star rating or a like/dislike statement. The corresponding degrees of user ID and item ID columns are the two columns in the right of Figure 3.1. The user-item degree-degree correlation ρ is the Pearson correlation coefficient between the user degree and item degree columns in a data set, which is shown as

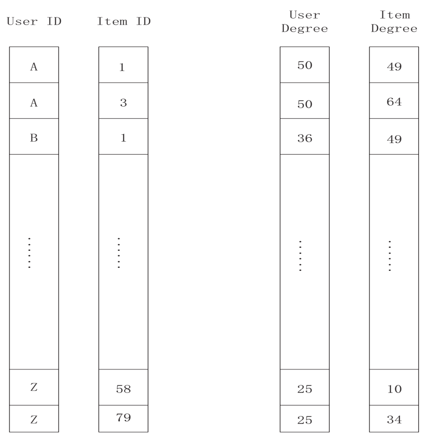


Figure 3.1: Calculating User-Item Degree-Degree Correlation. The left two columns are the user and item ID columns in a data set. The right two columns are their corresponding degrees. We define the user-item degree-degree correlation is the Pearson correlation coefficient between user and item degree columns.

$$\rho = \frac{E[D_u D_i] - E[D_u]E[D_i]}{\sqrt{Var[D_u]Var[D_i]}}, \tag{3.1}$$

where D_u and D_i are the degrees of users and items, $E[x]$ and $Var[x]$ are the mean and variance of x. The user-item degree-degree correlation represents preference for an active user to attach to popular items. In general, ρ ranges from -1 to 1. $\rho > 0$ means active users tend to rate popular items and $\rho < 0$ means active users tend to rate unpopular items. In real-world data sets for recommender systems we used, ρ is always less than zero.

3.1.2. Standard Deviation of Item Ratings

Variance or standard deviation of ratings (hereinafter to be referred as ratings STD) is also a potential important property for recommendation. Variance is the most informative rating-value-related measure to show item's popularity uncertainty. For example, a highly controversial item tends to have higher rating variance or standard deviation. Jonathan L. Herlocker holds the opinion that data with high rating variance should not be considered as bad data, but the high variance can cause the recommendation errors[2, 19]. For an item which receives the same ratings from the users, we can even show an accurate prediction on the item. Some researches have shown that high rating variance will affect users' choice[52]. For a product with low average rating, high rating variance will attract more potential customers. For high average rating products, high rating variance will drive away marginal customers[52]. We define the ratings STD metric as the average of STD of ratings for each item, which is shown as

$$\text{ratings STD} = E[STD(R_i)], \tag{3.2}$$

where R_i is the rating set of item i.

3.1.3. Standard Deviation of Item Degrees

Degree distribution is an important measure to distinguish complex network topologies on determining the number of nodes in the network with a specific degree[22]. Data sets of recommender systems are sparse and the degree of users and items perform long-tail distribution, where few popular items are rated many times and most items are rated only few times[34]. Modifying the user or item degree distribution can have a potential influence on performance. Yoon-Joo Park and Alexander Tuzhilin hypothesized the long-tail distribution can possibly get the accuracy of recommender systems worse since most users or items have little information for recommendation[34]. To measure degree distribution, Snijders firstly proposed a measure of heterogeneity for complex network with degree variance in 1981[50]. It was modified by Bell in 1992[4],

$$VAR = \frac{1}{N}\sum_{i}^{N}(k_i - < k >)^2, \tag{3.3}$$

where VAR is the degree variance of the network, k_i is the degree of i-th node, $< k >$ is the average degree in the network and N is the number of nodes in the network. In this thesis, we define STD of item degrees (hereinafter to be referred as degrees STD) metric as the STD of degrees for each item to measure the degree distribution of a network, which is

$$degrees\ STD = \sqrt{VAR}. \tag{3.4}$$

3.2. Information Theory

In information theory, Shannon entropy is used to measure the unpredictability of the information content or the chaos of a system. The entropy of a finite set $A = \{a_i, ...a_n\}$ is defined as

$$H(A) = -\sum_{i=1}^{n} p_i log_2 p_i, \tag{3.5}$$

where p_i is the probability of each element in set A. In this thesis, we proposed an entropy to measure the uncertainty of item ratings in information domain based on Shannon entropy. Suppose the ratings of item i are in a set whose values range from 1 to 5. We first compute the Shannon entropy for the rating set of each item. Next, we average all the entropy to obtain the entropy for the whole data set. Similar to Shannon entropy, the larger the value is, the more uncertain the ratings are. We predict that the high uncertainty will make the prediction accuracy worse.

3.3. Spectrum

Spectral theory plays an important role in network science. For instance, the largest eigenvalue of an adjacency matrix, which is also called the spectral radius of the network, powerfully characterizes dynamic processes on networks, such as spreading on the network[28]. The ratio of second smallest eigenvalue to largest eigenvalue of the Laplacian matrix can characterize synchronizability, which is an emerging phenomenon in complex networks[8]. Moreover, the topological structure of networks can be fully described by the associated adjacency matrices and their spectral density[12]. Since spectrum can describe network topological structure, we hypothesize that spectrum possibly explains the performance of recommender system algorithms. We use singular values as charactorizers of thespectrum because U-I matrices do not have eigenvalues, which are commonly used tools to analyze spectrum. There are too many singular value metrics. In our thesis, we identify whether the biggest three singular values affect the accuracy of algorithms or not.

3.4. Summary

In this chapter, we propose several metrics from three different domains: the network topology, the information domain and the spectrum. In the network topology, we present user-item degree-degree correlation, ratings STD and degrees STD. In information domain, we define an entropy to measure the uncertainty of item ratings. In spectrum, we analyze the biggest three singular values. With these metrics, we aim to identify which user-item matrix property affects or explain the accuracy of algorithms.

4

Network Modification Strategies

In this thesis, we mainly focus on investigating the impact of U-I matrix properties on the accuracy of recommendation algorithms. We should analyze the accuracy of algorithms on some comparable U-I matrices. For real-world U-I matrices, there are too many U-I interaction properties. It is extremely hard to compare these U-I matrices because of these dramatically different properties. We therefore want to systematically modify a few topology properties in real-world U-I matrices to create more U-I matrices for our experiments. As we mentioned in Chapter 1, a U-I matrix can be represented as a weighted bipartite graph, where users and items are two disjoint sets and rating values are the weight of the links between users and items. We therefore can use network rewiring methods to modify data set properties. In this chapter, we propose three rewiring algorithms to modify the data set properties that may influence the performance, which includes user-item degree-degree correlation, ratings STD and degrees STD.

4.1. Preliminary

As we introduced in Chapter 1, to modify U-I matrix to create more data sets, Gediminas Adomavicius proposed a "window sampling" method to extract subsets from an original data set. His idea is rearranging rows and columns of the U-I matrix according to the degree distribution. Move a fixed size rectangle-shaped window around the U-I matrix, and extract the ratings that fit within the boundaries of the window as a subset[2]. However, many properties are changed at the same time in this method. It is hard to evaluate how a specific property affects the algorithm performance. We therefore proposed some rewiring algorithms to systematically modify the original U-I matrices.

4.2. Degree-Preserving Rewiring

Degree-preserving rewiring is a rewiring mechanism that is often used in network science, which modifies network's inherent structural properties rather than a node's degree[54, 56]. It was firstly proposed in 1996, A. Ramachandra Rao used Monte Carlo algorithm to rearrange the network at random. After sufficient rewiring, although the degree distribution is still same as the initial network, the network topological structure is completely different from the original one[38]. There are many algorithms for degree-preserving rewiring. The simplest way to achieve degree-preserving rewiring is shown in the Figure 4.1. It can be described as follows. For any complex network $G(N,L)$ with N nodes and L links, randomly select two pairs tied nodes (A,C) and (B,D). After that, switch these two links to generate new node pairs (A,D) and (B,C).

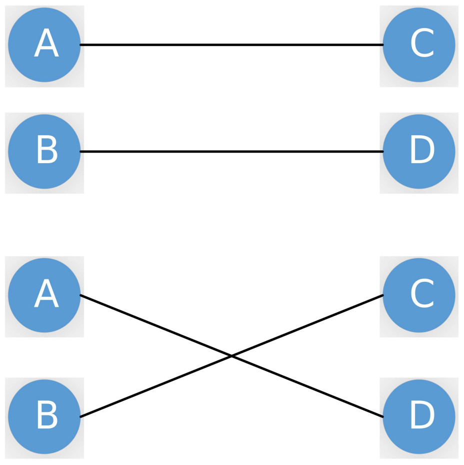


Figure 4.1: Degree-Preserving Rewiring. In complex network $G(N,L)$ with N nodes and L links, randomly select two pairs tied nodes (A,C) and (B,D) and switch these two links to generate new node pairs (A,D) and (B,C)

Degree-preserving rewiring is a useful tool to modify the network and keep the node properties constrained[54]. While adding some qualifications before switching the links, we can rearrange the network by adjusting a specific property, e.g., keeping a high-degree node connect to other high-degree nodes, we can increase the overall degree correlation of a whole network. Base on this algorithm, we proposed two algorithms that modify user-item degree-degree correlation and ratings STD respectively.

4.3. Modifying User-Item Degree-Degree Correlation

In this section, we present our first rewiring algorithm that modifies user-item degree-degree correlation, which is based on the degree-preserving rewiring algorithm we mentioned in the previous section.

Before introducing our algorithm, we firstly introduce a lemma and its detailed proof in P. Van Mieghem's paper, which showed the basic idea of modifying degree correlation in unweighted network[54]:

Lemma 1 *"Given a graph in which two links are degree-preservingly rewired. We order the degree of the four involved nodes as $d_{(1)} \geq d_{(2)} \geq d_{(3)} \geq d_{(4)}$. The two links are associated with the 4 nodes $n_{d_{(1)}}, n_{d_{(2)}}, n_{d_{(3)}}, n_{d_{(4)}}$, only in one of the following three ways: a)$n_{d_{(1)}} \sim n_{d_{(2)}}, n_{d_{(3)}} \sim n_{d_{(4)}}$, b)$n_{d_{(1)}} \sim n_{d_{(3)}}, n_{d_{(2)}} \sim n_{d_{(4)}}$, and c)$n_{d_{(1)}} \sim n_{d_{(4)}}, n_{d_{(2)}} \sim n_{d_{(3)}}$. The corresponding linear degree correlation introduced by these three possibilities obeys $\rho_a \geq \rho_b \geq \rho_c$"*

Based on Lemma 1, P. Van Mieghem designed some rewiring rules to increase or deduce the degree correlation in the network. His degree-preserving assortative random rewiring algorithms is defined as below. Randomly select two links with four different nodes. Rewire the links as in (a). If any of the new links exists before rewiring, discard this step and reselect a new pair of links. For degree-preserving disassortative random rewiring, he obeys the rules as in (c)[54]. This algorithm is designed for normal unweighted complex networks. In the thesis, we add more qualifications to design our algorithm that modify weighted bipartite graph's degree-degree correlation.

In bipartite graph, a node never connects with other nodes in the same disjoint set. Given a bipartite graph, $u1$, $u2$ are two nodes in set u and $i1$, $i2$ are two nodes in set i. The degree of these four involved nodes can be one of the following ways: 1)$d_{(u1)} \geq d_{(u2)} \& d_{(i1)} \geq d_{(i2)}$, 2)$d_{(u1)} \geq d_{(u2)} \& d_{(i1)} < d_{(i2)}$, 3)$d_{(u1)} < d_{(u2)} \& d_{(i1)} \geq d_{(i2)}$. To increase degree correlation, we switched the links under (2),(3) and we rewired the links under (1) to decrease degree correlation. Moreover, to keep the average ratings and STD of rating for users and items constant, we only switch the links $(A,1)$ $(B,2)$ with the same weight, which is shown in Figure 4.2. With sufficient switching, the network's topological structure is distinct from the original network. We describe our algorithm $MDDC$ as Algorithm 1:

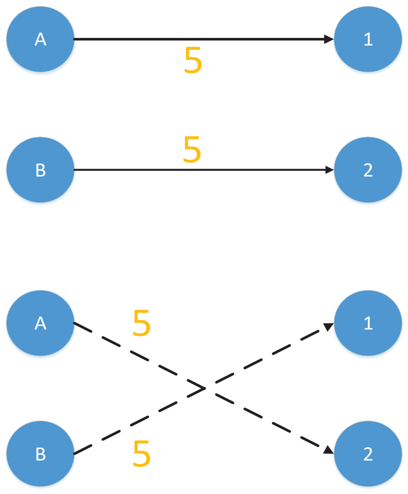


Figure 4.2: Modifying User-Item Degree-Degree Correlation. Random select two links $(A,1)$ $(B,2)$, switch these two links when their weight are the same. By this way, the average ratings, STD of ratings for users and items remain constant

Similar to degree-preserving assortative random rewiring, degree-preserving disassortative random rewiring for weighted bipartite graph just modifies the last IF conditional statements to ($Degree\ of\ User[r1] > Degree\ of\ User[r2]$ and $Degree\ of\ Item[r1] < Degree\ of\ Item[r2]$) or ($Degree\ of\ User[r1] < Degree\ of\ User[r2]$ and $Degree\ of\ Item[r1] > Degree\ of\ Item[r2]$). We therefore can increase or decrease degree correlation without changing other variables.

4.4. Modifying Standard Deviation of Ratings

This section outlines the second network modification algorithm we proposed, which modifies ratings STD. In this process, the ratings STD and user-item degree-degree correlation are changed at the same time. In this case, we explore whether ratings STD affects the accuracy of recommender system algorithms. We apply degree-preserving rewiring algorithm to avoid changing the average rating and item degree distribution, the process is shown in Figure 4.3. We switch the links $(A,1)$ $(B,2)$ with different weight that can increase or decrease both the ratings STD of node 1 and node 2. Our algorithm $MRSTD$ is Algorithm 2:

Algorithm 1 MDDC: Modify Degree-Degree Correlation

```
1: function MDDC(Array (User ID, Item ID, Ratings), step)
2:     for each $i \in$ step do
3:         length ← # Array rows
4:         r1 ← random(length)
5:         r2 ← random(length)
6:         if Array[$r1,2$] == Array[$r2,2$] then
7:             if Array[$r1,0$] $\neq$Array [$r2,0$] and Array[$r1,1$] $\neq$ Array[$r2,1$] then
8:                 if Array[$r1,0$] not rates Array[$r2,1$] and Array[$r2,0$] not rates Array[$r1,1$] then
9:                     if (Degree of Array[$r1,0$] > Degree of Array[$r2,0$] and Degree of Array[$r1,1$] > Degree of Array[$r2,1$]) or Degree of Array[$r1,0$] < Degree of Array[$r2,0$] and Degree of Array[$r1,1$] < Degree of Array[$r2,1$]) then
10:                        Switch Links $r1, r2$
11:                    end if
12:                end if
13:            end if
14:        end if
15:    end for
16: end function
```

Algorithm 2 MRSTD: Modify Standard Deviation of Item Ratings

```
1: function MRSTD(Array, step)
2:     for each $i \in$ step do
3:         length ← # Array rows
4:         r1 ← random(length)
5:         r2 ← random(length)
6:         if Array[$r1,2$] $\neq$ Array[$r2,2$] then
7:             if Array[$r1,0$] $\neq$ Array[$r2,0$] and Array[$r1,1$] $\neq$ Array[$r2,1$] then
8:                 Std Ori1 ← Std(Original Ratings of Array[$r1,1$])
9:                 Std Ori2 ← Std(Original Ratings of Array[$r2,1$])
10:                Std New1 ← Std(Ratings of Array[$r1,1$] After Rewiring)
11:                Std New2 ← Std(Ratings of Array[$r2,1$] After Rewiring)
12:                if Std New1 > Std Ori1 and Std New2 > Std Ori2 then
13:                    Switch Links $r1, r2$
14:                end if
15:            end if
16:        end if
17:    end for
18: end function
```

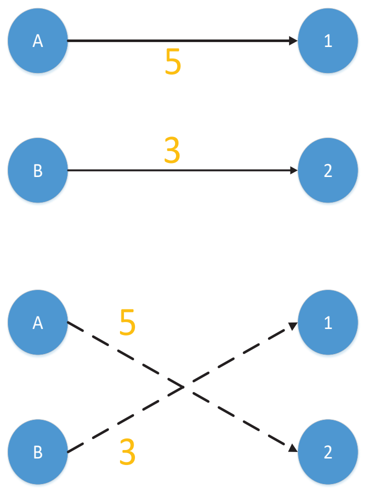


Figure 4.3: Modifying STD of Item Ratings. Select two links $(A,1)$ $(B,2)$ randomly. When their weights are different and the potential rewiring can increase/decrease ratings STD of both item 1 and item 2, switch the two links.

4.5. Modifying Item Degree Distribution

In this section, we introduce the third rewiring algorithm, which modifies the degree distribution of items in a data set. In our algorithm MDD, we randomly cut a link $(A,1)$ and connect user A to a new neighbor 2 to modify the item's degree distribution, which is shown in Figure 4.4. The steps are shown in Algorithm 3.

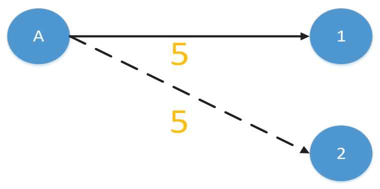


Figure 4.4: Modifying Item Degree Distribution. We randomly select a link $(A,1)$ and an item 2. If user A did not rate item 2 in the data set, we cut this link and connect user A to item 2.

During this process, the user-item degree-degree correlation, the ratings STD and the degrees STD are changed at the same time.

4.6. Summary

In this chapter, we proposed three network modification strategies to gradually modify real-world data sets systematically to create more U-I matrices while keeping the number of users/items, the sparsity and average rating unchanged. We modified user-item degree-degree correlation, ratings STD and item degree distribution. To modify user-item degree-degree correlation and ratings STD, we applied two algorithms that are based on degree-preserving rewiring algorithm. We switch the links between user-item pairs with the same rating in the former algorithm and switch the links between user-item pairs with the different ratings in the latter one. To modify item degree distribution, we cut

Algorithm 3 MDD: Modify Item Degree Distribution

```
1:  function MDD(Array, step,#item)
2:      for each $i \in$ step do
3:          length ← # Array rows
4:          r1 ← random(length)
5:          r2 ← random(# item)
6:          if Array[$r1,0$] not rates Item $r2$ then
7:              Delete link $r1$ and Connect Array[$r1,0$], Item $r2$
8:          end if
9:      end for
10: end function
```

a link between a user and an item randomly and randomly connect a new neighbor to the user. Table 4.1 shows how the topology properties are modified in these algorithms.

Table 4.1: Topology Properties Modified in Network Modification Algorithms

	U-I Degree-Degree Correlation	Ratings STD	Degrees STD
MDDC	Increased & Decreased	Unchanged	Unchanged
MRSTD	Increased	Increased & Decreased	Unchanged
MDD	Increased	Increased	Decreased

5

Experiment Setup

In this chapter we describe the setup used in our experiments, including original data sets and tools. We modified the original data sets, Movielens 100K, Movielens 1M, Filmtrust and Yahoo Music, by the network modification algorithms we introduced in Chapter 4 to create more user-item matrices. We used some recommender system three third party tools to obtain the accuracy of algorithm on these U-I matrices, including MyMediaLite, libFM and WrapRec. Additionally, custom Python code was written for network modification and properties acquisition tasks.

5.1. Data sets

In this section, we introduce the real-world data sets we used in our experiments, which include Movielens 100K, Filmtrust, Yahoo Music and Movielens 1M.

5.1.1. MovieLens

MovieLens[1] are several data sets from a non-commercial movie recommendation site published by GroupLens, which is a research lab in the Department of Computer Science and Engineering of the University of Minnesota[16]. We used MovieLens 100K and MovieLens 1M data sets in this project. The former one contains 100,000 ratings from 943 users on 1682 movies and the latter one includes 1,000,209 ratings of 3952 movies made by 6040 users. Each user rates at least 20 movies in the data sets.

Besides the ratings provided, extra information is also contained in the data set:

- Timestamp: The current time when a user rated an item.

[1]http://www.movielens.org

- User information: Demographic information provided voluntarily by the users, includes gender, age, occupation and zip-code.
- Movie information: Each movie's genre which is selected from one or more out of 18 pipe-separated genres.

5.1.2. Filmtrust

FilmTrust is a small dataset crawled from the entire FilmTrust website in June 2011, it contains 35497 ratings from 1508 users to 2071 items[15]. Besides the U-I rating matrix, this data set also have a trust matrix with explicit trust value, which is a social network information represented the trust and distrust relationship between users. Each user has rated at least 10 songs in this data set.

5.1.3. Yahoo

The Yahoo data set we used is "Yahoo! Music ratings for User Selected and Randomly Selected songs", which provided by Yahoo[2], this data set contains ratings for songs collected from two different sources, which are ratings supplied by users during normal interaction with Yahoo! Music services and ratings for randomly selected songs collected during an online survey conducted by Yahoo! Research. The rating data includes 365704 ratings provided by 15,400 users to 1000 songs. In addition, the data set contains implicit responses to seven multiple-choice survey questions about rating-behavior for each of the first 5400 users.

5.1.4. Data Sets Comparison

Listed here are basic U-I Matrix Properties of the real-world data sets that we used.

Table 5.1: Comparison Among Real-World Data Sets

	Movielen 100K	Filmtrust	Yahoo	Movielens 1M
Nodes	943	1508	15400	6040
Items	1682	2071	1000	3952
Nodes/Items	0.5606	0.7282	15.4	1.5283
Nr. of Ratings	100000	35497	365704	1000209
Sparsity	93.6953%	98.8634%	97.6253%	95.8098%
Ratings Scale	1-5	0.5-4	1-5	1-5
Average Rating	3.5299	3.0028	2.7336	3.5816
User-Item Degree-Degree Correlation	-0.2157	-0.4507	-0.1113	-0.2054
Ratings STD	0.9209	0.4814	1.4585	0.9002
Degrees STD	80.3555	91.766	543.7555	372.211
Entropy of Item Ratings	1.6209	0.8703	2.0453	1.6893

[2]http://research.yahoo.com/Academic_Relations

5.2. Tools

For our experiments we used the third party tools, MyMediaLite, libFM and WrapRec. MyMediaLite and libFM are two recommender system libraries. We used WrapRec to build recommendation models and evaluate them. Moreover, Python code was written to build more user-item matrices and analyze the properties of U-I matrices.

5.2.1. MyMediaLite

MyMediaLite[3] is a free, fast and multi-purpose recommender system library, written by C# and runs on the .NET platform. It offers two common recommendation scenarios: rating predication((e.g. movies' ratings on a scale of 1 to 5 stars) and item prediction from positive-only implicit feedback(e.g. from clicks or purchase actions to predict users' preference)[14]. This library contains many state-of-the-art recommender system algorithms for those scenarios. Moreover, it offers many routines on evaluation metrics, like RMSE, MAE for the rating prediction task and MAP, NDCG for the item prediction task[20].

In this thesis, our recommendation task is rating prediction, the algorithms in MyMediaLite used in the experiments are:

- User KNN: Weighted user-based K-Nearest-Neighbors.
- Item KNN: Weighted item-based K-Nearest-Neighbors.
- Matrix Factorization: Simple matrix factorization class, factorizing the observed rating values using latent factor vectors for users and items. Model learning is achieved by stochastic gradient descent (SGD).
- Biased Matrix Factorization: An implementation of PMF[43] which adding explicit user and item bias, learning process is also performed by SGD.
- SVD++: Uses Matrix factorization which utilizes user and item biases as foundation and also takes what users have rated(integrating implicit feedback) into account.

All the algorithms have hyper-parameters that need to be set for an optimal performance. For example, different learning rates and regularization values influence the performance in a large range. In our experiment, for the matrix factorization approaches, the number of latent factor vector is 10 and each recommendation is performed for 30 iterations. The number of neighbors to take into account for predictions of neighborhood-based approaches are 40. The evaluation metric is RMSE.

5.2.2. libFM

LibFM[4] is a publicly available library for FM, which is developed by Steffen Rendle[40]. The FM approach, which was also firstly proposed by Steffen Rendle, contains three different learning method, SGD and ALS optimization as well as Bayesian inference using Markov Chain Monte Carlo (MCMC)[40]. It supports both classification and regression tasks. Mostly, the order of factorization machine is 2, which means pairwise interactions between features are used.

[3]http://www.mymedialite.net

[4]http://www.libfm.org/

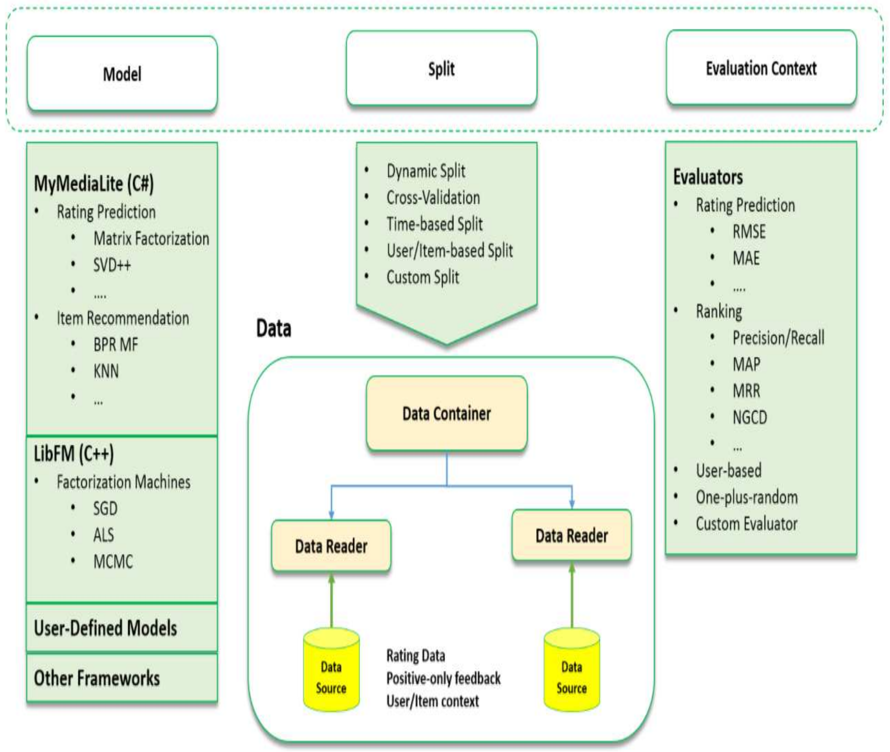


Figure 5.1: WrapRec Architecture, by Babak Loni, retrieved from *http://babakx.github.io/WrapRec/GetStarted.html*

There are also some mandatory parameters that need to be assigned, our command for the libFM command line interface is

task="r" dim="1-1-8" method="sgd" iter="30",

which means our task is regression and the learning approach we used is SGD. The dimensionality of the factorization machine is specified with *dim*. There are three parameters: k_0, k_1, k_2. $k_0 \in 0,1$ determines whether the global bias term would be used in the model(see ω_0 in equation 2.17). $k_1 \in 0,1$ denotes whether one-way interactions ω_1 (bias terms for each variable) should be used(ω_i in equation 2.17) and $k_2 \in N_0$ shows the number of factors that are used for pairwise interactions.

5.2.3. WrapRec

WrapRec[5] is an open-source toolkit for recommender systems, which is an easy implementation for users to build a recommender model and evaluate it. The toolkit is written in C# and is currently mainly developed by Babak Loni, in Delft University of Technology[29]. WrapRec is designed to wrap multiple algorithms (from different toolkits) and evaluate the models under a single evaluation framework. It performs experiments with a easily understand configuration file, including all designing choices and parameters(data, models, splits and evaluation metrics). By using WrapRec, multiple experiments can be performed in one run and a detailed experiment results file can be generated with multiple evaluation methods. The WrapRec Architecture shows in Figure 5.1,

[5]http://babakx.github.io/WrapRec/

5.2.4. Custom Code

To create more U-I matrices and obtain the properties of the U-I matrices we created, we wrote custom Python code for the this thesis project. The functions are listed below,

- Create U-I matrices based on network modification strategies mentioned in Chapter 4
- Analysis of U-I matrices statistics, including user-item degree-degree correlation, ratings STD, degrees STD, entropy of item ratings and the largest three singular values

5.3. Experiment Flow

We start our experiments based on the real-world data sets, the network modification algorithms and the tools we mentioned before. In this section, we introduce our experiment flow.

- First of all, we apply the network modification algorithms mentioned in Chapter 4 to create more U-I matrices based on 4 real-world data sets.
- Secondly, we use the third party tools to evaluate the accuracy of several classical collaborative filtering recommendation algorithms on these U-I matrices.
- Next, we obtain the properties of U-I matrices by the custom code.
- Finally, we analyze the relationship between algorithms' accuracy and the properties to identify which one affects or can possibly explain the accuracy of recommendation algorithms.

5.4. Summary

In this chapter, we introduce the real-world data sets and tools we used in our experiments. Four real-world data sets are used: Movielens 100K, Movielens 1M, Filmtrust and Yahoo Music. We use three third party tools, MyMediaLite, libFM and WrapRec, to evaluate the accuracy of several classical collaborative filtering algorithms. Custom code was written to modify real-world data sets and analyze the properties of created U-I matrices. Finally, we show the experiment flow of this thesis project.

6 Experiment Results

In the previous three chapters, we introduce the U-I matrix metrics we proposed, the network modification strategies and the experiment setup. In this chapter, we describe the main results of the experiments, discuss and analyze the results from three domains we mentioned in Chapter 3, namely, the network topology, the information domain and the spectrum.

6.1. Network Modification on Degree-Degree Correlation

In this section, we discuss the results when we modify the user-item degree-degree correlation. When we modified the original data sets with MDDC algorithm, obviously, the spectrum of U-I matrix and the entropy of item ratings are also changed. Next, we describe the results from the three domains. We use Spearman's rank correlation coefficient, which is defined as the Pearson correlation coefficient between the ranked variables, to calculate the rank correlation between the properties and the accuracy of algorithms. It measures how well the relationship between properties and the accuracy can be described by a monotonic function. In our thesis, we consider that the metric can better explain the accuracy changes, if it always has a strong rank correlation with the accuracy.

In this chapter, we take Movielens 100K as an example. The results are consistent on the other data sets. We show the results on other data sets in the Appendix. In network topology domain, Figure 6.1 shows the relationship among the accuracy of algorithms, user-item degree-degree correlation and three ratings STDs. The accuracy of algorithms almost remains constant when we modify the degree-degree correlation. The range of degree-degree correlation is not too large, since it is extremely hard to modify user-item degree-degree correlation in a large range when item degree distribution and ratings STD are fixed. The middle lines are modified from the original data set. The Spearman's rank correlation coefficient between user-item degree-degree correlation and the accuracy of algorithms are 0.1882, 0.1561, -0.0195, 0.1333, 0.3241 and 0.1266. Figure 6.2 shows similar relationships but under different degrees STD. As a result, we can approximately conclude degree-degree correlation does not affect the algorithm accuracy when ratings STD and item degree distribution are fixed. We conduct the following experiments under this assumption.

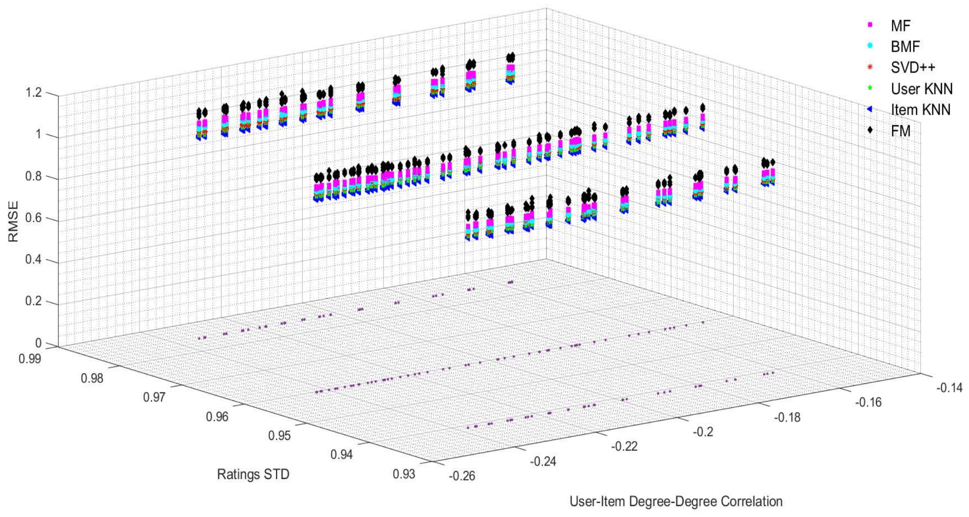


Figure 6.1: Relationship between Accuracy and User-Item Degree-Degree Correlation (Degrees STD = 80.355)

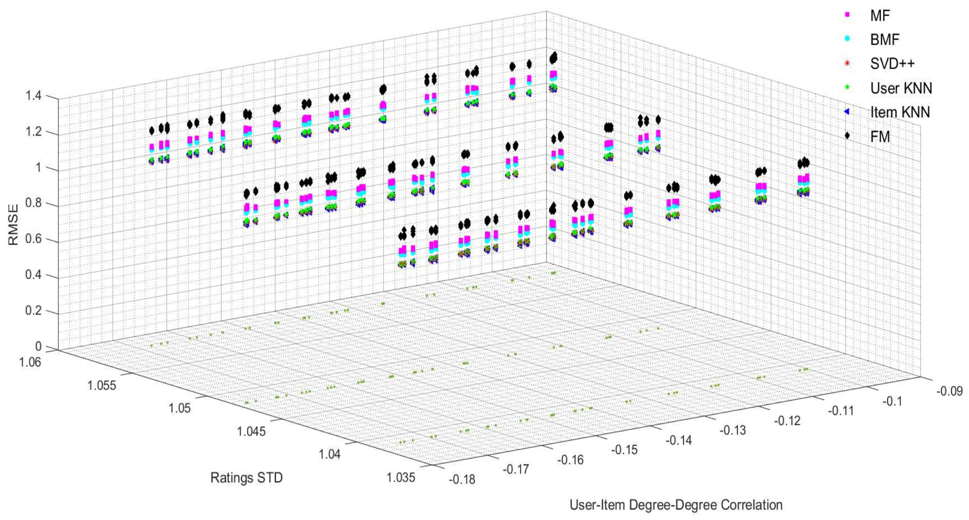


Figure 6.2: Relationship between Accuracy and User-Item Degree-Degree Correlation (Degrees STD = 54)

In information domain, our degree-degree correlation modification algorithm is equivalent to switch the user-item interactions in an item's rating set. The item rating entropy is therefore fixed. The item rating entropy stays at the same value and the accuracy of algorithms are changed in a little range when we only modify user-item degree-degree correlation. We might hold the opinion that when the entropy of item ratings is fixed, algorithm accuracy is almost fixed.

From the spectrum domain, we want to identify whether the biggest three singular values affect the accuracy of algorithms. However, no general rule is founded in this case.

6.2. Network Modification on Standard Deviation of Item Ratings

In this section, we discuss the results when we modify ratings STD. As we mentioned in Chapter 4, when we modify the original data sets with MRSTD algorithm, both the degree-degree correlation and ratings STD are changed in network topology domain. The entropy of item ratings and spectrum of U-I matrix are changed as topological properties are modified. Based on the assumption that user-item degree-degree correlation does not affect the accuracy of algorithms mentioned in Section 6.1, we only analyze the impact of ratings STD on the accuracy of algorithms in network topology domain.

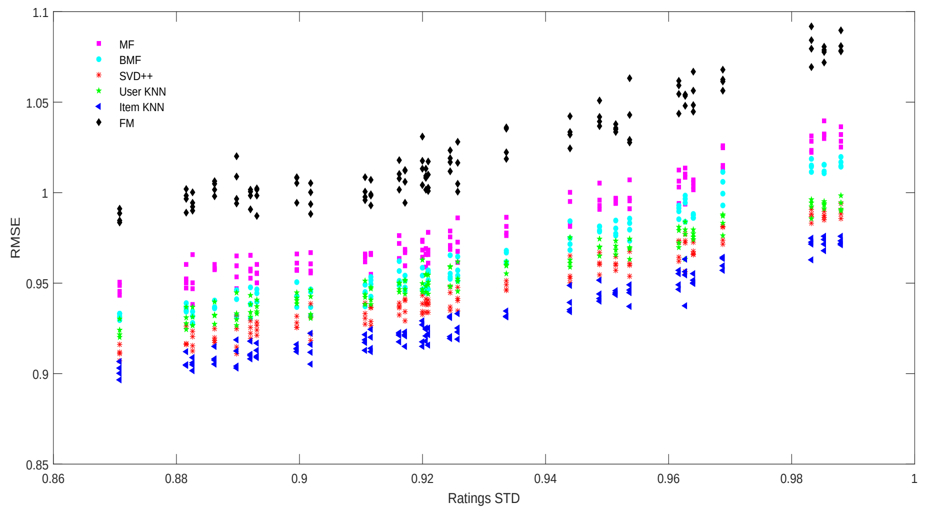


Figure 6.3: Relationship between Accuracy and Standard Deviation of Item Ratings on Movielens 100K

In Figure 6.3, we describe the relationship between algorithm accuracy and ratings STD. It seems RMSE is positive related with ratings STD. Their Spearman correlation coefficients are 0.9482, 0.9612, 0.9678, 0.9656, 0.9604 and 0.9139. We hypothesize that the more similar the ratings for an item are, the easier to provide recommendation. Suppose all the ratings for each item are the same. For example, all the ratings for item i are 3 stars and all the ratings for item j are 5 stars, we highly believe the missing ratings for item i are 3 stars and 5 stars for item j. However, this metric still has limitation. It does not consider the number of ratings for each item, that is, it does not distinguish between an item only with three hundred 3 stars ratings and an item with only ten 3 stars ratings.

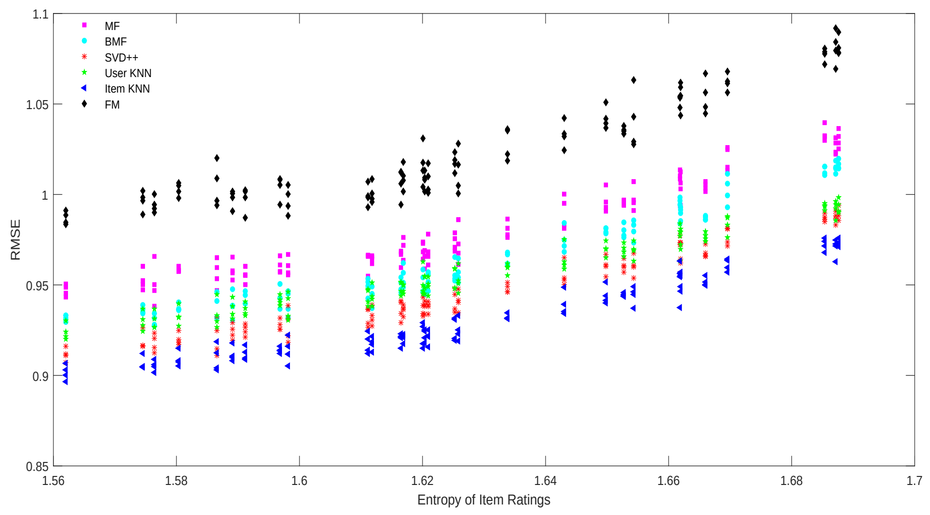


Figure 6.4: Relationship between Accuracy and Entropy of Item Ratings

Figure 6.4 shows that RMSEs of algorithms are advanced with the growth of entropy value. The Spearman correlation coefficients are 0.9476, 0.9606, 0.9677, 0.9652, 0.9605 and 0.9156. From this figure, we can reach a hypothesis that the more uncertainty the ratings of item is, the lower accuracy is. However, similar to ratings STD, it only calculate the entropy of available ratings and does not consider the number of ratings for each item.

The relationship between RMSEs and the biggest three singular value of Movielens 100k data set are described in Figure 6.5. From this figure, we can see there is an approximate linear relationship between RMSEs and the biggest singular value. The Spearman correlation coefficients are -0.917, -0.9594, -0.9659, -0.9667, -0.9601 and -0.9474.

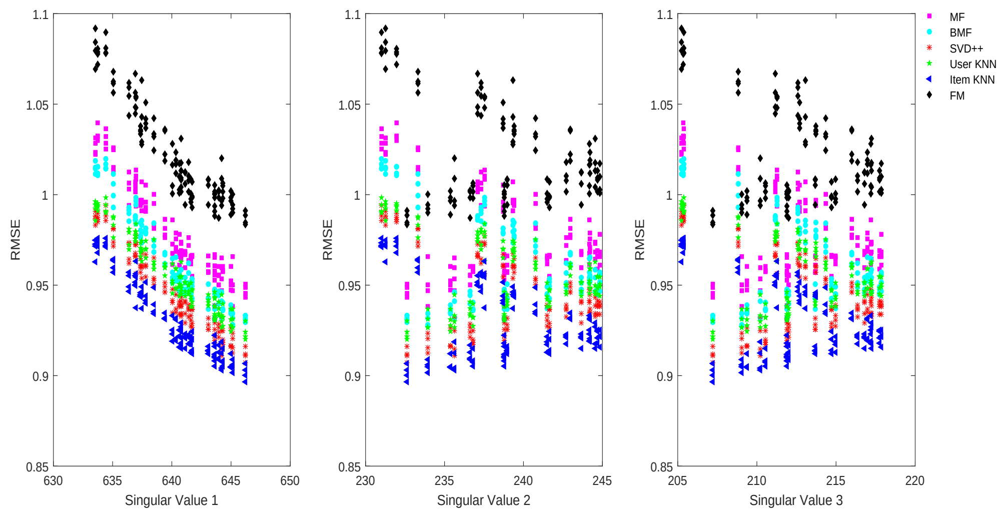


Figure 6.5: Relationship between the Accuracy of Algorithms and the Biggest 3 Singular Value on Movielens 100K

6.3. Network Modification on Item Degree Distribution

In this section, we use the MDD algorithm to modify item degree distribution. As we mentioned in Chapter 4, user-item degree-degree correlation, ratings STD and degrees STD are changed during this process. Moreover, the entropy of item ratings and spectrum are changed.

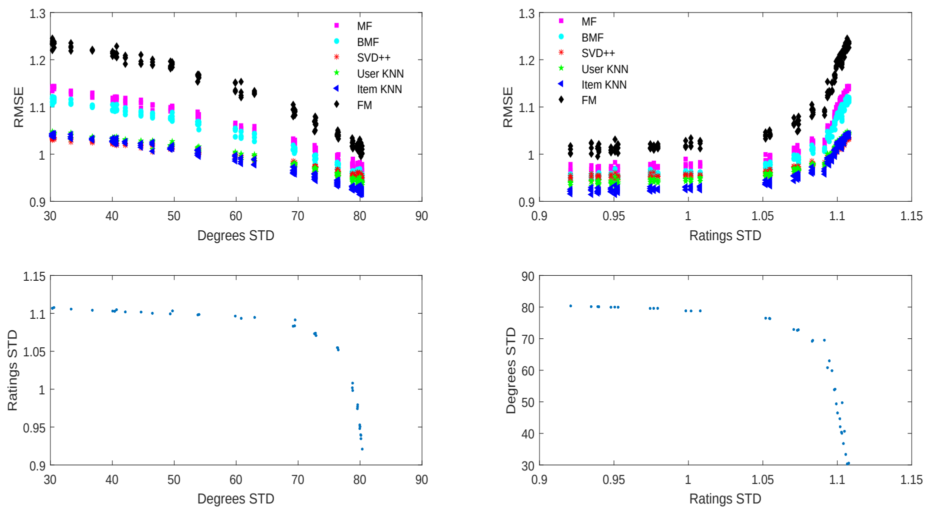


Figure 6.6: Relationship between Accuracy and Standard Deviation of Item Ratings, Standard Deviation of Item Degrees on MovieLens 100K

Figure 6.6 shows the relationship between accuracy and ratings STD/ degrees STD when we modify item degree distribution. When degrees STD is less than 70, ratings STD is almost unchanged. When ratings STD is less than 1.05, degrees STD is almost fixed. Therefore, we can see RMSE is positive related with ratings STD and negative related with degrees STD. Yoon-Joo Park and Alexander Tuzhilin hypothesized that the long tail distribution of items may affect algorithm performance, since it is hard to make prediction based on many items with only few ratings[34]. However, under our network modification strategy, our experiment results are in contrast with their hypothesis. Our results might partially prove that their hypothesis is not true when the sparsity and average rating are fixed.

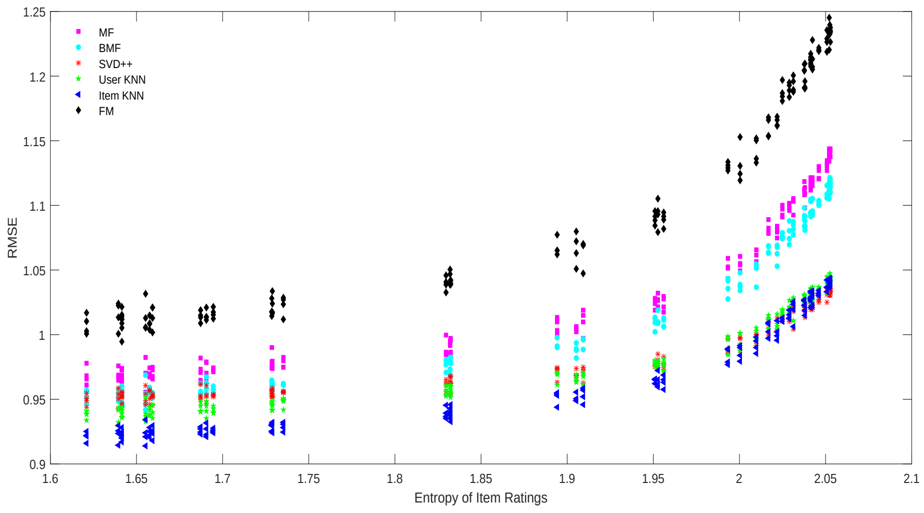


Figure 6.7: Relationship between Accuracy and Entropy of Item Ratings

From Figure 6.7, we can see, the more uncertainty the ratings of item is, the lower accuracy is. The Spearman correlation coefficient are 0.9823, 0.9829, 0.9661, 0.979, 0.9786 and 0.978. Degree distribution rewiring actually increase the uncertainty of item ratings to deduce the accuracy.

Figure 6.8 is the relationship between the accuracy of MF and the biggest 3 singular value on Movielens 100K. From this figure, we can see there is a negative correlation between RMSEs and these singular values. The ranges of Spearman correlation coefficients are -0.9844 ~ -0.9674, -0.9831 ~ -0.9675 and -0.9823 ~ -0.9667 respectively.

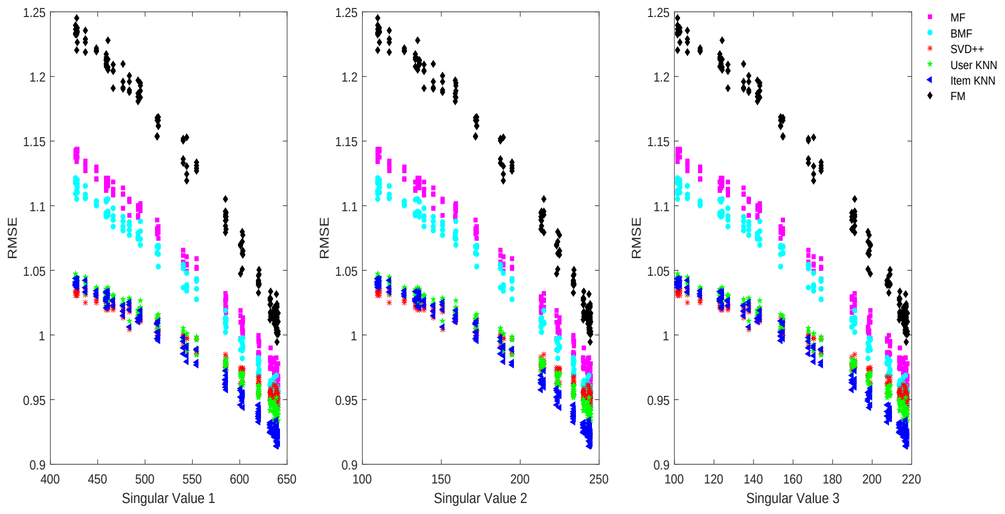


Figure 6.8: Relationship between the Accuracy of Algorithms and the Biggest 3 Singular Value on Movielens 100K

6.4. Comparison

In this section, comparing all the previous experiment results together, we aim to see which property can better explain the accuracy of recommender system algorithms. Figure 6.9, 6.10, 6.11, 6.12, 6.13

and 6.14 show the relationship between the accuracy of algorithms and the properties we mentioned before. The corresponding Spearman correlation coefficients are shown in Table 6.1. From Table 6.1, we can conclude that item rating entropy and ratings STD can better explain the accuracy of algorithms than the other properties. Entropy can measure the overall uncertainty of the U-I matrix and ratings STD is a useful metric to measure uncertainty of an item's rating from network topology. We hypothesize that the high uncertainty of data might cause the recommendation errors.

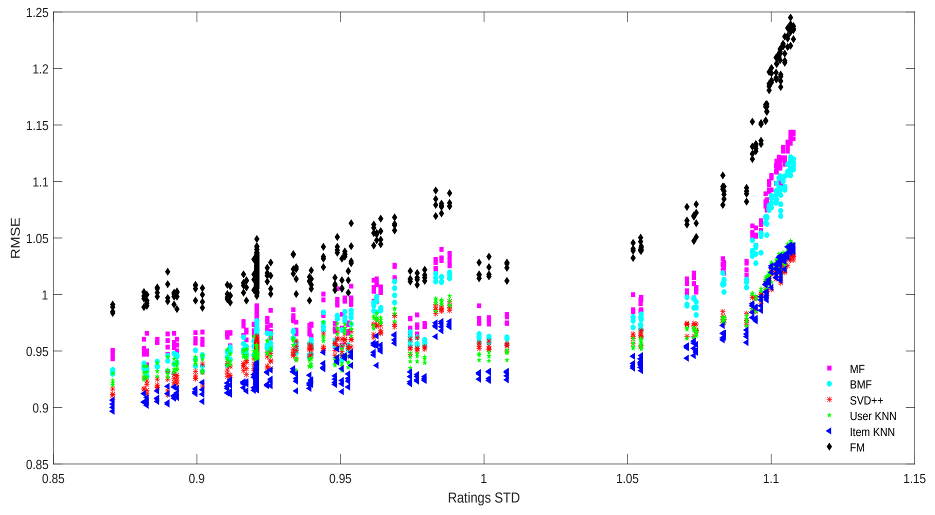


Figure 6.9: Relationship between the Accuracy of Algorithms and the STD of Item Ratings on Movielens 100K Under All Network Modification Algorithms

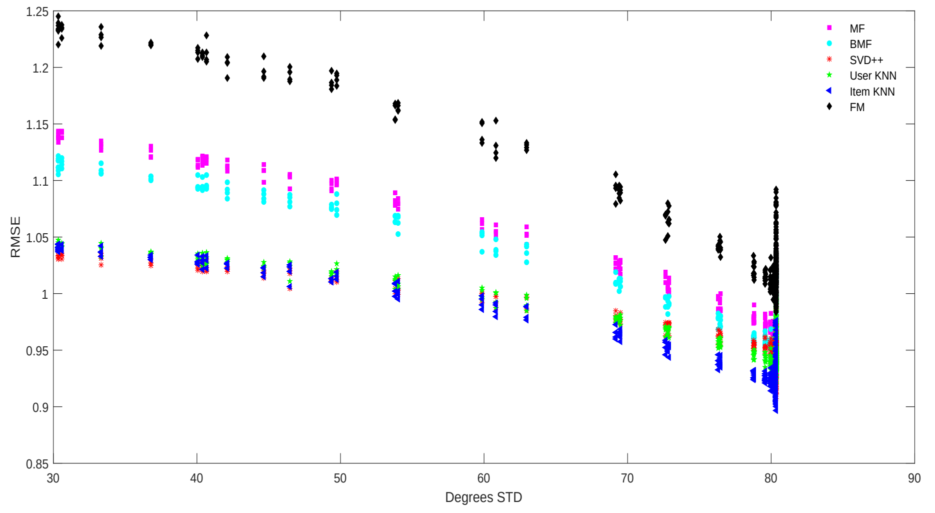


Figure 6.10: Relationship between the Accuracy of Algorithms and the STD of Item Degrees on Movielens 100K Under All Network Modification Algorithms

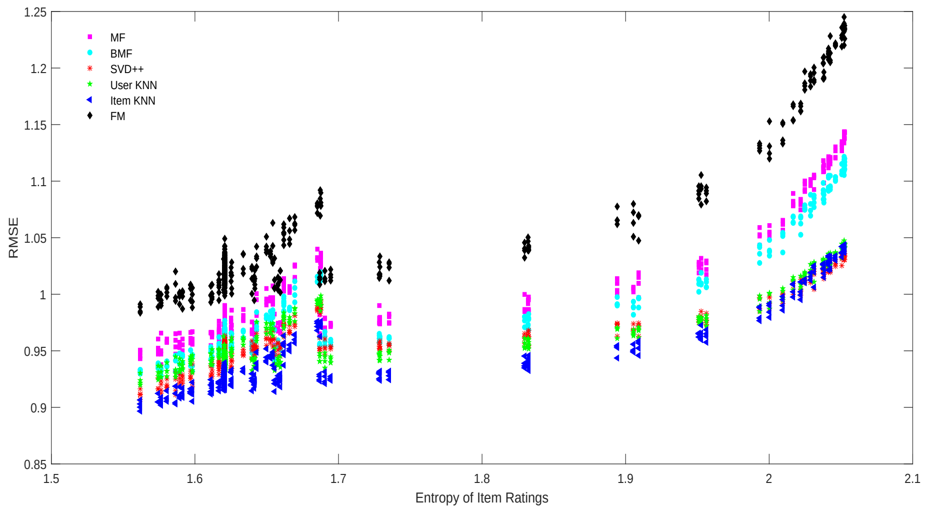


Figure 6.11: Relationship between the Accuracy of Algorithms and Item Ratings Entropy on Movielens 100K Under All Network Modification Algorithms

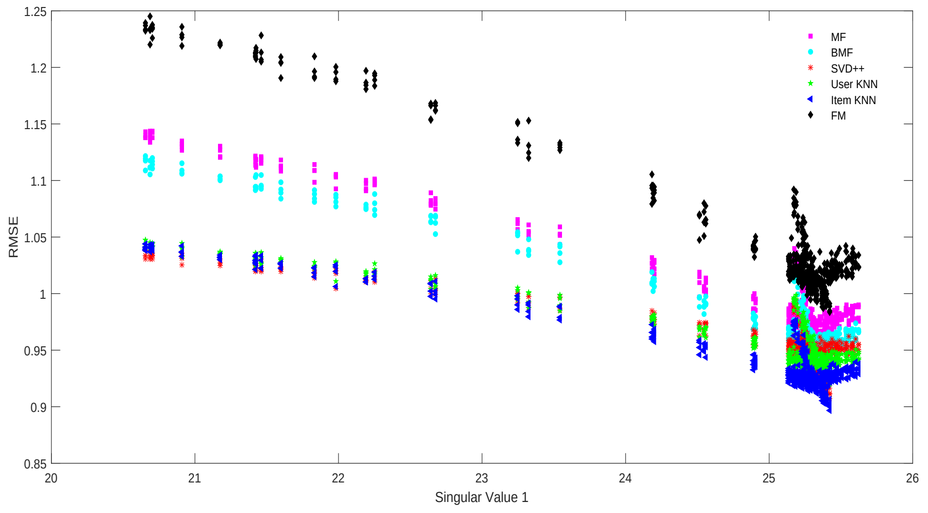


Figure 6.12: Relationship between the Accuracy of Algorithms and Singular Value 1 on Movielens 100K Under All Network Modification Algorithms

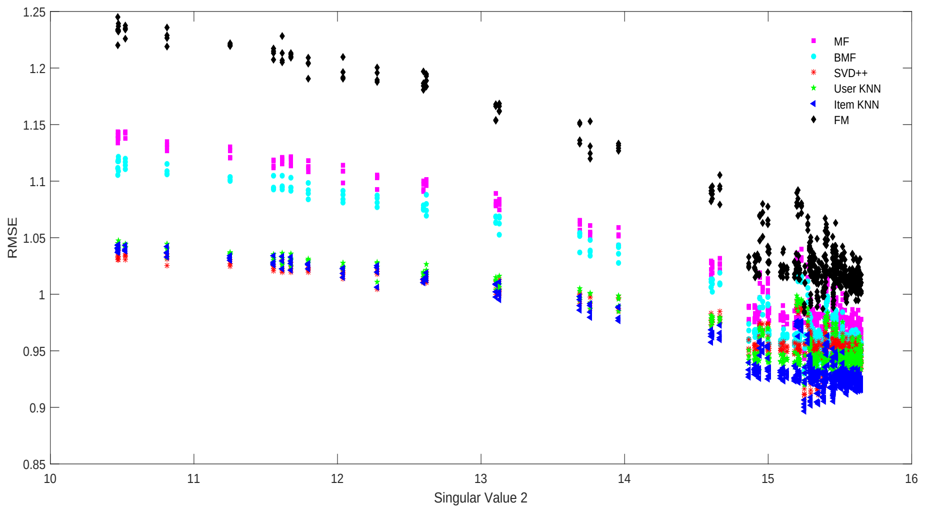


Figure 6.13: Relationship between the Accuracy of Algorithms and Singular Value 2 on Movielens 100K Under All Network Modification Algorithms

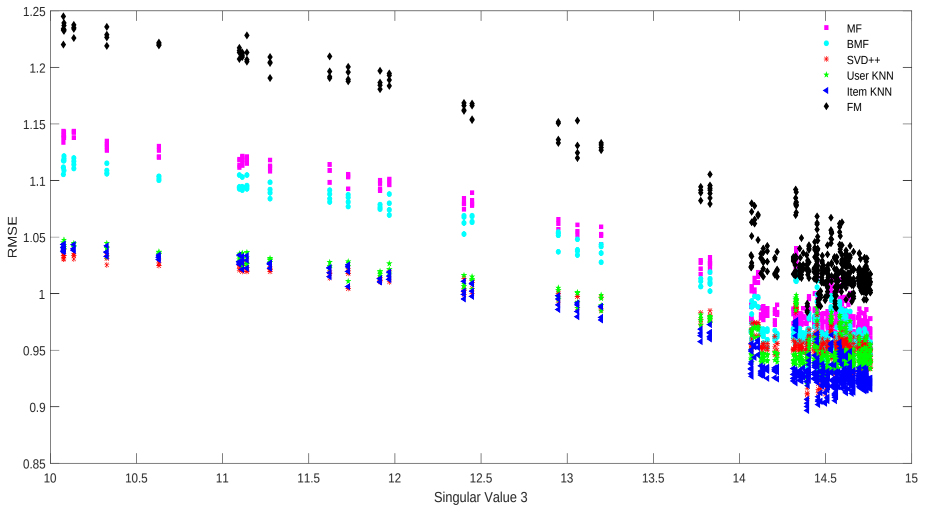


Figure 6.14: Relationship between the Accuracy of Algorithms and Singular Value 3 on Movielens 100K Under All Network Modification Algorithms

Table 6.1: Spearman correlation coefficients between the Accuracy of Algorithms and User-Item Matrix Properties

Algorithms	Ratings STD	Degrees STD	Entropy	S1	S2	S3
MF	0.7837	-0.572	0.7785	-0.6094	-0.6943	-0.7039
BMF	0.799	-0.5839	0.7945	-0.628	-0.6932	-0.7032
SVD ++	0.8338	-0.6535	0.8306	-0.672	-0.5919	-0.6
User KNN	0.7444	-0.589	0.7394	-0.6207	-0.4917	-0.4903
Item KNN	0.8045	-0.5946	0.7997	-0.5876	-0.6702	-0.6706
FM	0.7801	-0.583	0.775	-0.6233	-0.6958	-0.7064

6.5. Summary

In this chapter, we discuss and analyze the experiment results from network topology, information domain and spectrum. We measure the rank correlation between the properties and the accuracy of algorithms with Spearman correlation coefficient. The results of each network modification algorithm show that ratings STD, degrees STD, singular values and entropy of item ratings all possibly affect or explain the accuracy of algorithms. When we analyze the results of all the network modification algorithms together, we can see entropy of item ratings and ratings STD can better explain the accuracy of algorithms. We hypothesize the high uncertainty data might be the cause of recommendation errors.

7 Conclusion and Future Work

7.1. Conclusion

Recommender systems are one of the most effective tools to alleviate information overload problem. They search through a large amount of data to identify users' interest and provide users with personalized recommendations. The main focus of recommender systems' development is designing new algorithms to improve performance. However, a recommender algorithm could perform well in some platforms with its specific user item interactions captured by its user-item data/matrix but badly in other platforms. An open challenge stands: how do the properties of a U-I matrix affect or possibly explain the performance of recommender system algorithms? In order to answer this question, we start this thesis to identify which property influences the accuracy of algorithms.

In this thesis, we start a methodology to investigate which user-item matrix properties affect the accuracy of recommender system algorithms. First of all, we proposed several metrics to describe U-I matrix properties from three domains: the network topology, the information domain and the spectrum. They are introduced detailedly in Chapter 3. Since it is extremely hard to compare real-world U-I matrices, we therefore design three network modification algorithms to gradually modify user-item degree-degree correlation, ratings STD and item degrees distribution systematically, which are introduced in Chapter 4. After that, we evaluate several classical collaborative filtering algorithms on these U-I matrices we created. Based on the experiment results, we identify the impact of properties on the accuracy of recommendation algorithms.

According to our experiment results, we draw the following main conclusions:

- The accuracy of algorithms are influenced by the U-I matrix topological properties. Properties in spectrum and information domain can also approximately explain the performance of the algorithms. The effects of these properties are consistent across different collaborative filtering recommendation algorithms and various data sets in rating prediction tasks.
- When ratings STD and item degree distribution are fixed, user-item degree-degree correlation

approximately does not affect the algorithm accuracy.

- Both ratings STD and item entropy measure the uncertainty of item ratings. We find that the more uncertainty the ratings for each item are, the lower the algorithms accuracy are. The Spearman correlation coefficients between the accuracy and ratings STD, item entropy are larger than 0.7, which are strong correlations. These two metrics that measure the uncertainty of item ratings may be effective ways to explain the accuracy of algorithms.
- When the sparsity and average rating of a U-I matrix are fixed, it seems that the larger the degree STD is, the more accurate the algorithms are.
- It seems that there are positive relationships between the accuracy and the biggest 3 singular values of U-I matrices.

Our findings could help the companies to anticipate the accuracy changes of their recommender systems by analyzing the user-item interactions and understanding the evolution of properties in their data sets. Moreover, it allows the companies to strategically design their user interface and data collection (e.g., promote specific types of user-item interactions) to improve the performance of their systems.

7.2. Future Work

As a start, this thesis introduces methodologies to identify how U-I matrix property affects the accuracy of collaborative filtering recommender system algorithms and which one can better explain the accuracy changes. In our thesis, we consider that the metric can better explain the accuracy changes, if it always has a strong rank correlation with the accuracy. There are still many jobs to do for the future research. The following points are suggestions for the future work:

- Experiment with more data sets: More and more data sets with different U-I matrix properties are publicly available. Doing more experiments on these data sets are helpful to find more properties that affect the algorithms' accuracy.
- Define an new item ratings entropy: In our thesis, we only define a basic entropy that just considers the uncertainty of available ratings. However, different item has different number of ratings. The number of ratings for each item should be considered in the new entropy.
- Experiment with more U-I matrix properties: In this thesis, we just start with some basic properties. There are still a lot of properties which are need to be evaluated.
- Evaluate more performance of algorithms: In this thesis, we only focus on the accuracy of algorithms. However, in real life, other performances, such as time complexity, are also important for a recommender system.

A

Appendix

A.1. Results Under Modifying User-Item Degree-Degree Correlation

A.1.1. Filmtrust

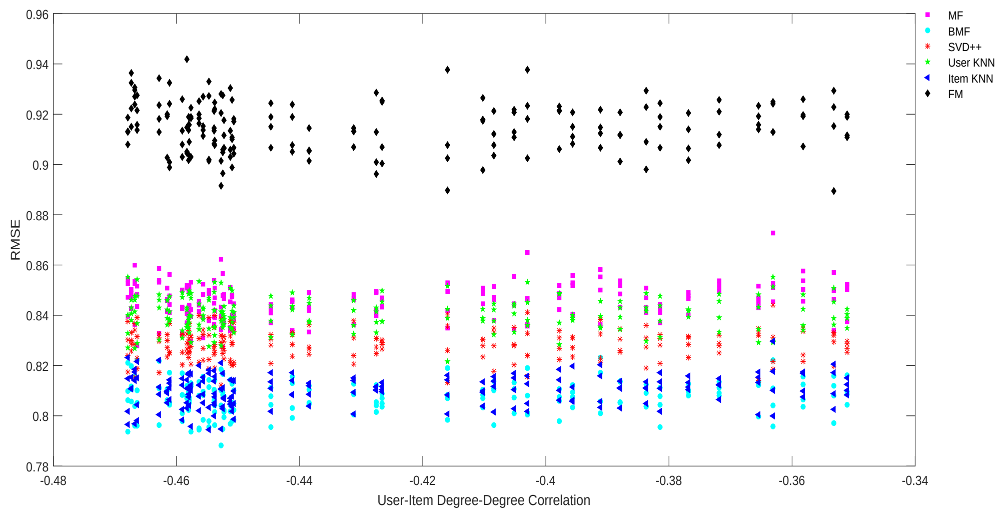


Figure A.1: Relationship between Accuracy and Degree-Degree Correlation on Filmtrust

A.1.2. Yahoo

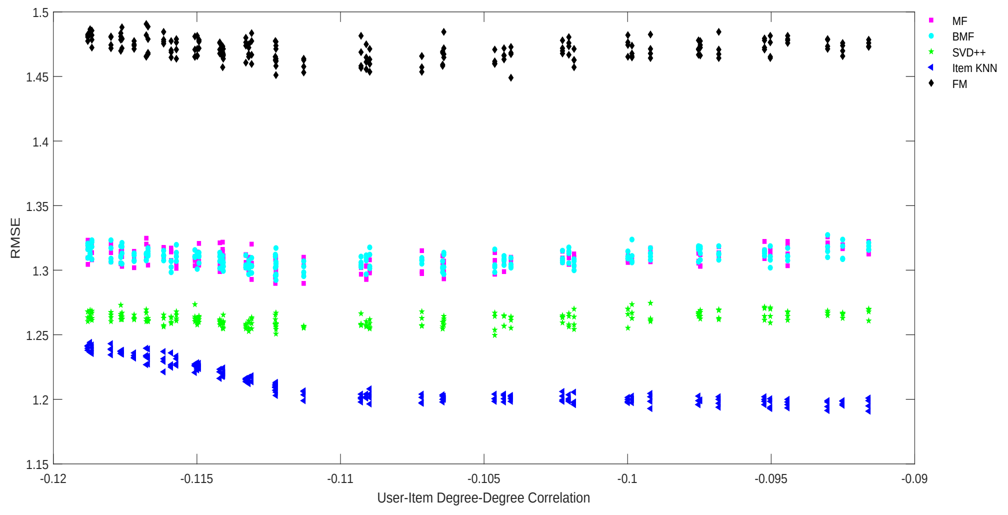


Figure A.2: Relationship between Accuracy and Degree-Degree Correlation on Yahoo

A.1.3. Movielen 1M

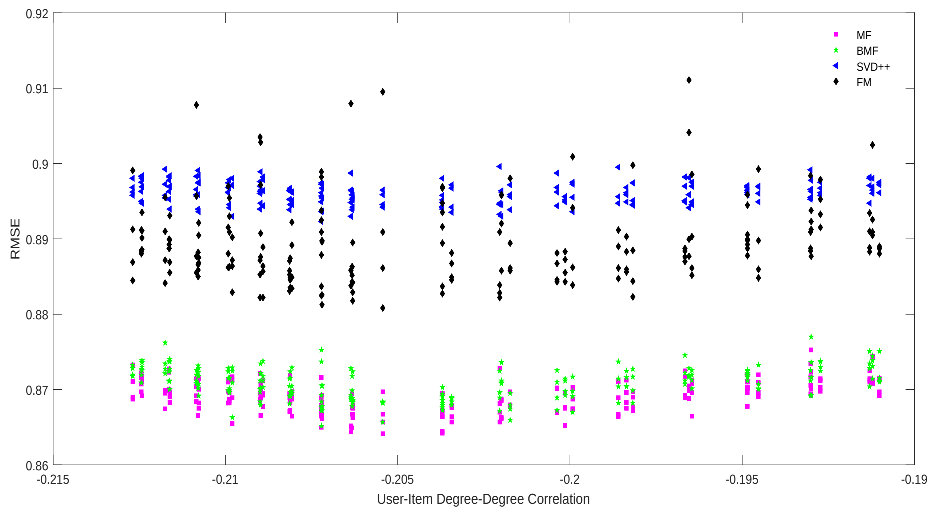


Figure A.3: Relationship between Accuracy and Degree-Degree Correlation on Movielens 1M

A.2. Results Under Modifying Ratings STD

A.2.1. Filmtrust

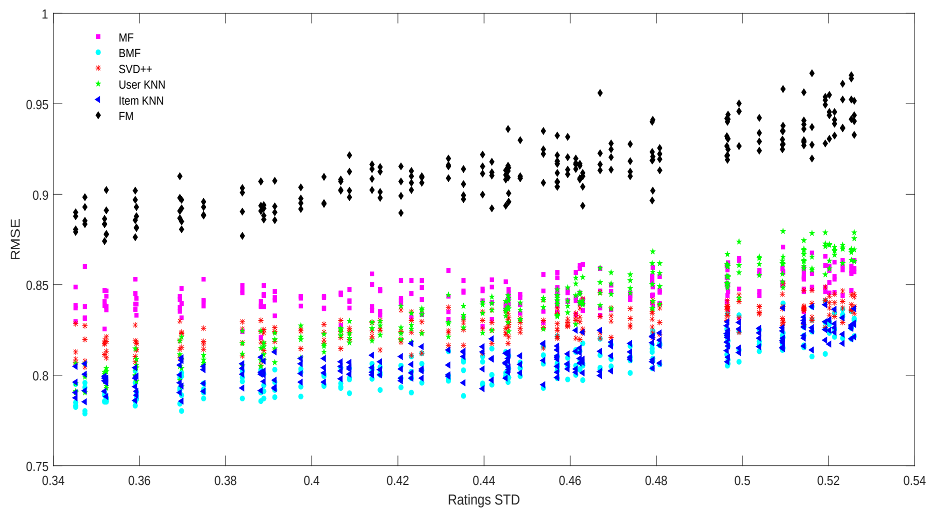


Figure A.4: Relationship between Accuracy and Standard Deviation of Item Ratings on Filmtrust

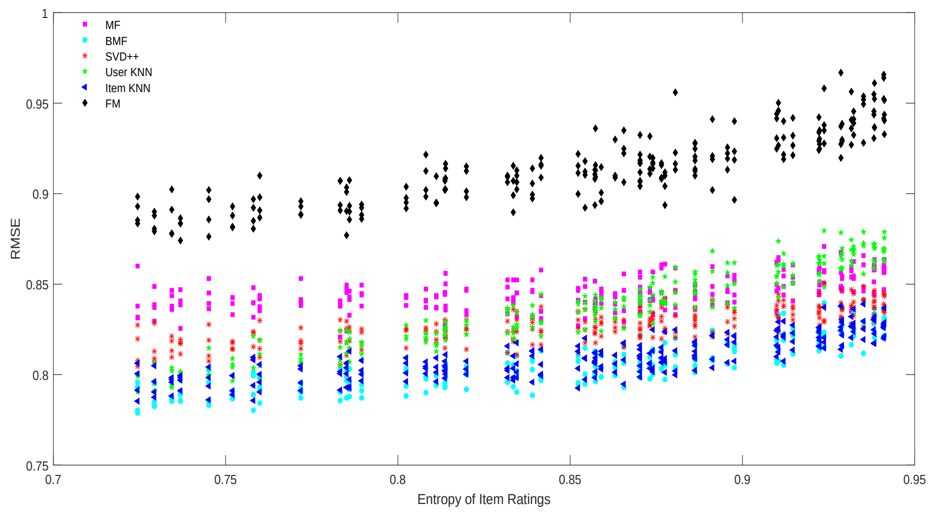


Figure A.5: Relationship between Accuracy and Entropy of Item Ratings on Filmtrust

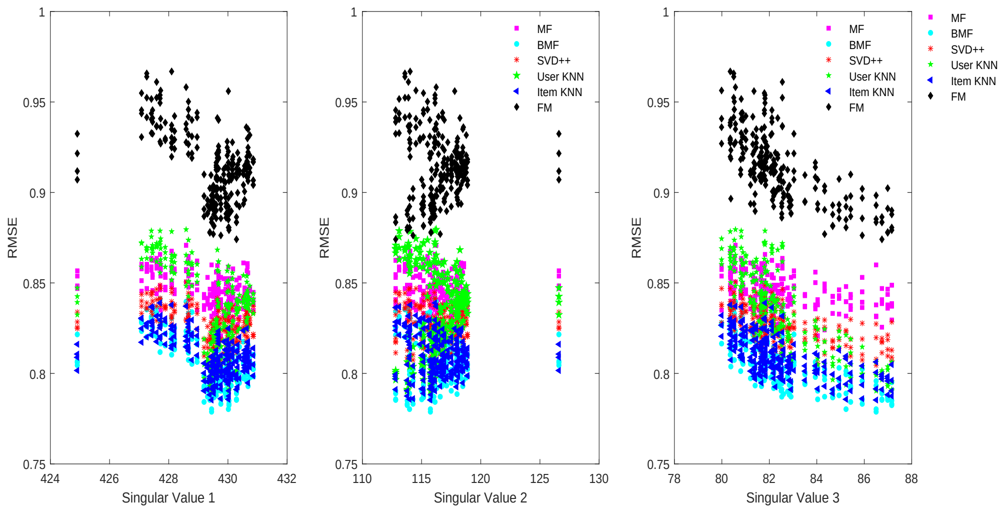


Figure A.6: Relationship between the Accuracy of Algorithms and the Biggest 3 Singular Value on Filmtrust

A.2.2. Yahoo

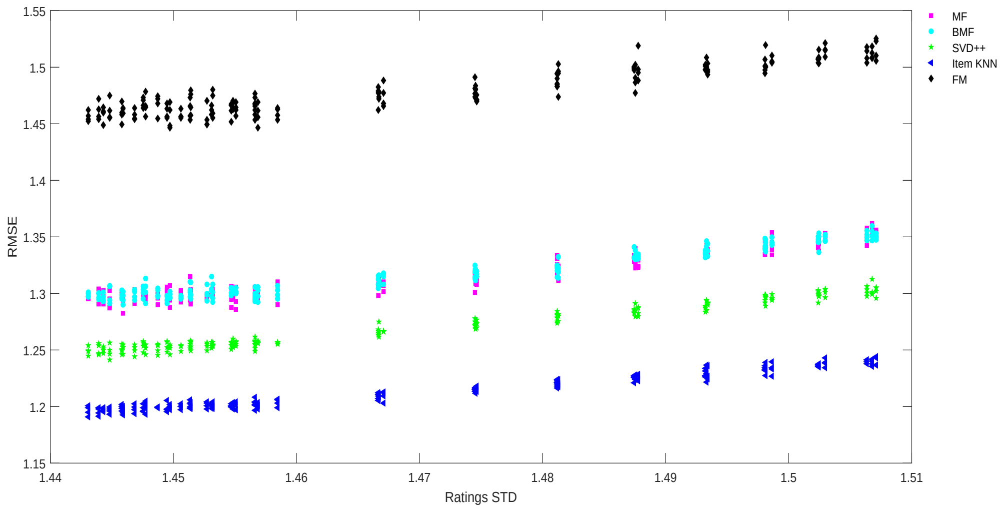


Figure A.7: Relationship between Accuracy and Standard Deviation of Item Ratings on Yahoo

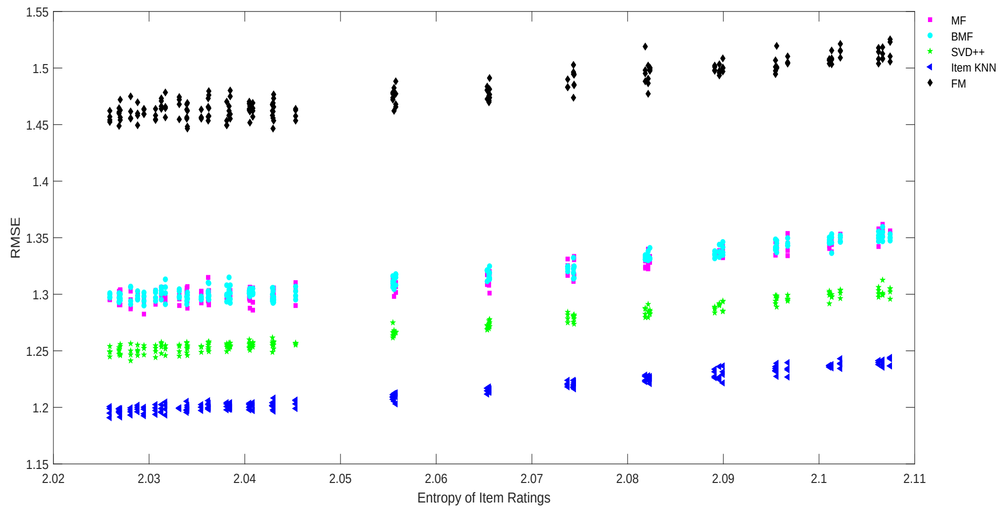


Figure A.8: Relationship between Accuracy and Entropy of Item Ratings on Yahoo

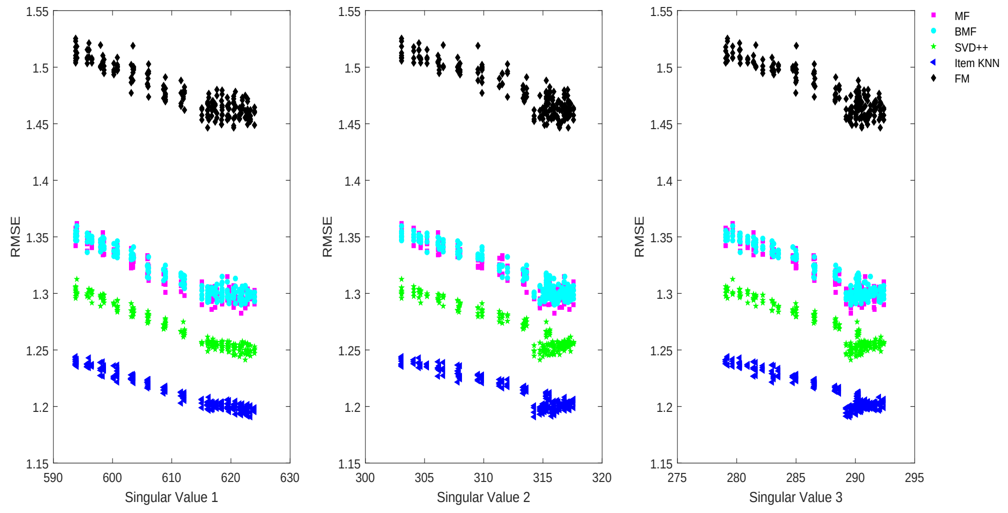


Figure A.9: Relationship between the Accuracy of Algorithms and the Biggest 3 Singular Value on Yahoo

A.2.3. Movielen 1M

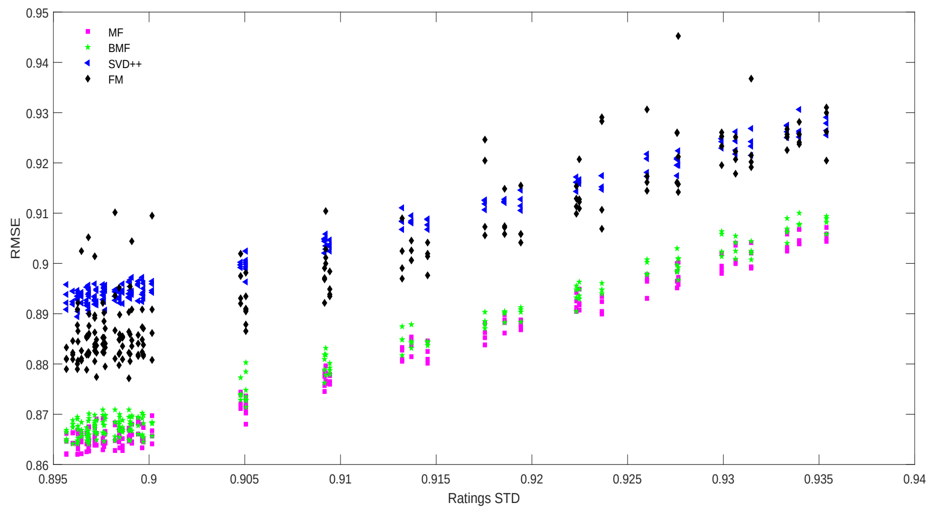


Figure A.10: Relationship between Accuracy and Standard Deviation of Item Ratings on Movielens 1M

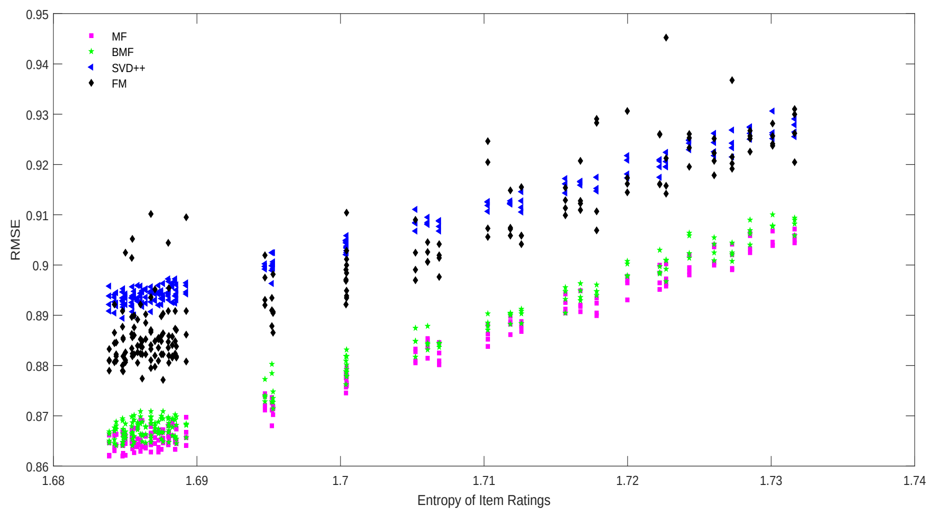


Figure A.11: Relationship between Accuracy and Entropy of Item Ratings on Movielens 1M

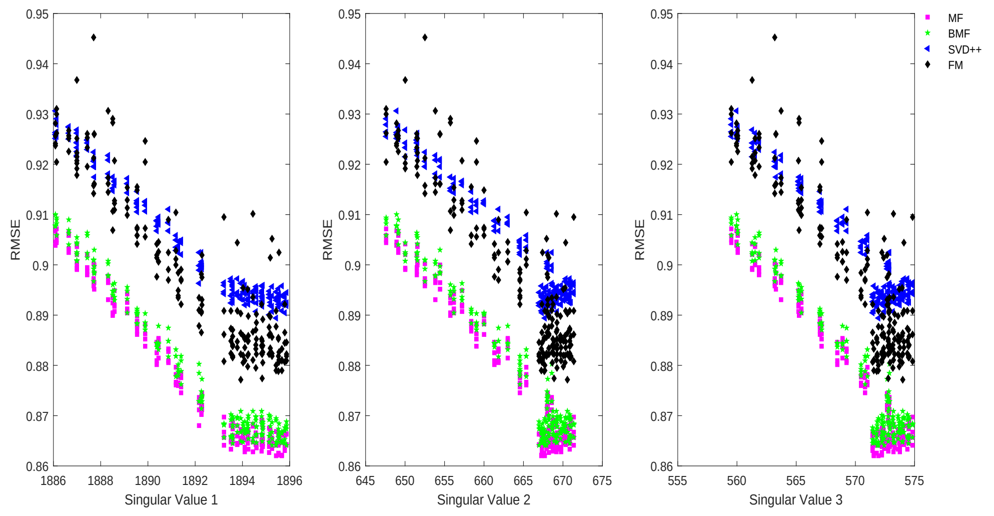


Figure A.12: Relationship between the Accuracy of Algorithms and the Biggest 3 Singular Value on Movielens 1M

A.3. Results Under Modifying Item Degree Distribution

A.3.1. Filmtrust

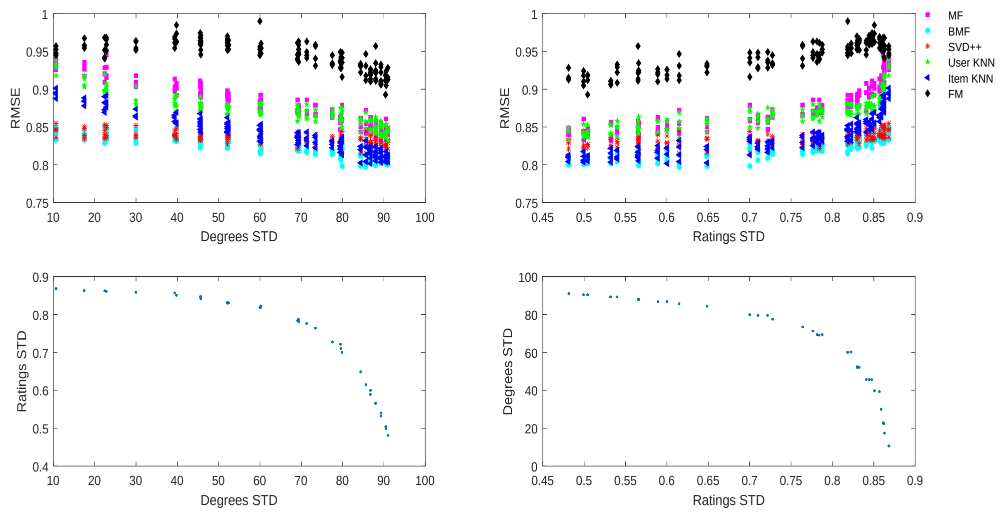


Figure A.13: Relationship between Accuracy and Ratings STD, Degrees STD on Filmtrust

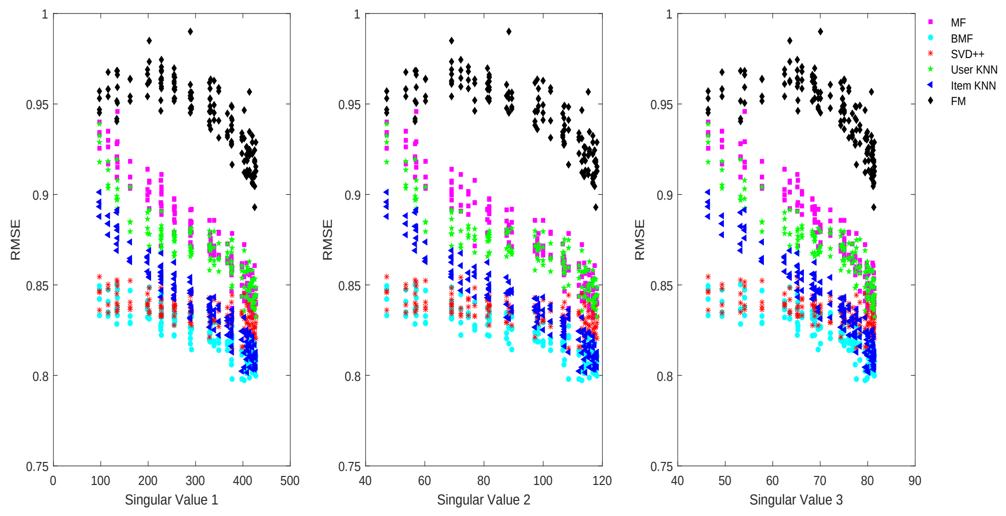


Figure A.14: Relationship between the Accuracy of Algorithms and the Biggest 3 Singular Value on Filmtrust

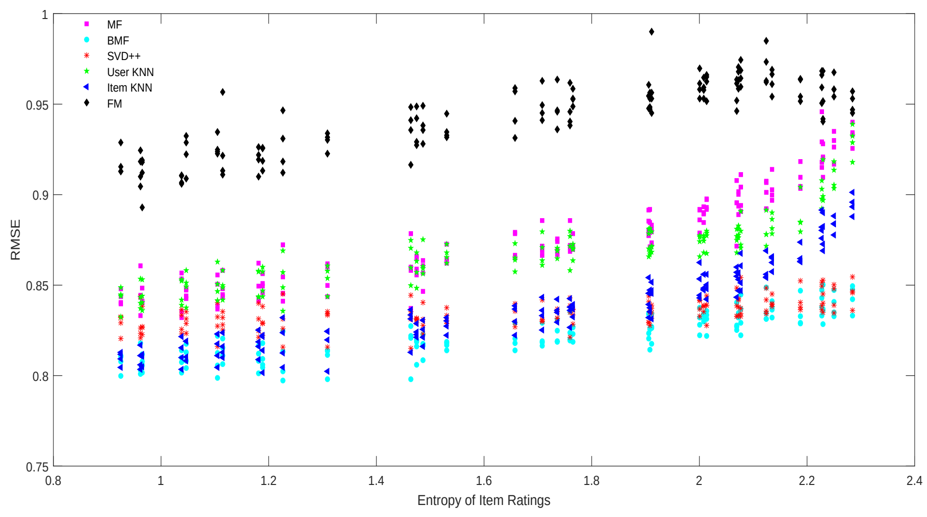


Figure A.15: Relationship between Accuracy and Entropy of Item Ratings on Filmtrust

A.3.2. Yahoo

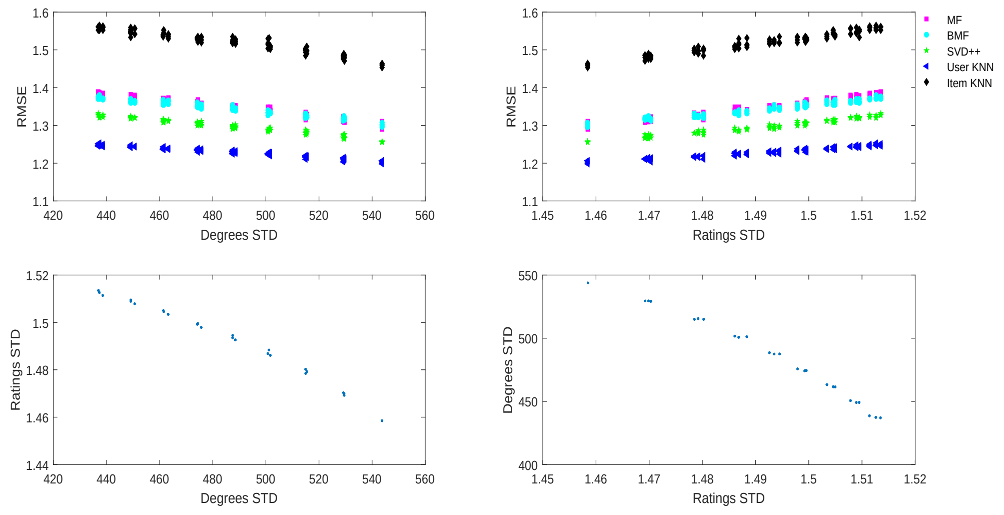


Figure A.16: Relationship between Accuracy and Ratings STD, Degrees STD on Yahoo

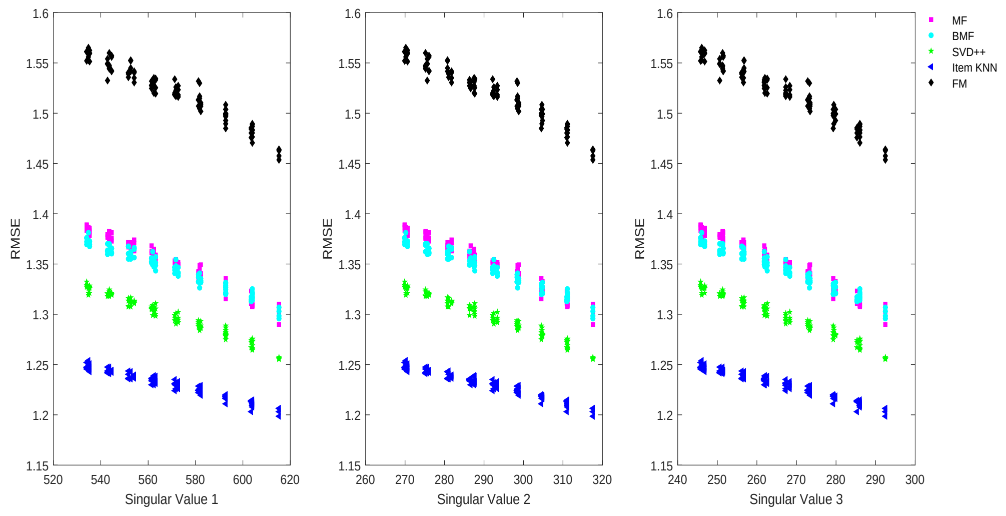


Figure A.17: Relationship between the Accuracy of Algorithms and the Biggest 3 Singular Value on Yahoo

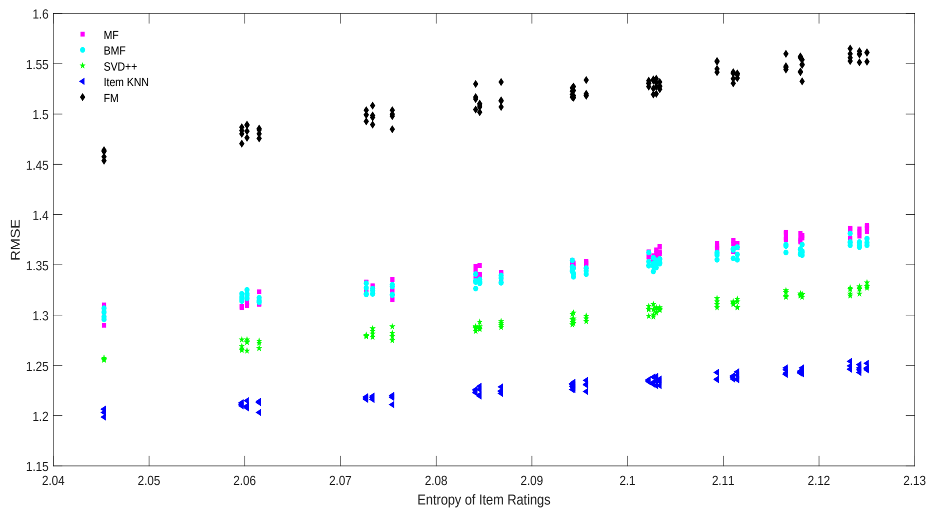


Figure A.18: Relationship between Accuracy and Entropy of Item Ratings on Yahoo

A.3.3. Movielen 1M

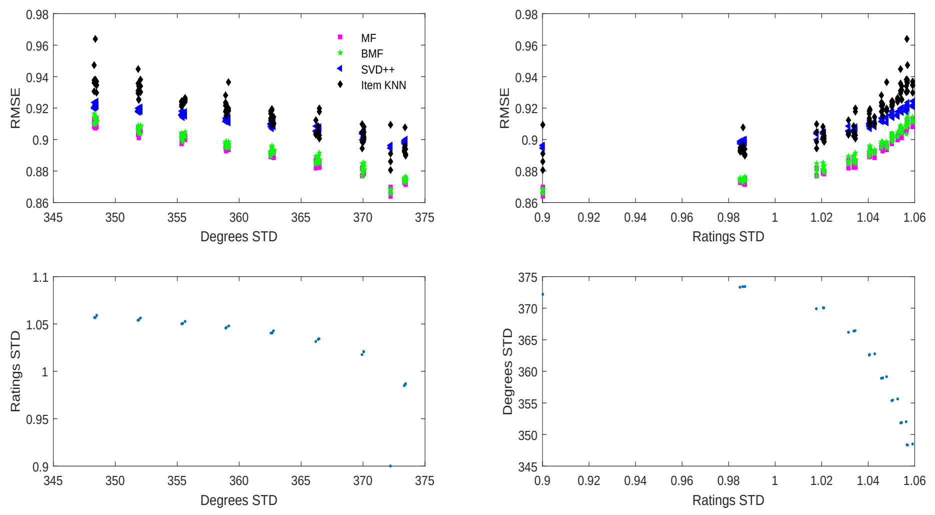


Figure A.19: Relationship between Accuracy and Ratings STD, Degrees STD on Movielens 1M

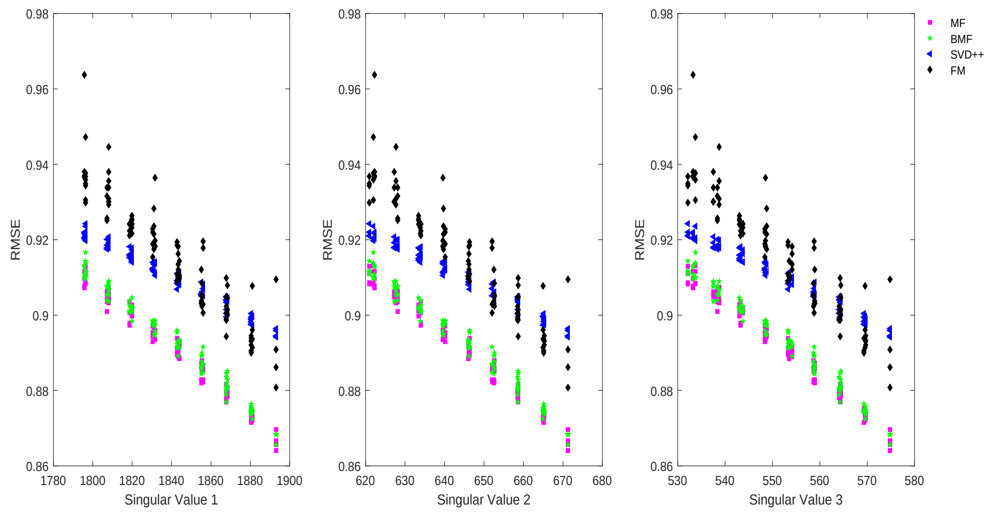


Figure A.20: Relationship between the Accuracy of Algorithms and the Biggest 3 Singular Value on Movielens 1M

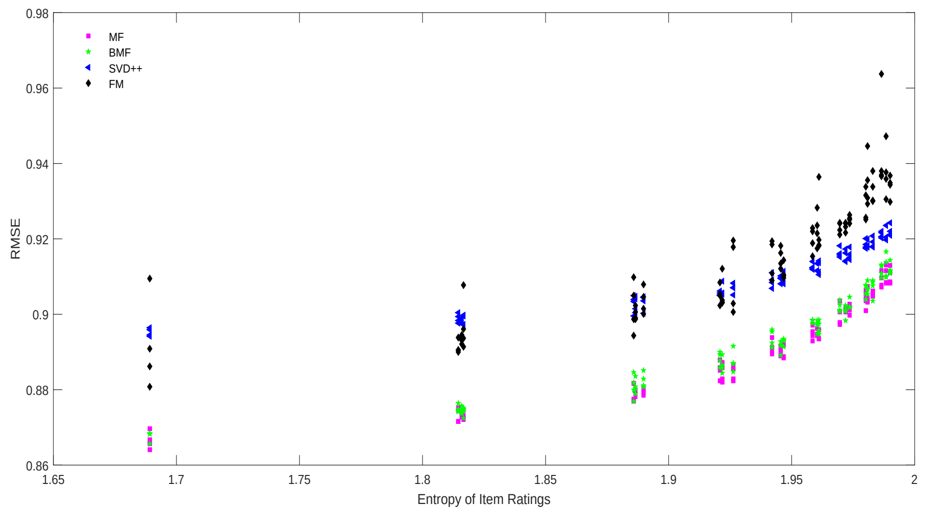


Figure A.21: Relationship between Accuracy and Entropy of Item Ratings on Movielens 1M

A.4. Results Comparison

A.4.1. Filmtrust

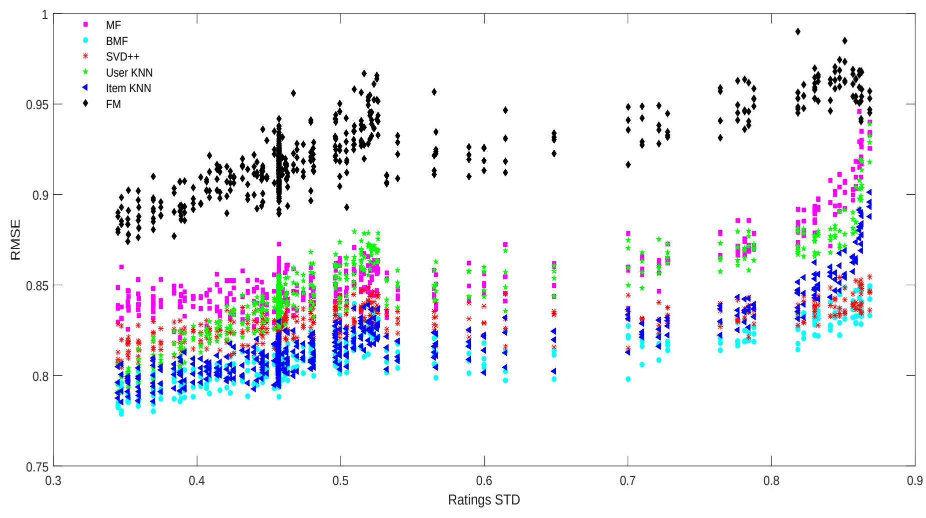


Figure A.22: Relationship between the Accuracy of Algorithms and the Ratings STD on Filmtrust Under All Network Modification Algorithms

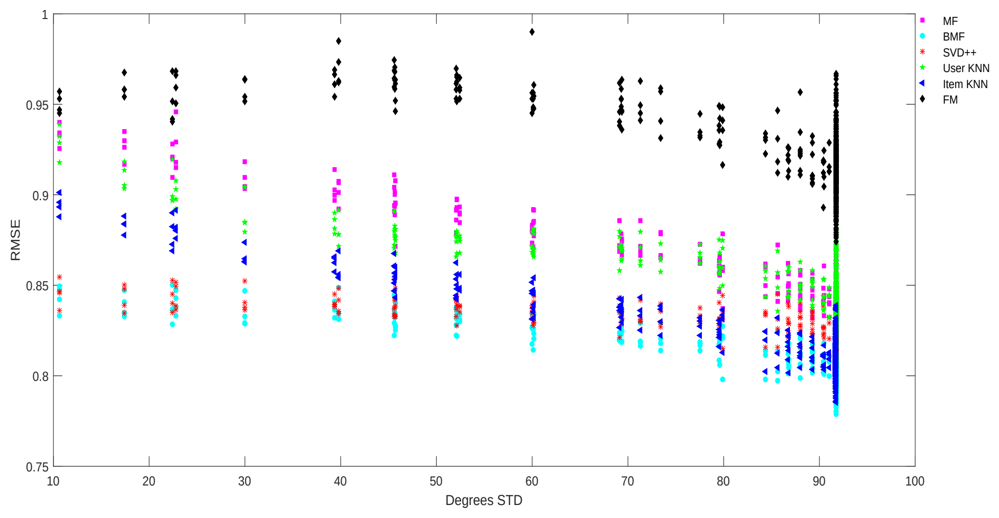


Figure A.23: Relationship between the Accuracy of Algorithms and the Degrees STD on Filmtrust Under All Network Modification Algorithms

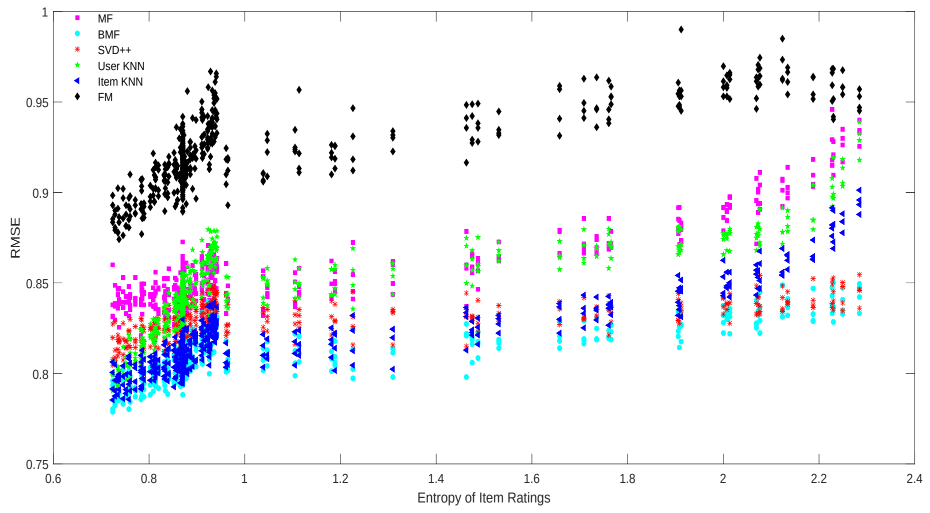


Figure A.24: Relationship between the Accuracy of Algorithms and Entropy of Item Ratings on Filmtrust Under All Network Modification Algorithms

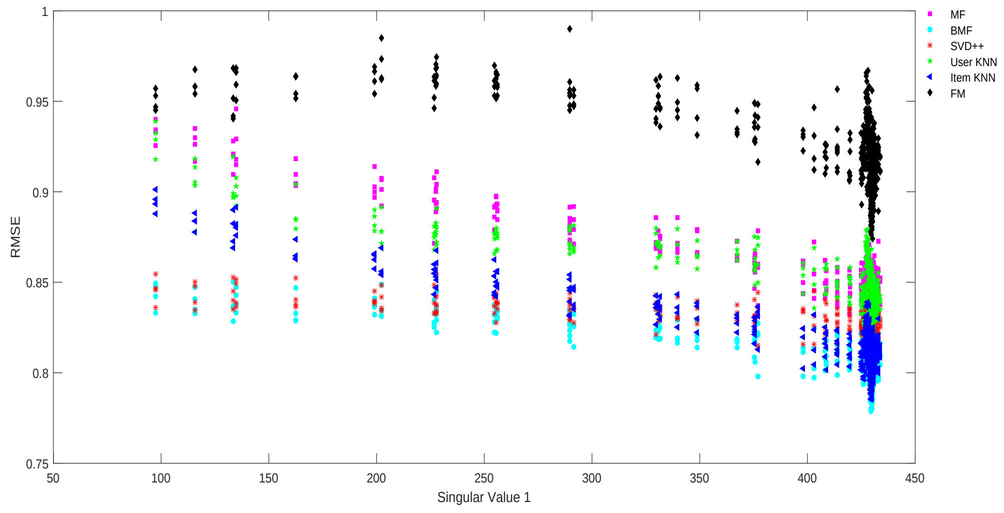


Figure A.25: Relationship between the Accuracy of Algorithms and Singular Value 1 on Filmtrust Under All Network Modification Algorithms

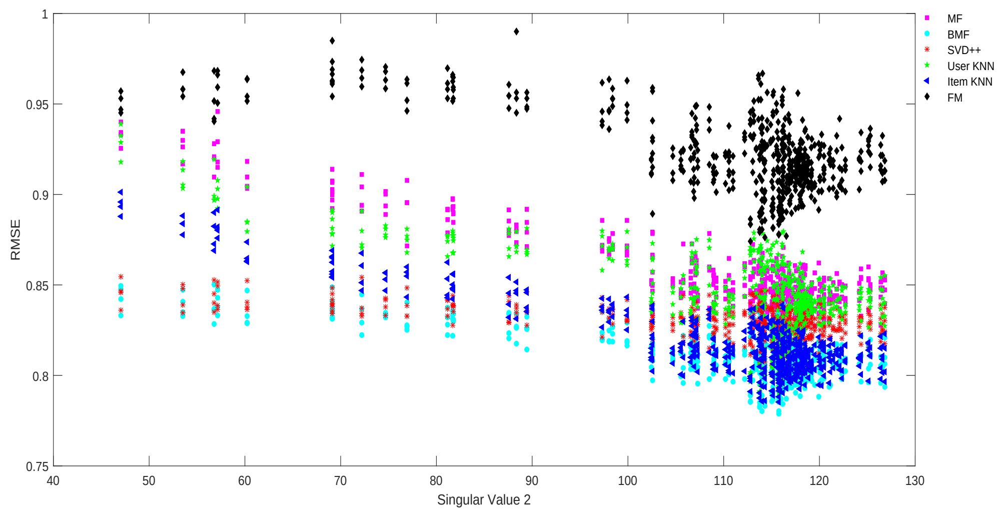


Figure A.26: Relationship between the Accuracy of Algorithms and Singular Value 2 on Filmtrust Under All Network Modification Algorithms

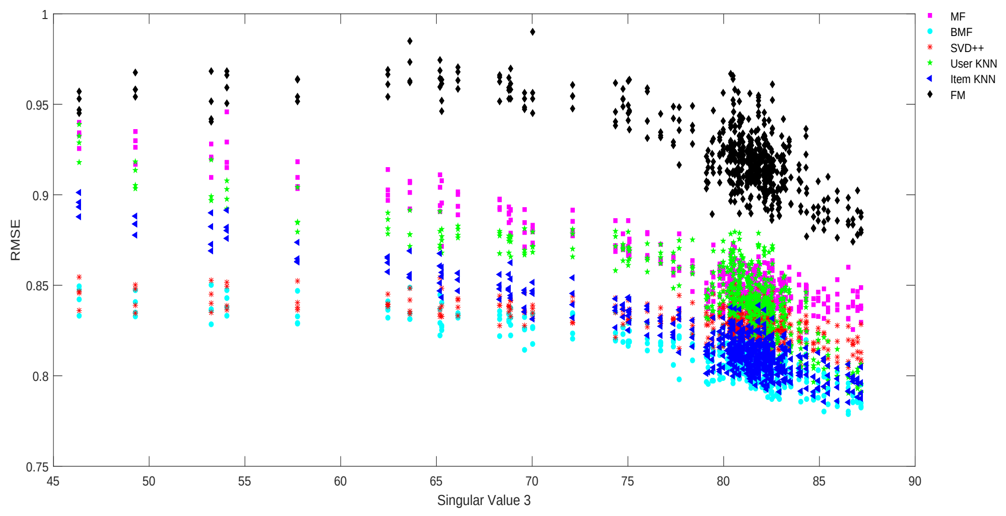


Figure A.27: Relationship between the Accuracy of Algorithms and Singular Value 3 on Filmtrust Under All Network Modification Algorithms

Table A.1: Spearman correlation coefficients between the Accuracy of Algorithms and U-I Matrix Properties

Algorithms	Ratings STD	Degrees STD	Entropy	S1	S2	S3
MF	0.6501	-0.5788	0.6448	-0.5181	-0.5248	-0.5816
BMF	0.7807	-0.5254	0.7735	-0.5117	-0.4399	-0.6431
SVD ++	0.6114	-0.3811	0.6057	-0.3734	-0.3025	-0.5005
User KNN	0.8867	-0.6204	0.8808	-0.5812	-0.4518	-0.7188
Item KNN	0.7946	-0.6048	0.7882	-0.5479	-0.5017	-0.6816
FM	0.8033	-0.5814	0.7962	-0.5739	-0.4346	-0.6529

A.4.2. Yahoo

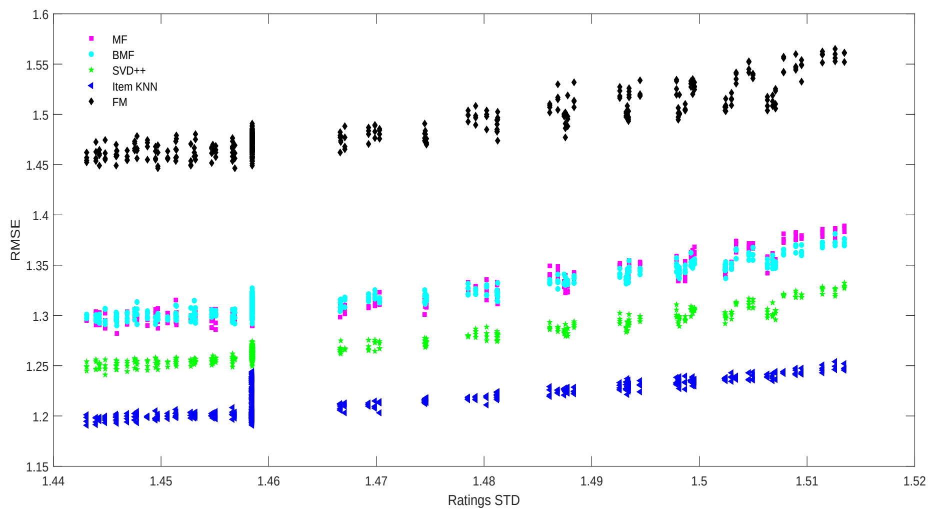


Figure A.28: Relationship between the Accuracy of Algorithms and the Ratings STD on Yahoo Under All Network Modification Algorithms

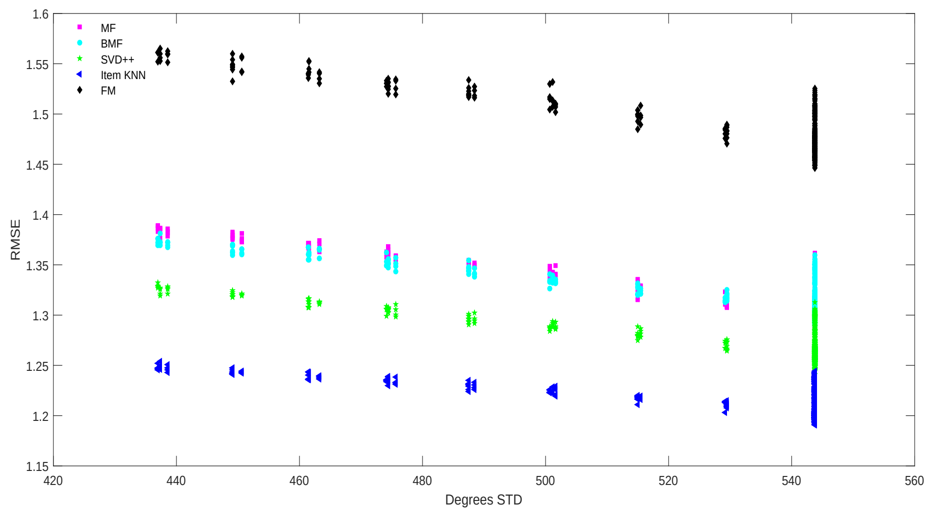


Figure A.29: Relationship between the Accuracy of Algorithms and the Degrees STD on Yahoo Under All Network Modification Algorithms

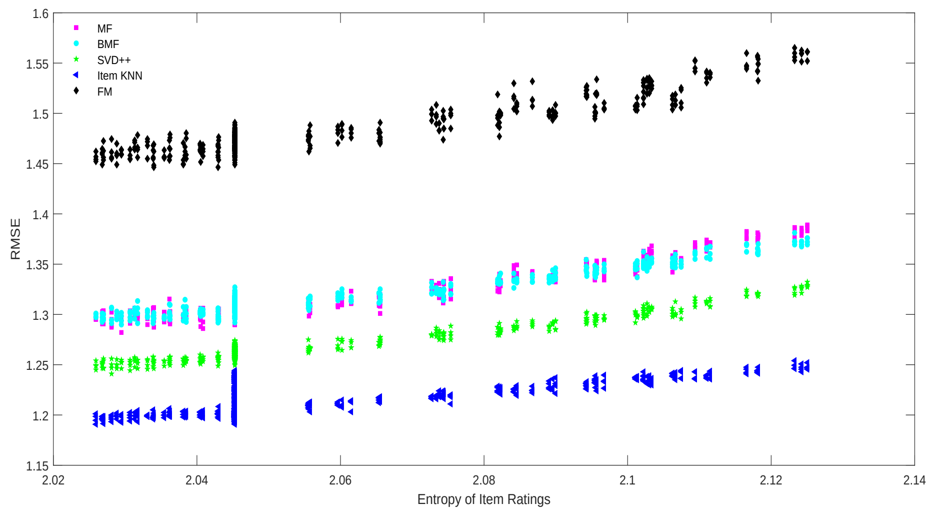


Figure A.30: Relationship between the Accuracy of Algorithms and Entropy of Item Ratings on Yahoo Under All Network Modification Algorithms

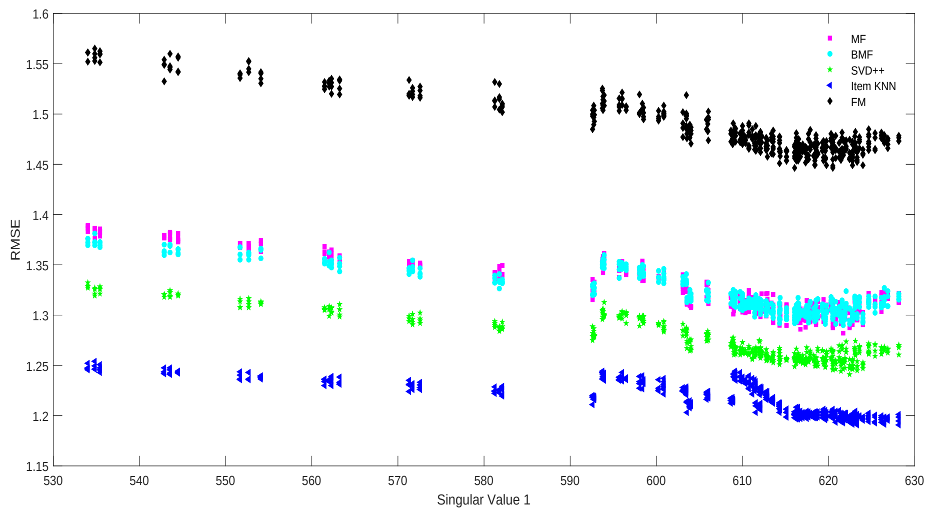


Figure A.31: Relationship between the Accuracy of Algorithms and Singular Value 1 on Yahoo Under All Network Modification Algorithms

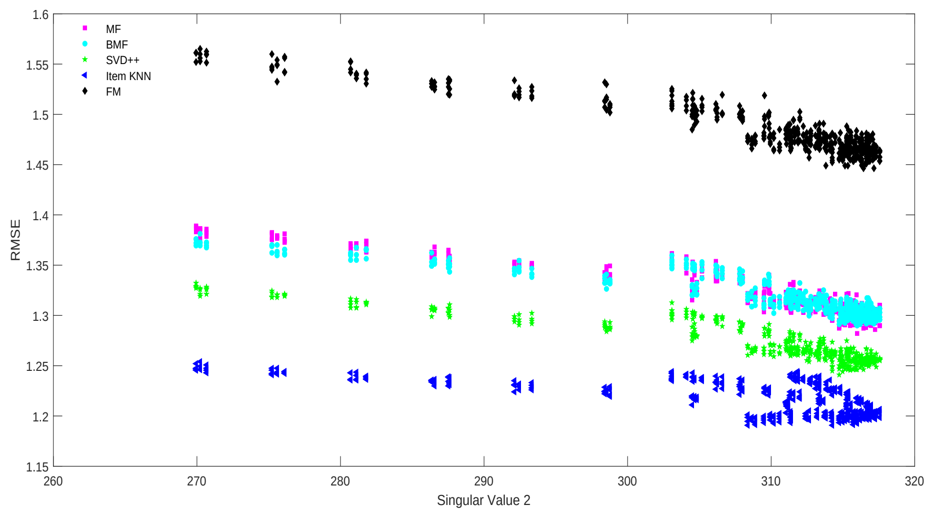


Figure A.32: Relationship between the Accuracy of Algorithms and Singular Value 2 on Yahoo Under All Network Modification Algorithms

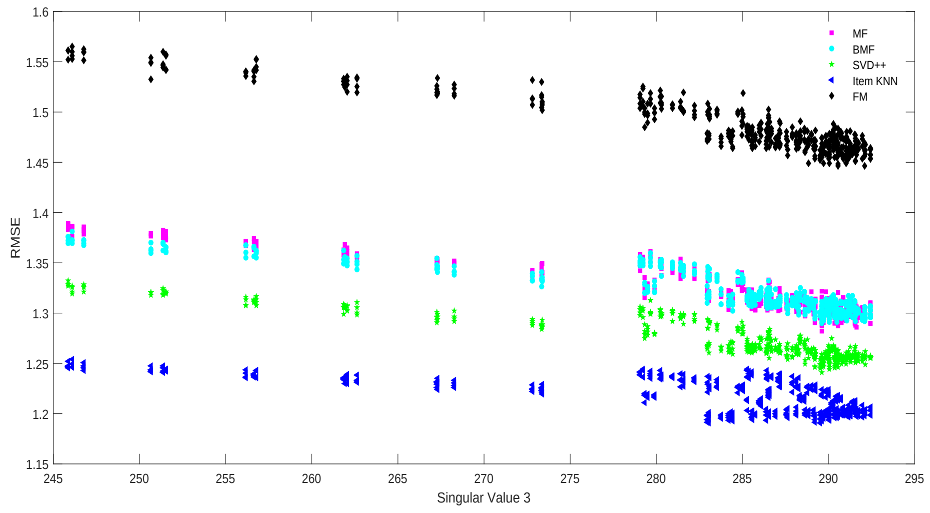


Figure A.33: Relationship between the Accuracy of Algorithms and Singular Value 3 on Yahoo Under All Network Modification Algorithms

Table A.2: Spearman correlation coefficients between the Accuracy of Algorithms and U-I Matrix Properties

Algorithms	Ratings STD	Degrees STD	Entropy	S1	S2	S3
MF	0.8725	-0.6105	0.8749	-0.7492	-0.8894	-0.8846
BMF	0.8977	-0.6042	0.8986	-0.7652	-0.8771	-0.8678
SVD ++	0.93	-0.6127	0.9314	-0.7807	-0.8775	-0.8645
Item KNN	0.751	-0.4676	0.7501	-0.8788	-0.6152	-0.6155
FM	0.8513	-0.6411	0.8546	-0.7949	-0.843	-0.8367

A.4.3. Movielen 1M

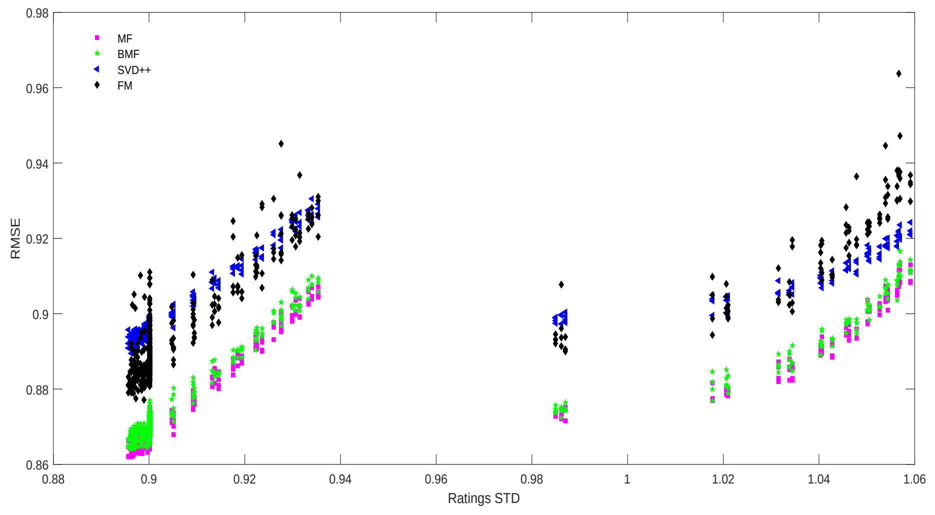


Figure A.34: Relationship between the Accuracy of Algorithms and the Ratings STD on Movielens 1M Under All Network Modification Algorithms

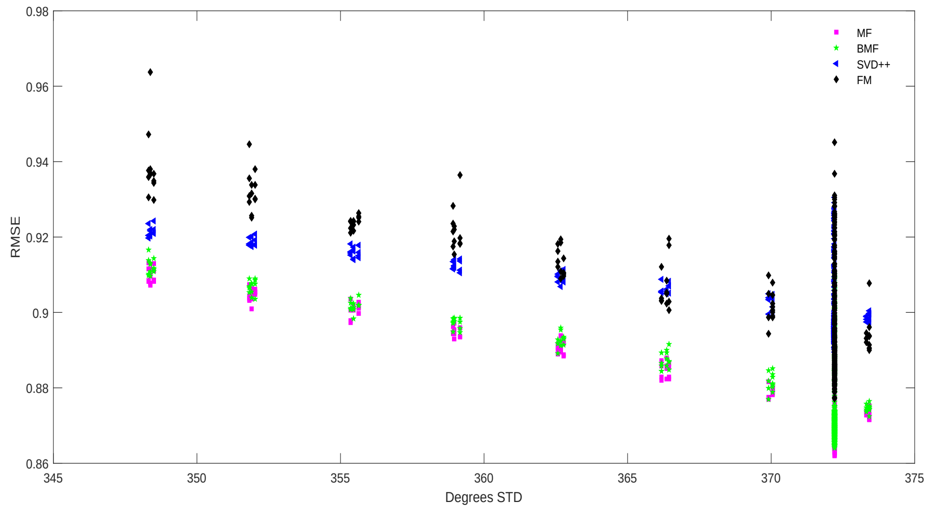


Figure A.35: Relationship between the Accuracy of Algorithms and the Degrees STD on Movielens 1M Under All Network Modification Algorithms

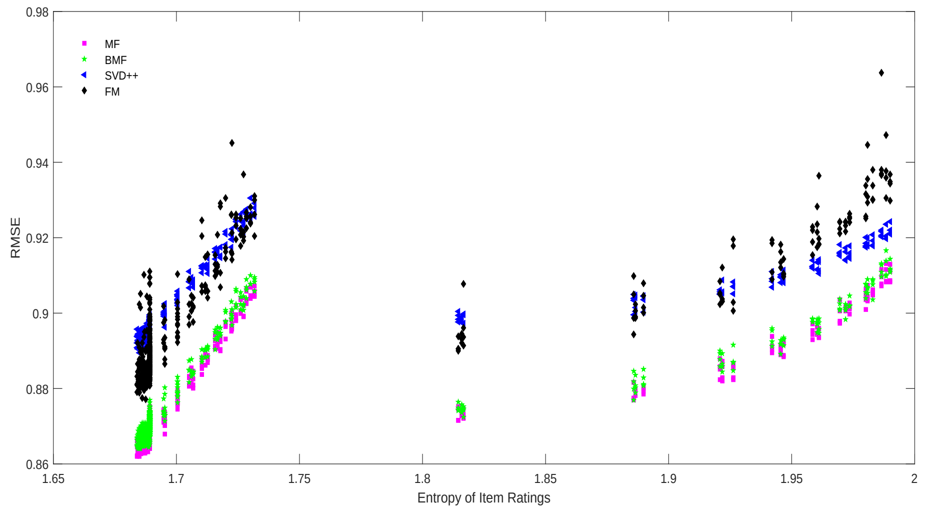


Figure A.36: Relationship between the Accuracy of Algorithms and Entropy of Item Ratings on Movielens 1M Under All Network Modification Algorithms

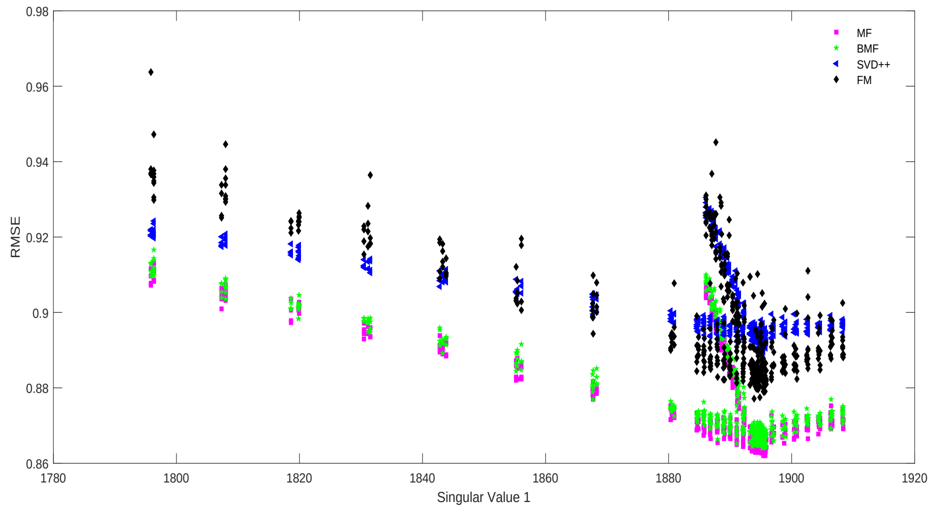


Figure A.37: Relationship between the Accuracy of Algorithms and Singular Value 1 on Movielens 1M Under All Network Modification Algorithms

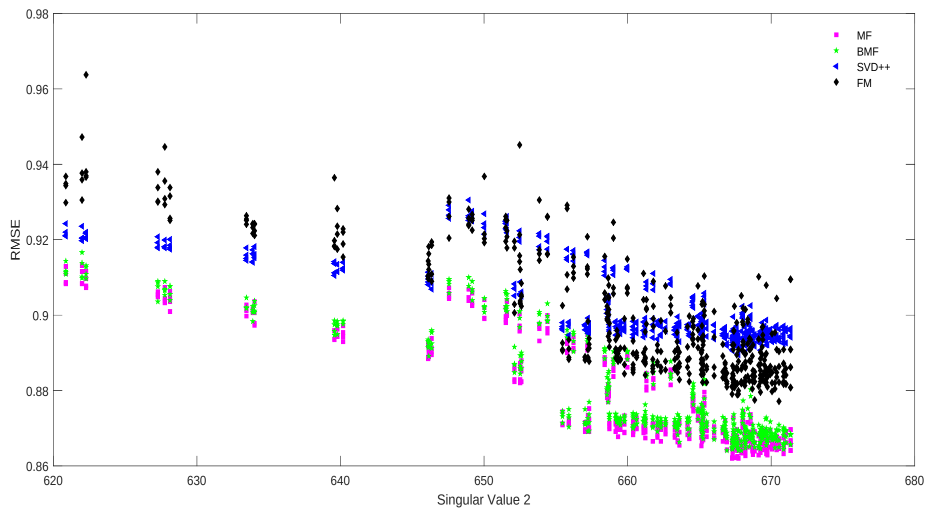


Figure A.38: Relationship between the Accuracy of Algorithms and Singular Value 2 on Movielens 1M Under All Network Modification Algorithms

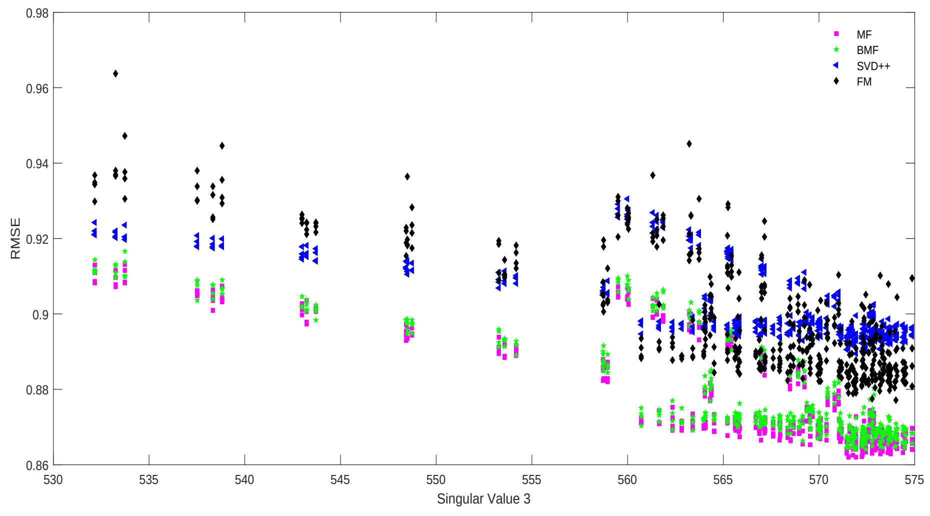


Figure A.39: Relationship between the Accuracy of Algorithms and Singular Value 3 on Movielens 1M Under All Network Modification Algorithms

Table A.3: Spearman correlation coefficients between the Accuracy of Algorithms and U-I Matrix Properties

Algorithms	Ratings STD	Degrees STD	Entropy	S1	S2	S3
MF	0.8991	-0.4909	0.8991	-0.6556	-0.8549	-0.822
BMF	0.8921	-0.4892	0.892	-0.665	-0.8494	-0.8171
SVD ++	0.8732	-0.4374	0.8734	-0.6508	-0.7725	-0.7305
FM	0.8142	-0.5038	0.8137	-0.601	-0.7449	-0.7134

Bibliography

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, June 2005. ISSN 1041-4347. doi: 10.1109/TKDE.2005.99.

[2] Gediminas Adomavicius and Jingjing Zhang. Impact of data characteristics on recommender systems performance. *ACM Trans. Manage. Inf. Syst.*, 3(1):3:1–3:17, April 2012. ISSN 2158-656X. doi: 10.1145/2151163.2151166. URL http://doi.acm.org/10.1145/2151163.2151166.

[3] Charu C. Aggarwal. *Recommender Systems: The Textbook*. Springer Publishing Company, Incorporated, 1st edition, 2016. ISBN 3319296574, 9783319296579.

[4] F.K. Bell. A note on the irregularity of graphs. *Linear Algebra and its Applications*, 161:45 – 54, 1992. ISSN 0024-3795. doi: http://dx.doi.org/10.1016/0024-3795(92)90004-T. URL http://www.sciencedirect.com/science/article/pii/002437959290004T.

[5] Dheeraj Bokde, Sheetal Girase, and Debajyoti Mukhopadhyay. Matrix factorization model in collaborative filtering algorithms: A survey. *Procedia Computer Science*, 49:136 – 146, 2015. ISSN 1877-0509. doi: http://dx.doi.org/10.1016/j.procs.2015.04.237. URL http://www.sciencedirect.com/science/article/pii/S1877050915007462. Proceedings of 4th International Conference on Advances in Computing, Communication and Control (ICAC3'15).

[6] Robin Burke, Alexander Felfernig, and Mehmet H Göker. Recommender systems: An overview. *Ai Magazine*, 32(3):13–18, 2011.

[7] Guido Jan de Nooij. Recommender systems: An overview. 2008.

[8] S. Dhuli and Y. N. Singh. Exact Analysis of Synchronizability for Complex Networks using Regular Graphs. *ArXiv e-prints*, November 2014.

[9] Ramesh Dommeti. Neighborhood based methods for collaborative filtering. *A Case Study, I*, pages 1–5, 2009.

[10] Michael D. Ekstrand, John T. Riedl, and Joseph A. Konstan. Collaborative filtering recommender systems. *Found. Trends Hum.-Comput. Interact.*, 4(2):81–173, February 2011. ISSN 1551-3955. doi: 10.1561/1100000009. URL http://dx.doi.org/10.1561/1100000009.

[11] Christoph Freudenthaler, Lars Schmidt-Thieme, and Steffen Rendle. Factorization machines factorized polynomial regression models. 2011.

[12] Silvia Gago. Spectral techniques in complex networks. *Selectec Topics on Applications of Graph Spectra, Matematicki Institut SANU, Beograd*, 14(22):63–84, 2011.

[13] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. MyMediaLite: A free recommender system library. In *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys 2011)*, 2011.

[14] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Mymedialite: A free recommender system library. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 305–308, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0683-6. doi: 10.1145/2043932.2043989. URL http://doi.acm.org/10.1145/2043932.2043989.

[15] G. Guo, J. Zhang, and N. Yorke-Smith. A novel bayesian similarity measure for recommender systems. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2619–2625, 2013.

[16] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, December 2015. ISSN 2160-6455. doi: 10.1145/2827872. URL http://doi.acm.org/10.1145/2827872.

[17] Jianming He and Wesley W. Chu. *A Social Network-Based Recommender System (SNRS)*, pages 47–74. Springer US, Boston, MA, 2010. ISBN 978-1-4419-6287-4. doi: 10.1007/978-1-4419-6287-4_4. URL https://doi.org/10.1007/978-1-4419-6287-4_4.

[18] Zhicheng He, Jie Liu, Caihua Liu, Yuan Wang, Airu Yin, and Yalou Huang. *Dropout Non-negative Matrix Factorization for Independent Feature Learning*, pages 201–212. Springer International Publishing, Cham, 2016. ISBN 978-3-319-50496-4. doi: 10.1007/978-3-319-50496-4_17. URL http://dx.doi.org/10.1007/978-3-319-50496-4_17.

[19] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, CSCW '00, pages 241–250, New York, NY, USA, 2000. ACM. ISBN 1-58113-222-0. doi: 10.1145/358916.358995. URL http://doi.acm.org/10.1145/358916.358995.

[20] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 1 2004. ISSN 1046-8188. doi: 10.1145/963770.963772.

[21] F.O. Isinkaye, Y.O. Folajimi, and B.A. Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261 – 273, 2015. ISSN 1110-8665. doi: https://doi.org/10.1016/j.eij.2015.06.005. URL http://www.sciencedirect.com/science/article/pii/S1110866515000341.

[22] Rinku Jacob, K. P. Harikrishnan, R. Misra, and G. Ambika. Measure for degree heterogeneity in complex networks and its application to recurrence network analysis. *Open Science*, 4(1), 2017. doi: 10.1098/rsos.160757. URL http://rsos.royalsocietypublishing.org/content/4/1/160757.

[23] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, October 2002. ISSN 1046-8188. doi: 10.1145/582415.582418. URL http://doi.acm.org/10.1145/582415.582418.

[24] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009. ISSN 0018-9162. doi: 10.1109/MC.2009.263. URL http://dx.doi.org/10.1109/MC.2009.263.

[25] Hyeong-Joon Kwon and Kwang-Seok Hong. Personalized real-time location-tagged contents recommender system based on mobile social networks. In *2012 IEEE International Conference on Consumer Electronics (ICCE)*, pages 558–559, Jan 2012. doi: 10.1109/ICCE.2012.6161972.

[26] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[27] Joonseok Lee, Mingxuan Sun, and Guy Lebanon. A comparative study of collaborative filtering algorithms. *CoRR*, abs/1205.3193, 2012. URL http://arxiv.org/abs/1205.3193.

[28] Cong Li, Huijuan Wang, and Piet Van Mieghem. Degree and principal eigenvectors in complex networks. In *International Conference on Research in Networking*, pages 149–160. Springer, 2012.

[29] Babak Loni and Alan Said. Wraprec: An easy extension of recommender system libraries. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 377–378, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2668-1. doi: 10.1145/2645710.2645717. URL http://doi.acm.org/10.1145/2645710.2645717.

[30] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. *Content-based Recommender Systems: State of the Art and Trends*, pages 73–105. Springer US, Boston, MA, 2011. ISBN 978-0-387-85820-3. doi: 10.1007/978-0-387-85820-3_3. URL http://dx.doi.org/10.1007/978-0-387-85820-3_3.

[31] Koji Miyahara and Michael J. Pazzani. Collaborative filtering with the simple bayesian classifier. In *Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence*, PRICAI'00, pages 679–689, Berlin, Heidelberg, 2000. Springer-Verlag. ISBN 3-540-67925-1. URL http://dl.acm.org/citation.cfm?id=1764967.1765055.

[32] Ian Soboroff. Charles Nicholas and Charles K. Nicholas. Combining content and collaboration in text filtering. In *In Proceedings of the IJCAI'99 Workshop on Machine Learning for Information Filtering*, pages 86–91, 1999.

[33] Rogier Noldus and Piet Van Mieghem. Assortativity in complex networks. *Journal of Complex Networks*, page cnv005, 2015.

[34] Yoon-Joo Park and Alexander Tuzhilin. The long tail of recommender systems and how to leverage it. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 11–18, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-093-7. doi: 10.1145/1454008.1454012. URL http://doi.acm.org/10.1145/1454008.1454012.

[35] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. 2007.

[36] Štefan Pero and Tomáš Horváth. *Comparison of Collaborative-Filtering Techniques for Small-Scale Student Performance Prediction Task*, pages 111–116. Springer International Publishing, Cham, 2015. ISBN 978-3-319-06773-5. doi: 10.1007/978-3-319-06773-5_16. URL http://dx.doi.org/10.1007/978-3-319-06773-5_16.

[37] MÁRTON PÓSFAI, GABRIELE MUSELLA, MAURO MARTINO, ROBERTA SINATRA, AMAL HUSSEINI, and PHILIPP HOEVEL. Network science.

[38] A. Ramachandra Rao, Rabindranath Jana, and Suraj Bandyopadhyay. A markov chain monte carlo method for generating random $(0,1)$-matrices with given marginals. *Sankhyā, Series A*, 58: 225–242, 1996.

[39] S. Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000, Dec 2010. doi: 10.1109/ICDM.2010.127.

[40] Steffen Rendle. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, May 2012. ISSN 2157-6904.

[41] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM. ISBN 0-89791-689-1. doi: 10.1145/192844.192905. URL http://doi.acm.org/10.1145/192844.192905.

[42] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender Systems Handbook*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010. ISBN 0387858199, 9780387858197.

[43] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 1257–1264, USA, 2007. Curran Associates Inc. ISBN 978-1-60560-352-0. URL http://dl.acm.org/citation.cfm?id=2981562.2981720.

[44] Claude Sammut and Geoffrey I. Webb, editors. *Latent Factor Models and Matrix Factorizations*, pages 571–571. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_887. URL https://doi.org/10.1007/978-0-387-30164-8_887.

[45] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM. ISBN 1-58113-348-0. doi: 10.1145/371920.372071. URL http://doi.acm.org/10.1145/371920.372071.

[46] Guy Shani and Asela Gunawardana. Evaluating recommender systems. Technical report, November 2009. URL https://www.microsoft.com/en-us/research/publication/evaluating-recommender-syste

[47] Z. Sharifi, M. Rezghi, and M. Nasiri. A new algorithm for solving data sparsity problem based-on non negative matrix factorization in recommender systems. In *2014 4th International Conference on Computer and Knowledge Engineering (ICCKE)*, pages 56–61, Oct 2014. doi: 10.1109/ICCKE.2014.6993356.

[48] Yue Shi, Martha Larson, and Alan Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Comput. Surv.*, 47(1):3:1–3:45, May 2014. ISSN 0360-0300. doi: 10.1145/2556270. URL http://doi.acm.org/10.1145/2556270.

[49] Ya-Yueh Shih and Duen-Ren Liu. Hybrid recommendation approaches: Collaborative filtering via valuable content information. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, pages 217b–217b, Jan 2005. doi: 10.1109/HICSS.2005.302.

[50] Tom A.B Snijders. The degree variance: An index of graph heterogeneity. *Social Networks*, 3(3): 163 – 174, 1981. ISSN 0378-8733. doi: http://dx.doi.org/10.1016/0378-8733(81)90014-9. URL http://www.sciencedirect.com/science/article/pii/0378873381900149.

[51] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2, January 2009. ISSN 1687-7470. doi: 10.1155/2009/421425. URL http://dx.doi.org/10.1155/2009/421425.

[52] Monic Sun. How does the variance of product ratings matter? *Manage. Sci.*, 58(4):696–707, April 2012. ISSN 0025-1909. doi: 10.1287/mnsc.1110.1458. URL http://dx.doi.org/10.1287/mnsc.1110.1458.

[53] A. Syvajarvi. *Data Mining in Public and Private Sectors: Organizational and Government Applications: Organizational and Government Applications*. Advances in Data Mining and Database Management:. Information Science Reference, 2010. ISBN 9781605669076. URL https://books.google.nl/books?id=sI2fcLf_rV8C.

[54] P. Van Mieghem, H. Wang, X. Ge, S. Tang, and F. A. Kuipers. Influence of assortativity and degree-preserving rewiring on the spectra of networks. *The European Physical Journal B*, 76(4):643–652, 2010. ISSN 1434-6036. doi: 10.1140/epjb/e2010-00219-x. URL http://dx.doi.org/10.1140/epjb/e2010-00219-x.

[55] Yining Wang, Liwei Wang, Yuanzhi Li, and Wei Chen. A theoretical analysis of ndcg ranking measures. 2013.

[56] Wikipedia. Degree-preserving randomization — wikipedia, the free encyclopedia, 2017. URL https://en.wikipedia.org/w/index.php?title=Degree-preserving_randomization&oldid=759661863. [Online; accessed 14-June-2017].

[57] Wikipedia. Music genome project — wikipedia, the free encyclopedia, 2017. URL https://en.wikipedia.org/w/index.php?title=Music_Genome_Project&oldid=779229313. [Online; accessed 18-May-2017].