Trust decomposition with classification in probabilistic matrix factorization for recommender systems

Yan, Yang

2019

Yan, Y. (2019). Trust decomposition with classification in probabilistic matrix factorization for recommender systems. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/103417>

<https://doi.org/10.32657/10220/48074>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

**Trust Decomposition with Classification in
Probabilistic Matrix Factorization for
Recommender Systems**

YAN YANG

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

2019

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

**Trust Decomposition with Classification in
Probabilistic Matrix Factorization for
Recommender Systems**

YAN YANG

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

**A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Master of Engineering**

2019

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

11-April-2019

.....
Date

Yan Yang

.....
Yan, Yang

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

11-April-2019

.....
Date

sinno pan

.....
Sinno Jialin Pan

Authorship Attribution Statement

This thesis **does not** contain any materials from papers published in peer-reviewed journals or from papers accepted at conferences in which I am listed as an author.

11-April-2019

.....

Date

Yan Yang

.....

Yan, Yang

Acknowledgments

To begin with, I would like to express my sincere thanks and great gratitude to my supervisor Professor Sinno Jialin Pan and Professor Zhang, Jie for giving me guidance and advice during these years.

January 2019

Table of contents

Statement of Originality	iii
Supervisor Declaration Statement	iv
Acknowledgments	vi
Abstract	viii
List of Figures.....	ix
List of Tables	x
Chapter 1	1
Introduction	1
1.1 Background.....	1
1.2 Motivation	7
1.3 Literature Review	8
1.4 Organization of the Report	14
Chapter 2	15
Preliminaries.....	15
2.1 Probabilistic Matrix Factorization (PMF).....	15
2.2 Support Vector Regression (SVR)	17
2.3 Stochastic Gradient Descent (SGD)	22
Chapter 3	25
Methodology and Implementation.....	25
3.1 Trust Analysis	25
3.2 Problem Definition	28
3.3 Trust Model	29
3.3.1 Making Trust Classification.....	29
3.3.2 Trust Prediction.....	32
3.4 Trust Decomposed Matrix Factorization	33
Chapter 4	38
Experimental Study	38
4.1 Benchmark Approach	38
4.2 Evaluation Metrics.....	39
4.3 Model Comparison	41
4.4 Model Effectiveness and Sensitivity.....	43
Chapter 5	45
Conclusion.....	45
Future Work.....	46
References	47

Abstract

Trust has become more and more effective in the recommender system, which complements rating-based similarity to help to improve the final performance of rating prediction. However, trust cannot represent everything, e.g., trust may not prove that one will share the same preference on items. In my study, I focus on the trust decomposition in different specific classes of items, i.e., the action movie, horror movie, adventure movie and so on. Then, I will use the support vector regression method to combine all the trust aspects of the model to predict the latent trust value. Finally, I adopt them into the Probabilistic matrix factorization model for rating prediction in recommender systems. What's more, the experiments on Epinions, Ciao, Douban, FilmTrust four datasets show there is an improvement of the performance of my model

List of Figures

Fig 2. 1	<i>The soft margin loss setting for a linear SVR</i>	19
Fig 2. 2	<i>Left represents the process of GD and right represents the process of SGD</i>	24
Fig 3. 1	<i>(a)The distribution of the ratio of users who makes the rating with the numbers of items rated. (b)The distribution of correlation between users' ratings and their trust neighbors.</i>	27
Fig 3. 2	<i>Tags of Ciao dataset and the relation between itemID and movie_categoryID</i>	30
Fig 3. 3	<i>Equal division of the items</i>	31
Fig 3. 4	<i>Mix-equal division of the items</i>	31
Fig 3. 5	<i>The left panel shows the graphical model for Probabilistic Matrix Factorization (PMF) and the right panel shows the proposed model.</i>	34
Fig 3. 6	<i>The process of prediction in the trust-aware recommender system (PMF)</i>	35
Fig 4. 1	<i>Original logistic function $g(x) = 1/(1 + e^{-x})$</i>	40
Fig 4. 2	<i>Revised logistic function (used in my model) $g(x) = 2/(1 + e^{-x}) - 1$</i>	40
Fig 4. 3	<i>The effectiveness of SVR methods used in my model on MSE and MAE with $K=5$ and 10.</i>	44
Fig 4. 4	<i>Performance with different λ with $K=10$.</i>	44

List of Tables

Table. 1 <i>User-Item rating matrix</i>	2
Table. 2 <i>Statistics of the four datasets</i>	26
Table. 3 <i>Parameters settings in methods</i>	38
Table. 4 <i>Performance comparison of different methods with $K=5$</i>	41
Table. 5 <i>Performance comparison of different methods with $K=10$</i>	41

Chapter 1

Introduction

1.1 Background

Recommender systems are extensively applied in our life. For example, when I go out to market for shopping, it is always unavoidable to encounter a salesperson who comes close to you asking some questions and making recommendations of some products expected to satisfy your potential needs. In addition, as the society and technology advance, people start shifting to online shopping. When visiting Amazon, the website will offer me some advice and recommendations for some products to choose from based on my history of purchases. This would give me a comfortable feeling if the recommendations happen to be what I have intention to buy. In addition, it makes me feel like I am offered service by a real person.

Meanwhile, this gives rise to the research into recommender system. The pattern of prior experience can be learnt for application in the recommender systems. In recent years, it is involved with deep learning for further study.

In a recommender system, I need to deal with a wide variety of essential elements such as users, items, ratings, times and the likes. However, users, items and ratings are classified as fundamental elements: I need to analyze their ratings that indicate the willingness/feedback from users to items. Subsequently, I am capable of making recommendations of the specific items to the corresponding users. Nevertheless, in practice, my objective is to make predictions for the ratings the users are expected to give concerning “new” items. For instance, the cinema could get various data from the history of my purchase of movie tickets. The information would allow it to perform an analysis of the pattern of my spending in order to better understand my preference in

watching movies. For example, I like watching horror movies more than the others. Subsequently, the cinema will predict a value of rating given by me on the latest horror film. Based on the value, they will be able to decide whether to send an email or offer a coupon in a bid to induce me to purchase the ticket.

There are a number of commonly known types of recommender systems available:

Collaborative Filtering

The simplest *CF* to make rating predictions is to take advantage of similarity. When calculating the similarity among different users, a set of ratings given by the users who have high similarity will be selected and factored into the model for calculation of the final ratings. The user-item matrix is indicated in Table. 1.1 and the value 1 represents the user ratings on the item, value 0 otherwise.

Table. 1 *User-Item rating matrix*

User u / Item x	x_1	x_2	x_3	x_4	x_5	x_6	x_7
u_1	0	1	1	1	1	?	1
u_2	1	0	1	1	1	?	?
u_3	?	1	1	1	?	1	?
u_4	?	1	1	1	1	?	?
u_5	1	1	?	?	1	1	1
u_6	?	1	1	1	?	?	?

There are two techniques employed most frequently in the systems.

1) Memory-based Collaborative Filtering Technique:

The user has some neighbors, for which the analysis of the user's neighbors who may have similar preferences is conducive to leading the user to make similar ratings to them. When targeting the neighbors. There are many algorithms to integrate the preferences and neighbors into the recommender system. For example, a k-Nearest

neighbor algorithm is typical of employing this sort of technique.

a) User-based Filtering Technique

The similarity is determined by the users' ratings given on the same item. The similarity can be calculated in two different methods.

The first one is given as $w_{u,t}$:

$$w_{u,t} = \frac{\sum_{x \in I} (r_{t,x} - \bar{r}_t)(r_{u,x} - \bar{r}_u)}{\sqrt{\sum_{x \in I} (r_{t,x} - \bar{r}_t)^2 \sum_{x \in I} (r_{u,x} - \bar{r}_u)^2}}.$$

In the formula, $w_{u,t}$ represents the similarity between user u and t . $r_{t,x}$ denotes the rating given by user t on item x , and $\bar{r}_t$ indicates the average on all ratings given by user t . I is referred to as the user set which is comprised of the users who give ratings on the same item. Therefore, the final predicted formula is defined as:

$$p_{t,x} = \bar{r}_t + \frac{\sum_{x \in S} (r_{u,x} - \bar{r}_u) \times w_{t,u}}{\sum_{x \in S} w_{t,u}}.$$

Where the $p_{t,x}$ is referred to as the predicted rating value given by user t on item x .

The second similarity is indicated as cosine function:

$$w_{t,u} = \cos(\vec{r}_u, \vec{r}_t) = \frac{\vec{r}_u \cdot \vec{r}_t}{\|\vec{r}_u\|_2 \cdot \|\vec{r}_t\|_2} = \frac{\sum_{x=1}^n r_{t,x} \cdot r_{u,x}}{\sqrt{\sum_{x=1}^n r_{t,x}^2 \cdot \sum_{x=1}^n r_{u,x}^2}}.$$

b) Item-based Filtering Technique

Distinct from a), it makes the prediction with respect to the sets of items that are similar to the item rated by the user. Therefore, the similarity between the item pairs x and y will be indicated as $w_{x,y}$:

$$w_{x,y} = \frac{\sum_{u \in R} (r_{u,x} - \bar{r}_x)(r_{u,y} - \bar{r}_y)}{\sqrt{\sum_{u \in R} (r_{u,x} - \bar{r}_x)^2 \sum_{u \in R} (r_{u,y} - \bar{r}_y)^2}}.$$

Where R is defined as the user sets who rates on both items x and y . The final

predicted function is written as:

$$p_{t,x} = \frac{\sum_{y \in S} r_{t,y} \cdot w_{x,y}}{\sum_{y \in S} w_{x,y}}.$$

2) *Model-based Collaborative Filtering*

Based on the basic *CF*, it performs further studies on the rating matrix with data mining techniques and machine learning approaches like *Clustering*, *Bayesian classifier*, *ANN* and so on, all of which are capable of improving the final performance in making predictions. A latent factor model represents a well-recognized model based on *CF*. Matrix decomposition technique is crucial to the latent factor model which is aimed at determining an approximate low-rank user latent matrix U' and item latent matrix I' and their production is treated as the original matrix. The model assumes that the similarity between users and items can be discovered in the low-rank matrix U' and I' which can represent their characters well. Distinct from the original matrix R , the dimensions of U' and I' are k , which can be pre-determined. In matrix U' , I can regard every single user u as close associate with a corresponding vector $p_u \in \mathbb{R}^k$ in U' , which p_u demonstrate the preference of the user for different items. In contrast, the vector $q_i \in \mathbb{R}^k$ in I' indicates the matching degree of the item with the latent factors, either positive or negative. Correspondingly, the dot product $q_i^T \cdot p_u$ combines all of the user factors with positive or negative and establish the final evaluation on the item.

There are a range of different methods for mining the latent matrix U' and I' , which presents the major challenge in this model regarding how to get well-performed U' and I' . For instance, *Singular Value Decomposition (SVD)* is applied extensively.

As compared to the other *CF* models, *SVD* weighs up between prediction errors and the latent vectors. Therefore, it measures the latent vector p_u^T and q_i while bringing the following regularized square error to a minimum level:

$$\min_{p,q} \sum_{u',i' \in K} (r_{u',i'} - q_{i'}^T p_{u'})^2 + \lambda (\|q_{i'}\|^2 + \|p_{u'}\|^2).$$

3) *Content-based Filtering*

Content-based filtering involves the additional information received from the contents left by users. More specifically, it is capable of performing analysis of the contexts given by users to identify the characteristics of users on items which have been either noticed or purchased by them. For example, it will extract the keywords containing my review in order to identify my attitude toward the items, either positive or negative. Text documents are significant resources which are utilized extensively with content-based filtering. The fundamental method for term analysis is extracting the single word one by one from the documents. Besides, *vector space model* and *latent semantic indexing* are two primary approaches applied in different learning techniques to represent the documents.

Vector Space Model

Vector space model is the representation of the methods using terms as a basis for learning components. In the model, document D is treated as an m-dimensional vector, where each dimension corresponds to a separate term and m is the total numbers of terms used in the collection of documents. D is proposed as:

$$w_x = tf_x \times \log\left(\frac{n}{df_x}\right) .$$

In the formula, w_x represents the weights of term t_x . f_x and df_x indicate the number of occurrences of term t_x in document D and the number of documents in which term t_x appears at least once. n is the total number of documents. If the term t_x does not appear in document D, the value of w_x would be zero. This is also called *tf – idf* method. Therefore, the user's profile can be regarded as the document D. Also, the similarity among users will be handled with cosine similarity.

There are various of common problems of the recommender systems:

1) *Data Sparsity*

In majority of datasets, the users tend to have a few ratings on items which will cause the user-item rating matrix to be very sparse. This may further lead to the difficulties in identifying the accurate similarity among the users or other similarities. Hence, it will influence the outcome of the prediction made by the recommender system.

2) *Cold Start Problem*

This reveals that the loss of the information in the datasets would result in the circumstance where I generate a model and calculate the parameters with less accuracy. For instance, if there is a user t who did not express any ratings on the items, I am thus unable to receive any useful information from the user, which hampers our ability to recommend new items to the user. This is considered as the major problem as it will reduce the final accuracy in making predictions.

3) *Scalability*

It is not likely concerning the questions as mentioned before. It is the question raised with the capability of the datasets. When the datasets grow, I will encounter the need to seek a much easier computational method to cope with heavy computation. *SVD* presents an excellent way to reduce the dimensionality which could remedy scalability and accelerate the process of computation.

1.2 Motivation

Fang et al. raise a latent factor model that is capable of identifying more useful aspects of trust for a recommender system to facilitate establishing the relationship between trust and user latent preference. They decompose the trust value into four pre-determined aspects to acquire more information about the ratings. Inspired by this, I believe in myself for having the ability to decompose the trust by applying the other method – standard classification on items.

There are a number of reasons for one user to trust another one. For example, I have trust in my friend who is deemed more dependable as an expert in film review. So, when he gives a movie high rating, I would be interested in watching it. In another scenario, I have trust in my best friend “A”, meanwhile, I also trust a friend “B” who knows movies better. When I want to watch a movie but have no idea which movie I shall choose to watch, the question would arise as to whose advice should I take and whose experience will influence me more? That is the point. In this case, I tend to focus more on the trust aspects of the explicit classification. There are various types of movies, like Action, Adventure, War, Comedy and so on. When I decompose trust in this way, it is explicit for me to combine it with the ratings in the recommender system.

1.3 Literature Review

Trust-aware recommender systems are a focus of widespread studies in recent years. It provides alternative trust information other than item ratings, which could assist in improving the performance with another view of user preference. Trust can be split into multiple aspects.

Mayer et al. (1995) indicated there are 3 trust aspects between trustors (the one trust the others) and trustees (the one be trusted). The three aspects are ability (competence), benevolence and integrity, which has the primary influence on trust. Benevolence represents the characteristics that trustee's ratings influence trustor's actions, which means if the value of Benevolence is high, the preference of trustor is more similar with the trustee. Integrity considers the relation between the trustee and all the users. Regardless of the trustor, depends on the experience, Integrity is calculated by the similarity between the preference of trustee and the average. Competence is a potential relationship between trustor and trustee. If the relation is close, the trustor can make a choice more like the trustee.

Then, Mcknight and Chervany (2002) tried to add one aspect of trust – predictability to enhance the previous model. Predictability is an indicator to make the trustee's decision and trustor's purchase history consistently. Because in real life, the trustee may not always keep pace with the trustor. Fang, Bao, and Zhang (2014) focus more on the four aspects of social trust and use the *SVR* method to make it into a new rating matrix decomposed model.

There are two primary methods to utilize trust information in recommender systems. Trust-related neighborhood models are widely applied to deal with the recommender systems.

Massa and Avesani (2007) came up with a trust-aware recommender system, which is designed to use trust relation to replace the user similarity to reduce the influence of rates made by “malicious” users. They also promoted trust across the entire trust network. Besides, they estimated a trust weight which is purposed for use in place

similarity weight. The process can be divided into two phrases. Firstly, it attempts to identify the user's neighbors. Subsequently, it makes predictions of the final ratings depending on a weighted sum of item ratings made by neighbors. The weight will be calculated from the perspective of user similarity or the trust metric. This model is useful in addressing the cold start problem with low users rating for items.

Jamali and Ester (2009) put forward a random walk method (*TrustWalker*) which is designed to combine trust-based and item-based recommendation. It uses the ratings made on similar items to the target items in order to improve the performance. *TrustWalker* is also capable of providing the confidence in their prediction, which is what the previous recommender systems fail to do.

Hao Ma, Michael R. Lyu et al. (2009) conducted research into how to ensure an improve the effectiveness and efficiency by incorporating the trust and distrust information into the recommender systems. Not only the trust information is involved in recommender systems but also the distrust information is applied because they regard the distrust information is same significant as trust. Therefore, the trust and distrust relations are regarded as regularization terms and adopted into the objective function. Utilize the gradient descent method on the objective function, it is possible for us to learn the latent low-dimensional user-specific and item-specific matrices for predicting the user's preferences on different items.

Guo et al. (2012) proposed a method used to integrate the neighbors' ratings in recommender systems. They referenced the average ratings of the user's neighbors whom they trust in the final target item rating. It improves the performance exhibited by the recommender systems.

However, there are some drawbacks with these approaches as well. The assumption involved is that if the user trusts someone, he/she would like to share the similar user preference with his/her trustee. Nevertheless, in reality, it does not necessarily work this way. Then, they use the trust information in recommender systems directly, and they do not make the implicit relationship known between trust information and rating

information. Finally, when they use trust information to replace the other things to improve the performance, there remains the need for them to take the noise of trust networks into consideration, such as the mislabeling of trust and the label of specific trust value.

The other dominant type is trust related latent factor models.

Ma et al. (2008) suggested a method which involves social information to enhance the performance in making predictions. *SoRec (Social Regularization method)* combines the social network with a probabilistic factor model. Specifically, when being allowed the latent user-feature space, they tend to share the same space in decomposing the user-user trust matrix. It improves the performance more than the basic matrix factorization model. However, there remains a problem that it is difficult to cope with the low levels of interpretability.

Ma, King, and Lyu (2009) recommended a social trust ensemble method (*RSTE*). Depending on the *SoRec*, *RSTE* has 3 assumptions added on. Users have diversified preference on different items. Users can be influenced by the social network easily, and they are willing to take the advice offered by their friends. Finally, the decision made by the user is combined with their personal preference and the advice offered by their friends. *RSTE* fuses user-item rating matrix and user-user trust matrix linearly to make an improvement on accuracy.

Jamali and Ester (2010) introduced the novel *SocialMF* model with the basis of *RSTE*. *SocialMF* makes the trust propagate in the network to fill up the blank of trust network and establishes a new link among users, which makes the user latent preference vector resemble the vectors of their friends more. As a user's immediate neighbors could exert direct influence on the user's social behavior. The feature vector of each direct neighbor is reliant on the feature vector of his direct neighbors. Based on this assumption, it is effortless to determine the user feature vectors even if the user did not express any ratings in the past. Therefore, it reduces the likelihood of recommender errors significantly for cold start.

Yang, Lei, and Liu (2013) came up with another model *TrustMF*. They indicated that not only will the user's decisions be affected by the previous ratings, reviews, and their friends, but also it will affect the decisions made by the users who trust him/her. *TrustMF* maps users into two spaces, trustor and trustee spaces, with different meanings. The vector in the trustor space indicates the trusting in the ratings, reviews and the other vectors which have the influence to let other users trusted by himself/herself. Then, it combines user space and item space to factor in the user-item ratings to improve the accuracy in predictions.

Tong Zhao et al. (2013) investigated the potential association between the tags of items and trust between users. They put forward *topic-specific trust-based matrix factorization (TTMF)* that is based on *probabilistic matrix factorization (PMF)*. When discovering and understanding the potential properties of the users purchased items, it is easier to determine the users' various interests and distinguish the trust into different interests. Then, *TTMF* mine topics from tags of the items and make an estimation of topic-specific trust relations between users simultaneously. Using this topic-specific trust relation could improve the recommendation accuracy and address the issue with the cold start for the item.

W. Yao et al. (2014) proposed a model *RoRec*, which learns dual role preferences for trust-aware recommendation by establishing the model of specific interactions and implicit interactions among users. Users in trust rating networks are associated with two different roles at the same time. They are trustor and trustee. "Trustor" is the role who trusts others and "Trustee" is the role who is trusted by others. Trustor may be affected by the ratings or reviews made by the users he/she trusts. Likewise, trustee's actions such as ratings and reviews will also impress others who trust him/her. However, the preferences of the two roles are possible to be different from each other. For instance, James is my best friend who is a fan of comic but not familiar with opera. He is trustworthy in moral quality and the knowledge of the comic. However, I am crazy about opera. Therefore, for me, he is not that trustworthy in the field of opera

and his ratings or reviews on opera do not have an influence on me.

Haifeng Liu et al. (2015) put forward a collaborative filtering method *CF-TC*. In trust-based recommender systems, trustor and trustee are significant roles. Among most of the trust-based methods, the explicit links between the trustor and trustee are used to target similar neighbors in order to make recommendations. However, there are a number of implicit links between users which cannot be acquired directly. The recommended *CF-TC* method includes two primary components: 1) It mines implicit links between users with the same role. 2) It carries on to apply the mined implicit links for rating prediction of the user. Firstly, for each user, a user representation is built up using those users who perform the same role. The weight of each implicit link between the same roles users can be derived by assessing cosine similarity. Secondly, based on the computed weights, two variants of *CF-TC*, Memory-based *CF-TC*, and Matrix Factorization-based *CF-TC* are presented.

Ma et al. (2011) came up with an alternative model that is recommender systems with social regularization. The biggest disparity lies in the choice to fuse social information and trust information. Distinct from the trust information, social information is considered to be most reliable in real life and could be more easily available from websites such as Facebook or others. The other problem referred to before is an assumption that if the user trusts someone, he/she will have a similar preference to his/her trustee. However, the same drawbacks remain in the neighborhoods models.

G. Guo, J. Zhang et al. (2015) put forward a novel trust-based recommendation model *TrustSVD*. This trust-based matrix factorization model is characterized by the incorporation of ratings and trust information for rating predictions. However, trust information is fairly sparse. It is necessary to complement the implicit correlations by the ratings. Therefore, placing too much focus on either one kind of information could deliver only a slight improvement in predictive accuracy. In trust networks, users have a positive correlation with their trust neighbors. Besides, in trust-alike networks, users have a weak positive correlation with their neighbors. These findings provide

motivations for consideration to be given to both explicit and implicit influence of ratings. A weighted λ - regularization approach was employed to regularize the user-specific and item-specific latent feature vectors further. Even if the users express a few ratings or nothing, the user-specific vectors can still be learned from their trust information. Hence, data sparsity and cold start issues can be better addressed.

X. Yang, H. Steck, and Y. Liu (2012) presented a novel model with the concept of “inferred circles of friends”. Instead of expanding the user’s friends in other researches, they try to determine the most ideal subset of a user’s friends in order to make recommendations for an item category of interest more precisely. Hence, these inferred circles are reliant on the various item categories. They put some algorithms forwards to infer the category circles and identify the trust value on each link with the experience the user has in each category. The process of calculating is divided into two steps. First, estimate a user’s expertise level in a specific category depending on the experience of the user and his/her trustors have in the same category. Then, assign to users trust values proportional to corresponding expertise levels.

1.4 Organization of the Report

The objective of this paper is to build a bridge between trust network and users' preferences. I put forward a model with trust latent factors which make the trust information more convenient to be applied in the recommender systems.

In section 2, I describe preliminaries composed of *Probabilistic Matrix Factorization (PMF)*, *Support Vectors Regression (SVR)* and *stochastic gradient descent (SGD)*.

In section 3, according to the previous research (McKnight and Chervany 2002) which generate well-recognized aspects of trust, I filter some of them and alter them in order to fit my model. Specifically, the chosen aspects are called original similarity and classification similarity respectively which are determined by users' ratings only. Then, combine these two aspects with user own propensity into a *Support Vector Regression* model in order to predict the explicit trust value among users. Moreover, due to the imbalanced and non-linear problems, these specific aspects are captured with a Gaussian radial basis kernel function in the *SVR* model. Finally, I utilize the predicted trust values with the latent user features similarities as new conditions and adopt them into the previous *PMF* model to enhance the precision of rating predictions.

In section 4, I compare the results of my method with other models of recommender systems, i.e. *PMF*, *SVD++*, *SocialMF*, and *TrustMF*. Then, with the experiments, I perform an analysis of different advantages and disadvantages among these models.

In section 5, I draw a conclusion of my model and raise some future works.

Chapter 2

Preliminaries

2.1 Probabilistic Matrix Factorization (PMF)

There are M movies and N users. The user set is denoted as $U = \{u_1, u_2, \dots, u_N\}$ and the item set is described as $I = \{i_1, i_2, \dots, i_M\}$. The user rates on movies from 1 to 5. PMF denotes r_{ij} as the rating value given from user u_i on item i_j . $R_{ui}^{N \times M}$ represents the rating matrix. When decomposing the user-item rating matrix R , I get the latent user feature matrix $U^{K \times N}$ and the latent movie feature matrix $V^{K \times M}$. More specific, the columns of U (U_i) and V (V_j) represent the K -dimensional user-specific and movie-specific latent feature vectors respectively. The objective is to learn the matrixes U and V from the decomposition of the R . The *root-mean-square error (RMSE)* is used to evaluate the performance of model PMF . It combines the probabilistic model with Gaussian observation and gives the conditional distribution provided from Salakhutdinov and Mnih (2007).

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

where $\mathcal{N}(R_{ij} | U_i^T V_j, \sigma_R^2)$ is the Gaussian distribution with mean μ_R and variance σ_R^2 . I_{ij}^R is the indicator function that if user u_i rates on movie i_j then the value equals 1, otherwise the value equals 0. PMF also sets the user and movie vectors with zero-mean Gaussian prior:

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

Then, PMF uses the Bayesian method to get the next formula:

$$p(U, V | R, \sigma_U^2, \sigma_V^2, \sigma_R^2) = p(U, V, R | \sigma_U^2, \sigma_V^2, \sigma_R^2) / p(R | \sigma_U^2, \sigma_V^2, \sigma_R^2)$$

Because $p(R | \sigma_U^2, \sigma_V^2, \sigma_R^2)$ is a constant. Subsequently, the formula will be revised as:

$$\begin{aligned}
p(U, V \mid R, \sigma_U^2, \sigma_V^2, \sigma_R^2) &\propto p(R \mid U, V, \sigma_U^2, \sigma_V^2, \sigma_R^2) \cdot p(U \mid \sigma_U^2, \sigma_V^2, \sigma_R^2) \cdot p(V \mid \sigma_U^2, \sigma_V^2, \sigma_R^2) \\
&= p(R \mid U, V, \sigma_R^2) \cdot p(U \mid \sigma_U^2) \cdot p(V \mid \sigma_V^2) \\
&= \prod_{i=1}^N \prod_{j=1}^M [\mathcal{N}(R_{ij} \mid U_i^T V_j, \sigma_R^2)]^{I_{ij}^R} \times \prod_{i=1}^N \mathcal{N}(U_i \mid 0, \sigma_U^2 \mathbf{I}) \times \prod_{j=1}^M \mathcal{N}(V_j \mid 0, \sigma_V^2 \mathbf{I}) \quad (3)
\end{aligned}$$

Where the density function of U is (same as V):

$$\mathcal{N}(U_i \mid 0, \sigma_U^2 \mathbf{I}) = -\frac{1}{(2\pi)^{N/2} |\sigma_U^2 \mathbf{I}|^{1/2}} \exp\left(-\frac{1}{2} U_i^T (\sigma_U^2 \mathbf{I})^{-1} U_i\right) \quad \text{and} \quad (\sigma_U^2 \mathbf{I})^{-1} = \frac{1}{\sigma_U^2} \mathbf{I}$$

Therefore, the corresponding log function is:

$$\begin{aligned}
\ln \mathcal{N}(U_i \mid 0, \sigma_U^2 \mathbf{I}) &= \ln\left(-\frac{1}{(2\pi)^{N/2} |\sigma_U^2 \mathbf{I}|^{1/2}}\right) - \frac{U_i^T U_i}{2\sigma_U^2} \\
&= -\ln(|\sigma_U^2 \mathbf{I}|^{1/2}) - \frac{U_i^T U_i}{2\sigma_U^2} + C_U \\
&= -\frac{1}{2} \ln(\sigma_U^{2N}) - \frac{U_i^T U_i}{2\sigma_U^2} + C_U \\
&= -\frac{N}{2} \ln(\sigma_U^2) - \frac{U_i^T U_i}{2\sigma_U^2} + C_U \quad (4)
\end{aligned}$$

In order to make the formula easily computable, *PMF* takes the log functions to make the formula become the linear equation:

$$\begin{aligned}
\ln p(U, V \mid R, \sigma_U^2, \sigma_V^2, \sigma_R^2) &= \sum_{i=1}^N \sum_{j=1}^M I_{ij}^R \ln \mathcal{N}(R_{ij} \mid U_i^T V_j, \sigma_R^2) \\
&\quad + \sum_{i=1}^N \ln \mathcal{N}(U_i \mid 0, \sigma_U^2 \mathbf{I}) + \sum_{j=1}^M \ln \mathcal{N}(V_j \mid 0, \sigma_V^2 \mathbf{I}) \\
\ln p(U, V \mid R, \sigma_U^2, \sigma_V^2, \sigma_R^2) &= -\frac{1}{2\sigma_R^2} \sum_1^N \sum_1^M I_{ij} (R_{ij} - U_i^T V_j)^2 \\
&\quad - \frac{1}{2\sigma_U^2} \sum_1^N U_i^T U_i - \frac{1}{2\sigma_V^2} \sum_1^M V_j^T V_j \\
&\quad - \frac{1}{2} \left(\left(\sum_{i=1}^N \sum_{j=1}^M I_{ij} \right) \ln \sigma_R^2 + NK \ln \sigma_U^2 + MK \ln \sigma_V^2 \right) + C \quad (5)
\end{aligned}$$

Where C is a constant which has no relation with the other parameters. Then, *PMF* needs to maximize the log-posterior of the user and movie features. Because there

are many negative parts and constant parts in the formula, *PMF* extracts some parts and rewrites the original objective function into new one:

$$E = \frac{1}{2} \sum_1^N \sum_1^M I_{ij} (R_{ij} - U_i^T V_j)^2 + \frac{\lambda_U}{2} \sum_1^N \|U_i\|_F^2 + \frac{\lambda_V}{2} \sum_1^M \|V_j\|_F^2 \quad (6)$$

Where $\lambda_U = \sigma_R^2 / \sigma_U^2$, $\lambda_V = \sigma_R^2 / \sigma_V^2$ and $\|M\|_F$ represents the Frobenius norm. Now, the original question turns into the new minimum question. Using the *gradient descent* method in equation (6) of U and V to identify the minimum E out with corresponding U and V .

What's more, in order to limit the range of the prediction rating values, *PMF* adopts the logistic function $g(x) = 1 / (1 + \exp(-x))$ to bound the range of the predictions. Hence, the new formulas are rewritten as:

$$p(R | U, V, \sigma_R^2) = \prod_{i=1}^N \prod_{j=1}^M [\mathcal{N}(R_{ij} | g(U_i^T V_j), \sigma_R^2)]^{I_{ij}^R} \quad (7)$$

$$E = \frac{1}{2} \sum_1^N \sum_1^M I_{ij} (R_{ij} - g(U_i^T V_j))^2 + \frac{\lambda_U}{2} \sum_1^N \|U_i\|_F^2 + \frac{\lambda_V}{2} \sum_1^M \|V_j\|_F^2 \quad (8)$$

Corresponding to the new predictions, *PMF* maps the original ratings range from $[1, 5]$ to $[0, 1]$ with the map function $m(x) = (x - 1) / 4$. Then, the range of the predictions can match the original ratings well.

2.2 Support Vector Regression (SVR)

ϵ -SV Regression (Vapnik 1995)

There is a training dataset $\{(x_1, y_1), \dots, (x_n, y_n)\} \subset \mathcal{X} \times \mathbb{R}$, where $\mathcal{X}$ is the input space. Specifically, $\mathcal{X} = \mathbb{R}^d$. For instance, the price of a stock is affected by many factors. Using the principles of economics to set the coefficients w of these factors on the i^{th} day observed data x_i , people can acquire the i^{th} day predicted stock price y_i . In the ϵ -SV regression model, it is necessary to find a function $f(x)$ to constrain that the deviation between $f(x_i)$ and the observed target y_i is less than ϵ . In other words, it is acceptable if the errors are less than ϵ , but it is rejected if the errors exceed ϵ . Back to the previous example, the operator does not care about the stock price when they

are just less than the predicted price in ε . But if the stock price difference is exceeding ε , the operator will perform some actions as soon as possible.

Back to ε -SV regression model. The form of the linear function is described as:

$$f(x) = \langle w, x \rangle + b \quad \text{with } w \in \mathcal{X}, b \in \mathbb{R} \quad (1)$$

Where $\langle \cdot, \cdot \rangle$ represents the dot product in space $\mathcal{X}$. The objective is to seek one suitable vector w to meet the flatness of the function. Generally, the frequently-used method is minimizing the norm of w ($\|w\|^2 = \langle w, w \rangle$) to meet the flatness. So, the problem will be written as follow:

$$\begin{aligned} & \text{minimize} && \frac{1}{2} \|w\|^2 \\ & \text{subject to} && \begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon \end{cases} \end{aligned} \quad (2)$$

The potential assumption is that all the (x_i, y_i) pairs' predictions are ε -deviation with real data. What's more, this problem is a convex problem.

Soft Margin Regression (Cortes and Vapnik 1995)

Furthermore, I want to make the model more flexible and error acceptable with soft margin loss function. The slack variable ξ_i and ξ_i^* are used to handle the infeasible constraints of the previous problem (2). The new form of the problem can be described as follow:

$$\begin{aligned} & \text{minimize} && \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l (\xi_i + \xi_i^*) \\ & \text{subject to} && \begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \end{aligned} \quad (3)$$

where the constant C describes the trade-off the flatness of the regression function between the total amount of deviations which is larger than ε (cannot acceptable). Figure 2.1 shows the details of the process. Also, ε -insensitive loss function is denoted as:

$$|\xi|_\varepsilon = \begin{cases} 0 & \text{if } |\xi| \leq \varepsilon \\ |\xi| - \varepsilon & \text{otherwise} \end{cases} \quad (4)$$

Fig 2. 1 The soft margin loss setting for a linear SVR

The problem has $d + 1 + 2n$ variables ($d + 1$ is the dimension of w on space $\mathcal{X}$) and $2d + 2n$ constraints. Due to the high dimension of the problem, the SVR Dual problem is proposed.

SVR Dual Problem and Quadratic Program

The dual variables and the dual form come out by constructing a Lagrange function from the objective function (3) and the corresponding constraints. (Here I call the previous objective function as prime function and the new one as dual function):

$$L = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l (\xi_i + \xi_i^*) - \sum_{i=1}^l (\eta_i \xi_i + \eta_i^* \xi_i^*) - \sum_{i=1}^l \alpha_i (\varepsilon + \xi_i - y_i + \langle w, x_i \rangle + b) - \sum_{i=1}^l \alpha_i^* (\varepsilon + \xi_i^* + y_i - \langle w, x_i \rangle - b) \quad (5)$$

where L is the Lagrangian and $\eta_i, \eta_i^*, \alpha_i, \alpha_i^*$ are Lagrange multipliers. Meanwhile, the dual variables in (5) are under the constraints shown below:

$$\alpha_i, \alpha_i^* \geq 0 \quad \eta_i, \eta_i^* \geq 0 \quad (6)$$

It is necessary to identify the saddle point by the partial derivations of function L :

$$\partial_b L = \sum_{i=1}^l (\alpha_i - \alpha_i^*) = 0 \quad (7)$$

$$\partial_w L = w - \sum_{i=1}^l (\alpha_i - \alpha_i^*) \times x_i = 0 \quad (8)$$

$$\partial_{\xi_i} L = C - \alpha_i - \eta_i = 0 \quad (9)$$

$$\partial_{\xi_i^*} L = C - \alpha_i^* - \eta_i^* = 0 \quad (10)$$

Subsequently, find the variables set (w, b, ξ_i, ξ_i^*) as the saddle point of function L and use it to solve the previous problem further. From equation (7) to (10), the transformations of them can be written as:

$$w = \sum_{i=1}^l (\alpha_i - \alpha_i^*) \times x_i \quad \& \quad \eta_i^* = C - \alpha_i^* \quad \& \quad \eta_i = C - \alpha_i$$

Replace the transformations into (5) and I can get the new dual optimization problem:

$$\begin{aligned} & \text{maximize} \begin{cases} -\frac{1}{2} \sum_{i,j=1}^l (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \langle x_i, x_j \rangle \\ -\varepsilon \sum_{i=1}^l (\alpha_i + \alpha_i^*) + \sum_{i=1}^l y_i (\alpha_i - \alpha_i^*) \end{cases} \quad (11) \\ & \text{subject to } \sum_{i=1}^l (\alpha_i - \alpha_i^*) = 0 \quad \text{and} \quad \alpha_i, \alpha_i^* \in [0, C] \end{aligned}$$

What's more, the regression function is rewritten as:

$$f(x) = \sum_{i=1}^l (\alpha_i - \alpha_i^*) \times \langle x, x_i \rangle + b \quad (12)$$

This is the SV expansion. From the equations shown above, the weight w is linear combination of observed training data x_i .

More specifically, the complexity of the function represented by SVs is only controlled by the number of the SVs, and it has no relation with the dimension of input space $\mathcal{X}$. Meanwhile, in this situation, the explicit weight w is not necessary to be calculated to get the final regression function $f(x)$.

Computing b

KKT (Karush-Kuhn-Tucker) conditions are utilized into calculating the value of b , because *KKT* conditions vanish the product between dual variables and constraints. Therefore, the parameters sets will have new constraints shown below:

$$\alpha_i (\varepsilon + \xi_i - y_i + \langle w, x_i \rangle + b) = 0 \quad \& \quad \alpha_i^* (\varepsilon + \xi_i^* + y_i - \langle w, x_i \rangle - b) = 0 \quad (13)$$

$$\eta_i \xi_i = 0 \quad \& \quad \eta_i^* \xi_i^* = 0 \quad (14)$$

Substituting (10) into (14), I can get:

$$(C - \alpha_i) \xi_i = 0 \quad \& \quad (C - \alpha_i^*) \xi_i^* = 0 \quad (15)$$

There are several conclusions found:

1) If $\alpha_i = C$ or $\alpha_i^* = C$, then the corresponding sample (x_i, y_i) is out of the ε -insensitive tube, which shown in the Fig.2.1.

2) $\alpha_i \alpha_i^* = 0$. Because $\varepsilon > 0, \xi \geq 0, \xi^* \geq 0$, then $\varepsilon + \xi_i > 0 \quad \& \quad \varepsilon + \xi_i^* > 0$. But in (13), $-y_i + \langle w, x_i \rangle + b \quad \& \quad y_i - \langle w, x_i \rangle - b$ are definitely opposite sign so that $\varepsilon + \xi_i - y_i + \langle w, x_i \rangle + b \quad \& \quad \varepsilon + \xi_i^* + y_i - \langle w, x_i \rangle - b$ cannot be zero at the same time. Finally, I get $\alpha_i \alpha_i^* = 0$.

3) If $\alpha_i < C$, then $\xi_i = 0$. With the inequation $\varepsilon + \xi_i - y_i + \langle w, x_i \rangle + b \geq 0$, I can infer $b \geq y_i - \varepsilon - \langle w, x_i \rangle$. Same as α_i^* , if $\alpha_i^* < C$, then $\xi_i^* = 0$ and $\varepsilon + \xi_i^* + y_i - \langle w, x_i \rangle - b \geq 0$. So the final inequation is $b \leq \varepsilon + y_i - \langle w, x_i \rangle$, and I infer $\min\{y_i - \varepsilon - \langle w, x_i \rangle\} \leq b \leq \max\{y_i + \varepsilon - \langle w, x_i \rangle\}$. (16)

Kernels

Nonlinearity Process:

The previous problems employ linear algorithms. Then, it is needed to adopt kernels to handle with the nonlinear algorithm. In brief, it uses $\phi(x)$ to map the training data x_i from space $\mathcal{X}$ into feature space $\mathcal{F}$, and then use the standard SV linear $x \in \mathbb{R}^2$ algorithm mentioned before to solve the problem.

Here is an example. Quadratic feature in $\mathbb{R}^2$: There is a map $\phi: \mathbb{R}^2 \rightarrow \mathbb{R}^3$ and the map is $\phi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$. x_1, x_2 are the components of the vector $x \in \mathbb{R}^2$. Moreover, training a linear SV machine on the preprocessed features may perform on one quadratic function. This example makes the approach more reliable, which can help the problems become computationally infeasible for both polynomial features of a high order and high dimensionality.

Implicit Mapping with Kernels

It is necessary to establish a cheap computational method to solve the problem. The key point is mentioned in the previous example. That is:

$$\langle (x_1^2, \sqrt{2}x_1x_2, x_2^2), (x_1'^2, \sqrt{2}x_1'x_2', x_2'^2) \rangle = \langle x, x' \rangle^2 \quad (17)$$

As a result, the complexity of the function represented by SVs is only controlled by the number of the SVs. Meanwhile, the SV algorithm is only controlled by the dot products between patterns x_i . Rather than explicitly computing the ϕ , it is better to get the kernel $k(x, x') = \langle \phi(x), \phi(x') \rangle$, which makes the computation cheaper.

Substituting the kernels into the original formula and the new form of SV optimization problem will be written as:

$$\text{maximize} \begin{cases} -\frac{1}{2} \sum_{i,j=1}^l (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(x_i, x_j) \\ -\varepsilon \sum_{i=1}^l (\alpha_i + \alpha_i^*) + \sum_{i=1}^l y_i(\alpha_i - \alpha_i^*) \end{cases} \quad (18)$$

$$\text{subject to } \sum_{i=1}^l (\alpha_i - \alpha_i^*) = 0 \quad \text{and} \quad \alpha_i, \alpha_i^* \in [0, C]$$

The expansion of weight and the final function are rewritten at the same time:

$$w = \sum_{i=1}^l (\alpha_i - \alpha_i^*) \phi(x_i) \quad (19)$$

$$f(x) = \sum_{i=1}^l (\alpha_i - \alpha_i^*) k(x, x_i) + b \quad (20)$$

The most significant distinction from linear SV algorithm is that the weight w does not need to be calculated explicitly anymore. What's more, the new flattest function corresponding with the optimization problem with the nonlinear setting is extracted from new mapping feature space.

2.3 Stochastic Gradient Descent (SGD)

The *gradient descent* (GD) is a popular optimization method in machine learning, which provides a track for the answer set to reach local or global optimization.

For example, there is a linear model:

$$h(x) = \sum_{i=0}^n \theta_i x_i = \theta^T x \quad (1)$$

where $\theta = (\theta_1, \dots, \theta_n)$ is the vector.

And the corresponding loss function is:

$$L(\theta) = \frac{1}{2} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 \quad (2)$$

Firstly, I calculate the maximum of the derivate of the loss function. Then, I choose that as the direction to make gradient descent in the model. In the process, the weights are updated in each iteration with the direction of the gradient. When the process runs n iterations, it will acquire a final optimal value and an optimization point. The process in iteration t is described as:

$$\frac{\partial}{\partial \theta_j} L(\theta) = \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_{\theta}(x) - y)^2 \quad (3)$$

$$= 2 \cdot \frac{1}{2} (h_{\theta}(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_{\theta}(x) - y)$$

$$= (h_{\theta}(x) - y) \cdot \frac{\partial}{\partial \theta_j} (\sum_{i=0}^n \theta_i x_i - y)$$

$$= (h_{\theta}(x) - y) x_j$$

$$\theta_j(new) = \theta_j - \alpha \frac{\partial}{\partial \theta_j} L(\theta) \quad (4)$$

Which can be written as:

$$\theta^{t+1} = \theta^t - \alpha \cdot \nabla_{\theta} L(\theta) \quad (5)$$

Where α represents step size which is also called the learning rate in machine learning. In the original method of *GD*, it is necessary to use all the data into updating, which leads to the process is time-consuming.

Fig.2.2 indicates the difference between basic *GD* and *SGD*. The direction of *GD* can be found clearly with one direction (blue arrow) and *SGD* has no rules to reach the final point.

Fig 2. 2 *Left represents the process of GD and right represents the process of SGD*
Different from basic *GD*, *SGD* takes one training data $(x^{(i)}, y^{(i)})$ into the updating of parameter. And the final form of the update will be described as:

$$\theta_j^{(new)} = \theta_j - \alpha(y^{(i)} - h_{\theta}(x^{(i)}))x_j^{(i)} \quad (6)$$

which also can be written as:

$$\theta^{t+1} = \theta^t - \alpha \cdot \nabla_{\theta} L(\theta; x^{(i)}, y^{(i)}) \quad (7)$$

Compared to basic *GD*, *SGD* has a fast speed to perform the process, but its disadvantage is also apparent. If the original function is a convex function, *SGD* can also achieve a global optimization value. However, if the original function is nonconvex, *SGD* will only find a local optimization value nor global one except they are same in some specific cases. Another disadvantage is that the process will be slow when the solution is near to the local optimization point because it will shock around the point with the random direction in each iteration. In order to solve this question, momentum is adopted in *SGD* to constrain the shock and speed up the time to reach the local optimization point. Specifically, in each update iteration, it will add the last iteration's gradient with weight into the new one:

$$v_t = \gamma \cdot v_{t-1} + \alpha \cdot \nabla_{\theta} L(\theta) \quad (8)$$

$$\theta_{t+1} = \theta_t - v_t \quad (9)$$

Where v_t is the gradient calculated in iteration t and γ is the value of momentum which will be set in advance.

Chapter 3

Methodology and Implementation

3.1 Trust Analysis

There are four datasets used in my experiments:

Epinions(http://www.trustlet.org/downloaded_epinions.html), FilmTrust(<https://www.librec.net/datasets/filmtrust.zip>), Douban(<https://sites.google.com/site/erhengzhong/datasets>) and Ciao (<https://www.librec.net/datasets/CiaoDVD.zip>). These four datasets not only include the ratings of items but also contain the relationships among the users. The ratings of Ciaos and Epinions are integers from 1 to 5. The ratings of Douban are real values from 0.5 to 5 with step 0.5, and the ratings of FilmTrust are also real values from 0.5 to 4 with step 0.5. In trust part, only the relationship in FilmTrust is friendship, and the other three are trust relationships. The main distinction between friend and trust relationship is that friend relationship is bidirectional, but trust relationship is unidirectional. For example, in Facebook, the relationship is bidirectional. Two users are friends, and they can be treated as trusting with each other. Nevertheless, in Twitter, the relationship is unidirectional. A user can follow many superstars, which implies the user trusts these superstars but they may not be followed and trusted by these superstars. Friendship network is often regarded as a trust network, and the trust is symmetric. The trust in the other three datasets is asymmetric. From the statistics, I can get the table 2 and some observations on that.

Observation 1. The rating matrix is so sparse that the trust information can make up the lack of ratings.

The density of trust is sparser than the rating in Epinions and FilmTrust. In Douban,

the density of trust is near the ratings, but the value is quite low. However, in Ciao, the density of trust is larger than ratings. Recently, many pieces of research combine trust information into the recommender systems to improve the performance. However, most focus too much on the explicit trust relationship nor the implicit trust influence. That is, the utility of trust is not entirely found and used. Meanwhile, the low sparsity of trust also shows there is a potential place in trust where can be exploited.

Fig.3.1 indicates that: 1) There is a large ratio of users who have not expressed any ratings in the datasets. However, they are also influenced by the social network because they may play the role of trustors or trustees in the network. These users in trust-aware recommender system can provide the inherent information to help improve the performance. 2) For cold-start users (who rates less than five items in my case) and warm-start users (who rates more than 20 items in my case), trust information help to improve the final performance very naturally.

This implies I also need to exploit the inherent aspects of the trust and put them into use for improving the performance.

Table. 2 : *Statistics of the four datasets*

Feature	Epinions	FilmTrust	Douban	Ciao
user	40,163	1,508	158897	7,375
item	139,738	2,071	478229	99,746
ratings	664,824	35,497	5199473	280,391
density	0.051%	1.14%	0.98%	0.03%
trustors	33,960	609	92367	6,792
trustees	49,288	732	92367	7,297
trust	487,183	1853	5130107	111,781
density	0.029%	0.42%	0.89%	0.23%

Fig 3. 1 (a)The distribution of the ratio of users who makes the rating with the numbers of items rated. (b)The distribution of correlation between users' ratings and their trust neighbors.

Observation 2. A user's ratings have a weak correlation with the average of his/her social neighbors' ratings in the trust-alike relationship, while a strong correlation in trust relationship.

Then, I consider the effect of putting trust information into rating prediction. The Fig.3.1(b) shows the correlation between the user's ratings and his/her social neighbors' ratings (Person Correlation Coefficient). I can find there is a weak correlation (PCC) between users' rating and their neighbors in the datasets FilmTrust and Douban. The Douban uses the friendship as the social network, and the FilmTrust applies the specific value of trust nor the binary value. On the contrary, there is a strong correlation (PCC) between users' rating and their neighbors in the datasets Epinions and Ciao. Specifically, in the range of $[0.9, 1.0]$, the percentage of Epinions is nearly 18%, and Ciao is nearly 13%. What's more, in the range of $[0.5, 1.0]$, the percentage of Epinions is nearly 55%, and Ciao is nearly 40%. From the data, I can observe, sometimes, the implicit trust can provide more accuracy than specific trust value.

The observation 2 shows that it can promote the generality of trust-based recommender systems in trust or trust-alike relationships when combining the explicit and inherent influence with trust.

Observation 3. The user shares a similar preference with the trustees.

In real life, I can find some useful information depending on my intuition directly. For example, if Tom trusts his friend James, the trust value is described with the binary value 1. If Tom does not know the actress Emma Waston who is a superstar in Movie, the trust value of Tom would be 0, which implies Tom does not trust Emma. Furthermore, the binary value only indicates the trust relationship, but they do not clarify the specific degree of the trust which shows how much the user trusts the others. So, if Tom has specific trust values on James and Emma, which imply Tom's preference in movies influenced by James and Emma. In my case, I need to predict the specific trust values between the user pairs and propagate specific trust values over the whole network.

Furthermore, in real life, the user shares similar preferences with his/her trustees, not the same preferences. For example, when Tom considers the trust on others, he may divide his trust into several parts on his trustees. More specifically, in the movie recommender systems, there are several classifications like Action, Horror, Affectional and so on. Tom may like Action movies, and he would like to be affected more by his trustees or friends who have more experience in Action movies. In other words, if Tom trusts his friend James more depending on James is dominant on Action movies, Tom would more readily accept James' advice on Action movies nor the other movies. The basic idea is the same as PCA. Decompose the trust into several specific classifications which contain the significant aspects and common aspects.

These observations help me to form the basic trust model.

3.2 Problem Definition

In my scenario, there are two information resources – user-item rating matrix and user-user trust network used into the recommender systems. Specifically, the users set U is denoted as $U = \{u_1, u_2, \dots, u_N\}$ and the items set $I = \{i_1, i_2, \dots, i_M\}$. The ratings

given by the users on items can be denoted as $R_{ui}^{N \times M}$, where R_{ui} represents the value user i rating on item j . As mentioned before, in different datasets, the ratings have various ranges and values. For convenience, I normalize the rating to the new range $[0,1]$ without losing generality. In the trust network, each user u has trust neighbors denoted as N_u and the trust value denoted as T_{uv} , where T_{uv} represents the value user u trust his/her neighbor user v . If it is trust-alike network or implicit trust network, $T_{uv} = 0$ shows the user u does not trust user v . $T_{uv} = 1$ shows the user u do trust user v . If it is explicit trust network with the real trust value, then normalize the trust value to the range $[0,1]$. The objective of the problem is: when I randomly choose a user $u \in U$ and an item $i \in I$, the rating of R_{ui} can be predicted.

3.3 Trust Model

3.3.1 Making Trust Classification

Depending on Observation 3, I separate the trust into several parts. In real life, it is easy to get the classifications of movies such as action, horror, comic movies and so on. However, shown in the datasets of Epinions, FilmTrust, and Douban, there is no classifications of the items. Fortunately, in Ciao dataset shown in Fig 3.2, there is a tag of the movies which is called movie-categoryid. I use two methods to deal with the problem that datasets are lack of tag of movie-categoryid. Firstly, I divide them equally into N classifications by their ID numbers sequence. It seems like the random choose method to divide the items. ut in fact, the items' ID numbers are sequential, which means the items classifications have a latent relationship with their ID numbers. More specifically, the action movies' ID numbers are from 1 to 100, and the horror movies' ID numbers are from 101-300 and so on. However, there are several disadvantages in this partition: 1) If the ID sequence does not have a relationship with their classification, the final result will be less interpretable. 2) If I assume that the ID sequence is

influenced by the classifications, I still do not know the exact number of items in different classifications. 3) Each movie may not belong to only one classification. For example, “*Mission Impossible*” is not only an action movie but also an adventure movie.

CiaoDVD

CiaoDVD is a dataset crawled from the entire category of DVDs from the dvd.ciao.co.uk website in December, 2013 [download] .		
movie-ratings.txt: 72.7K movie ratings w format: userID, movieID, movie-categoryID, reviewID, movieRating, reviewDate		
1 userID	2 itemID	3 movie_category
46	1	1
110	7	1
111	8	1
113	9	1
116	10	1
159	12	1
230	21	1
2	23	1
2173	682	2
2178	683	2
27	684	2
12	777	3
713	782	3
2319	783	3

Fig 3. 2 Tags of Ciao dataset and the relation between itemID and movie_categoryID

In Fig 3.2, the relation between movieID and movie_categoryID proves that my assumption makes sense. Then, I assume that it will be adopted in the other three datasets. What’s more, I assume that the sequence is determined by the primary character of the movies. For example, “*Mission Impossible*” is not only an action movie but also an adventure movie, but I insist on that its primary character is action and it belongs to the classification of action movies.

To handle the problem of characters, I separate it into two situations: 1) I divide the movies equally only depending on their primary characters. I set k classifications $C_1, C_2, \dots, C_k$, where each C_i contains M/k movies. Fig.3.3 indicates the specific segmentation and corresponding classification. 2) Consider the real life and make the model more interpretable, I make the classification cross the nearest one. In the second condition, I also set k classifications $C_1, C_2, \dots, C_k$. For convenience, I set the movie numbers of each cross part same. Fig.3.4 shows the process of the second segmentation.

Fig 3. 3 Equal division of the items

Fig 3. 4 Mix-equal division of the items

Formulating Trust Aspects:

According to (Shafer 1976), I formulate the trust aspects based on the users' experience which is achieved from the rating matrix. For examples, there are two users u and v , where u is the trustor and v is the trustee. User u and v have their own experience which denoted as E_u and E_v respectively. More specifically, $E_u = \{e_{u1}, e_{u2}, \dots, e_{um}\}$ and $E_v = \{e_{v1}, e_{v2}, \dots, e_{vn}\}$. In the sets, m and n represent the numbers of experience user u and v have on different movies. And $e_u = (u, i, R_{ui}, t)$ which means user u rated on movie i with rating R_{ui} at time t . Based on the experience (Mayer, Davis, and Schoorman 1995), I model two aspects of trust and the details are shown below:

Original Similarity denoted as $S_0(u, v)$, which is the extent of the trustee v caring about the trustor u 's preference. In other words, trustor u 's willingness will be same as trustee u 's. If the value of $S_0(u, v)$ is large, which shows u 's preference is more likely to be similar to v 's. Here, it is shown as user u will give the same rating on most movies as user v does. I can comprehend $S_0(u, v)$ as the fundamental similarity between user u and v and given as:

$$S_0(u, v) = \frac{\sum_{i \in E_{u,v}} (R_{ui} - \overline{R_u})(R_{vi} - \overline{R_v})}{\sqrt{\sum_{i \in E_{u,v}} (R_{ui} - \overline{R_u})^2} \times \sqrt{\sum_{i \in E_{u,v}} (R_{vi} - \overline{R_v})^2}} \quad (1)$$

Where $E_{u,v} = E_u \cap E_v$ is the same experience user u and v have. The experience of u is the movies user u express ratings on. $\overline{R_u}$ and $\overline{R_v}$ are average ratings of user u and v 's experience respectively.

Classification Similarity denoted as $S_k(v, C_k)$, refers to the extent to which trustee v has a common attitude on the specific classification. In other words, $S_k(v, C_k)$ shows the trustee v 's characters with organized rules and segments them into specific classifications. Here k is the pre-defined number and C_k is the corresponding classification. In contrast to original similarity, classification similarity has no connection with trustor u . Based on these, the value is mostly depending on the experience of user v . Compared to original similarity, the meaning of this value can be considered as the similarity between the trustee v 's preference of classification k and the average. Therefore, the final formula can be written as:

$$S_k(v, C_k) = \frac{\sum_{i \in E_{v,k}} (R_{v,i} - \overline{R_{v,k}})(\overline{R_i} - \overline{R_k})}{\sqrt{\sum_{i \in E_{v,k}} (R_{v,i} - \overline{R_{v,k}})^2} \times \sqrt{\sum_{i \in E_{v,k}} (\overline{R_i} - \overline{R_k})^2}} \quad (2)$$

Where $E_{v,k}$ is the trustee v 's experience which belongs to classification k . $\overline{R_i}$ represents the average of the ratings given by all the users with $i \in E_{v,k}$ and $\overline{R_k}$ represents the average of the ratings given by all users on all the items in classification k . $\overline{R_{v,k}}$ represents the average of the ratings given by user v on all items in classification k .

3.3.2 Trust Prediction

T_{uv} is affected by the aspects given before, which is denoted as $F_{uv} = \{b_u, S_0(u, v), S_1(v, C_1), \dots, S_k(v, C_k)\}$. b_u is trustor's propensity, which is trustor's inherent propensity to trust others. The range of b_u is $[0, 1]$. When the value of b_u is large, the user is more easily trusting others. For example, if $b_u = 1$, u is willing to

trust anyone and if $b_u = 0$, u is hard to trust the other people. In my case, without explicit information, I treat them all the users as neutral and the value of $b_u = 0.5$.

In real life, the trust connection between users is easy to establish and trust propagates in the social network. I need to find a method to establish the connection and predict the trust value. Hence, *SVR* becomes a suitable method for me to complete this work.

The reasons for choosing *SVR* are:

1) *SVR* is a well-known method to handle imbalance datasets problems. In the four datasets in real life, most users have a few trustees in the social network, and there is much relationship between the users are unknown.

2) Using kernels can help me find the nonlinear relationship in these trust aspects. More specifically, I have training data denoted as $L = \{y_k, x_k\}_{k=1}^l$. In the data, x_k represents the value of the k -th aspects of trust. For example, $x_1 = b_u$, $x_2 = S_0(u, v)$, $x_3 = S_1(u, v)$ and so on. y_k represents the corresponding value of trust. Then, using the information in preliminaries, I can get the form of the prediction model:

$$y = f(x) = \sum_{i=1}^l (\alpha_i - \alpha_i^*) \times K(x, x_k) + \bar{b} \quad (3)$$

Where $K(x, x_k)$ is *Gaussian radial basis function (RBG)*, and $K(x, x_k) = \exp(-\|x - x_k\|^2 / 2\sigma^2)$ and α_i, α_i^* are determined by the Lagrange multipliers:

$$\begin{aligned} \max_{\alpha, \alpha^*} W(\alpha, \alpha^*) = \max_{\alpha, \alpha^*} & -\frac{1}{2} \sum_{k=1}^l \sum_{j=1}^l (\alpha_k - \alpha_k^*)(\alpha_j - \alpha_j^*) K(x_k, x_j) \\ & -\varepsilon \sum_{k=1}^l (\alpha_k + \alpha_k^*) + \sum_{i=1}^l y_k (\alpha_k - \alpha_k^*) \end{aligned} \quad (4)$$

subject to $\sum_{k=1}^l (\alpha_k - \alpha_k^*) = 0$ and $\alpha_k, \alpha_k^* \in [0, C]$

In the equation (4), ε and C are previously defined parameters which influence the final performance on the SVR method.

3.4 Trust Decomposed Matrix Factorization

After analyzing the information of trust, then it is essential to find a method to adopt

trust into a recommender system. According to Fang et al. (2014), there is a relationship between latent user feature and trust, which shows the trust is not only affected by the elements mentioned above but also influenced by the latent features of two users. The assumption observed in the social influence theory is that if a user trusts the other user, the trustor's preference will be more similar to the trustee's. Therefore, using the structure of the *PMF* model shown in preliminaries, the trust between the trustor and trustee can be modeled as a jointly conditional distribution on trust aspects and the similarity between trustor and trustee (Fig.3.5 and Fig.3.6). More specifically, in my model, I set the joint conditional distribution as $p(T|F, U, \sigma_T^2)$:

$$p(T | F, U, \sigma_T^2) = \prod_{i=1}^N \prod_{j=1}^M [\mathcal{N}(T_{i,j} | \lambda \times h(U_i, U_j) + (1 - \lambda)f(F_{i,j}), \sigma_T^2)]^{I_{uv}^T} \quad (5)$$

In the formula, $f(F_{i,j})$ is the predicted trust value depending on the trust aspects mentioned in the previous section, and $h(U_i, U_j)$ is the function calculating the similarity between the users' latent feature vectors, where $h(U_i, U_j) = e^{-(U_i - U_j)^T (U_i - U_j)}$. Then, I adopt the parameter λ to balance the different influence on trust, where $\lambda \in (0,1)$. I_{uv}^T is the indicator function. If u trust v , $I_{uv}^T = 1$ and $I_{uv}^T = 0$ otherwise.

Fig 3. 5 The left panel shows the graphical model for Probabilistic Matrix Factorization (PMF) and the right panel shows the proposed model

Fig 3. 6 The process of prediction in the trust-aware recommender system (PMF)
 Compared to *PMF*, in fig 3.5 and fig 3.6, it is clear to see that I add new items in the Bayesian inference. I can get the conditional distribution of U and V based on the ratings and trust information:

$$\begin{aligned}
 & p(U, V \mid R, T, F, \sigma_T^2, \sigma_R^2, \sigma_U^2, \sigma_V^2) \\
 &= p(U, V, R, T \mid F, \sigma_T^2, \sigma_R^2, \sigma_U^2, \sigma_V^2) / p(R, T \mid F, \sigma_T^2, \sigma_R^2, \sigma_U^2, \sigma_V^2) \\
 &= p(R \mid U, V, T, F, \sigma_T^2, \sigma_R^2, \sigma_U^2, \sigma_V^2) \times p(U, V \mid \sigma_U^2, \sigma_V^2) \\
 &\quad \times p(T \mid F, U, \sigma_T^2) / [p(R \mid \sigma_R^2) \times p(T \mid F, \sigma_T^2)] \\
 &\propto p(R \mid U, V, \sigma_R^2) \times p(U \mid \sigma_U^2) \times p(V \mid \sigma_V^2) \times p(T \mid F, U, \sigma_T^2) \quad (6)
 \end{aligned}$$

Same as *PMF*, I also take the log of the posterior distribution:

$$\begin{aligned}
 & \ln p(U, V \mid R, T, F, \sigma_T^2, \sigma_R^2, \sigma_U^2, \sigma_V^2) \\
 &= \ln p(R \mid U, V, \sigma_R^2) + \ln p(U \mid \sigma_U^2) + \ln p(V \mid \sigma_V^2) + \ln p(T \mid F, U, \sigma_T^2) \\
 &= \prod_{i=1}^N \prod_{j=1}^M [\mathcal{N}(R_{ij} \mid U_i^T V_j, \sigma_R^2)]^{I_{ij}^R} \times \prod_{i=1}^N \mathcal{N}(U_i \mid 0, \sigma_U^2 \mathbf{I}) \times \prod_{j=1}^M \mathcal{N}(V_j \mid 0, \sigma_V^2 \mathbf{I}) \\
 &\quad \times \prod_{i=1}^N \prod_{j=1}^N [\mathcal{N}(T_{i,j} \mid \lambda \times h(U_i, U_j) + (1 - \lambda)f(F_{i,j}), \sigma_T^2)] \quad (7)
 \end{aligned}$$

Using the formula in preliminaries:

$$\ln \mathcal{N}(U_i \mid 0, \sigma_U^2 \mathbf{I}) = -\frac{N}{2} \ln(\sigma_U^2) - \frac{U_i^T U_i}{2\sigma_U^2} + C_U$$

$$\begin{aligned}\ln \mathcal{N}(V_i | 0, \sigma_V^2 \mathbf{I}) &= -\frac{M}{2} \ln(\sigma_V^2) - \frac{V_i^T V_i}{2\sigma_V^2} + C_V \\ \ln[\mathcal{N}(R_{ij} | U_i^T V_j, \sigma_R^2)]^{I_{ij}^R} &= -\frac{1}{2} \left(\sum_{i=1}^N \sum_{j=1}^M I_{ij}^R \right) \ln \sigma_R^2 \\ &\quad - \frac{1}{2\sigma_R^2} \sum_{i=1}^N \sum_{j=1}^M I_{ij}^R (g(U_i^T V_j) - R_{ij})^2 + C_R\end{aligned}$$

Therefore, I can get:

$$\begin{aligned}\ln \mathcal{N}(T_{i,j} | \lambda \times h(U_i, V_j) + (1-\lambda)f(F_{i,j}), \sigma_T^2) &= -\frac{1}{2} \left(\sum_{i=1}^N \sum_{j=1}^M I_{ij}^T \right) \ln \sigma_T^2 \\ &\quad - \frac{1}{2\sigma_T^2} \sum_{i=1}^N \sum_{j=1}^M I_{ij}^T (\lambda \times h(U_i, V_j) + (1-\lambda)f(F_{i,j}) - T_{i,j})^2 + C_T\end{aligned}$$

Combining them into (7) and the final formula is written as:

$$\begin{aligned}\ln p(U, V | R, T, F, \sigma_R^2, \sigma_U^2, \sigma_V^2, \sigma_T^2) &= -\frac{1}{2\sigma_R^2} \sum_{i=1}^N \sum_{j=1}^M I_{ij}^R (g(U_i^T V_j) - R_{ij})^2 - \frac{1}{2\sigma_U^2} \sum_{i=1}^N U_i^T U_i - \frac{1}{2\sigma_V^2} \sum_{j=1}^M V_j^T V_j \\ &\quad - \frac{1}{2\sigma_T^2} \sum_{i=1}^N \sum_{j=1}^M I_{ij}^T (\lambda \times h(U_i, V_j) + (1-\lambda)f(F_{i,j}) - T_{i,j})^2 \\ &\quad - \frac{1}{2} \left(\sum_{i=1}^N \sum_{j=1}^M I_{ij}^R \right) \ln \sigma_R^2 - \frac{1}{2} \left(\sum_{i=1}^N \sum_{j=1}^M I_{ij}^T \right) \ln \sigma_T^2 \\ &\quad - \frac{1}{2} (KN \ln(\sigma_U^2) + KM \ln(\sigma_V^2)) + \text{Constant}\end{aligned}\tag{8}$$

The objective of mine is to maximize the log-posterior of latent features of users and items based on the ratings and trust information. There are some constant terms in the formula that do not affect the decomposition of R . Then, transfer the question to minimize the following part as a new objective function with keeping the same parameters:

$$L(U, V, R, T) = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^M I_{ij}^R (g(U_i^T V_j) - R_{ij})^2 + \frac{\lambda_U}{2} \sum_{i=1}^N U_i^T U_i + \frac{\lambda_V}{2} \sum_{j=1}^M V_j^T V_j$$

$$+\frac{\lambda_T}{2} \sum_1^N \sum_1^N I_{ij}^T (\lambda \times h(U_i, U_j) + (1 - \lambda)f(F_{i,j}) - T_{i,j})^2 \quad (9)$$

Where $\lambda_U = \sigma_R^2 / \sigma_U^2$, $\lambda_V = \sigma_R^2 / \sigma_V^2$ & $\lambda_T = \sigma_R^2 / \sigma_T^2$, $g(x) = 2 / (1 + e^{-x}) - 1$.

Then, I use the *SGD* for all users and items to find the minimum of the equation (9):

$$\begin{aligned} \frac{\partial L}{\partial U_i} &= \sum_{j=1}^M I_{ij}^R V_j g'(U_i^T V_j) (g(U_i^T V_j) - R_{ij}) + \lambda_U U_i \\ &\quad + \lambda_T \sum_{j=1}^N I_{ij}^T h'(U_i, U_j) (\lambda \times h(U_i, U_j) + (1 - \lambda)f(F_{i,j}) - T_{i,j}) \\ &\quad + \lambda_T \sum_{j=1}^N I_{ji}^T h'(U_j, U_i) (\lambda \times h(U_j, U_i) + (1 - \lambda)f(F_{i,j}) - T_{i,j}) \\ \frac{\partial L}{\partial V_j} &= \sum_{i=1}^M I_{ij}^R U_i g'(U_i^T V_j) (g(U_i^T V_j) - R_{ij}) + \lambda_V V_j \end{aligned} \quad (10)$$

Where g' and h' are differential of corresponding functions and the forms are:

$$\begin{aligned} g'(x) &= 2e^{-x} / (1 + e^{-x})^2 \\ h'(U_i, U_j) &= \partial h(U_i, U_j) / \partial U_i = -e^{-(U_i - U_j)^T (U_i - U_j)} \times \partial(-(U_i - U_j)^T (U_i - U_j)) / \partial U_i \\ &= -e^{-(U_i - U_j)^T (U_i - U_j)} \times (2U_i - 2U_j) \\ h'(U_j, U_i) &= \partial h(U_j, U_i) / \partial U_i = -e^{-(U_j - U_i)^T (U_j - U_i)} \times \partial(-(U_j - U_i)^T (U_j - U_i)) / \partial U_i \\ &= -e^{-(U_j - U_i)^T (U_j - U_i)} \times (2U_j - 2U_i) \end{aligned}$$

Finally, I want to make all the parameters (the parameters related to ratings like U , V , and the parameters related to the trust like α_i and α_i^*) reach to the optimization. Besides, the final U and V are decided by the equation (10), and α_i and α_i^* are determined by equation (4). What's more, I need to notice that the parameters α_i and α_i^* affect the last predictions on rating too. So, the whole process will be divided into two parts: First one, find a fixed α_i and α_i^* , then compute U and V . Second one, use the updated U and V to compute the new α_i and α_i^* until they trigger the stop conditions.

Chapter 4

Experimental Study

4.1 Benchmark Approach

I compare this new method with other four state-of-art methods which include *PMF* (Mnih and Salakhutdinov 2007), *SVD++* (Koren 2008), *SocialMF* (Jamali and Ester 2010) and *TrustMF* (Yang et al. 2013). The first two approaches do not use the trust information, while the other two adopt trust information into their models. Without loss of generality, I set corresponding parameters in my model the same values, which are extracted into Table.3. Meanwhile, in these five methods, I choose the number of latent features K as 5 and 10 respectively. Also, in my method, I keep the number of items classes K' as same as K .

For the other four methods, I adopt the source codes given by Librec (<https://www.librec.net/>).

Table. 3: *Parameters settings in methods*

Methods	Parameters
PMF	$\lambda_U=0.001, \lambda_V=0.001$
SVD++	$\lambda_U=0.001, \lambda_V=0.001$
SocialMF	$\lambda_U=0.001, \lambda_V=0.001, \lambda_T=1$
TrustMF	$\lambda_U=0.001, \lambda_V=0.001, \lambda_T=1$
My method	$\lambda_U=0.001, \lambda_V=0.001, \lambda_T=1$

4.2 Evaluation Metrics

I use 5-fold cross-validation for learning and testing, which means 80% of the data are chosen as training data and the others are testing data. Furthermore, I choose mean-square-error (MSE) and mean absolute error (MAE) to measure the performance of these five methods. MAE shows how much the prediction value deviate from the ground-truth. MSE gives the strong penalty on deviations from the ground-truth. In addition, I choose MSE nor RMSE because both the original rating values and predicted rating values are constrained lower than 1, which will lead the loss is much smaller than the loss using the initial data directly. Furthermore, the loss may be below than 1 sometimes. Therefore, it is better to use MAE in my model. Besides, I use the normalized function to clean the original rating values to $[0,1]$ in advance, then I adopt the transformation of logistic function $g(x)$ (shown in fig 4.2) to bound the predicted values into $[0,1]$. Because if I use the original logistic function (shown in fig 4.1) which maps predicted values from 0.5 to 1, It will make the model noisy. Different choice of the form of the logistic function will have different influences on results. Meanwhile, I take the original normalized data and the mapped predicted data into these two evaluation methods, which are also adopted into the baselines (*PMF*, *SVD++*, *TrustMF*, and *SocialMF*).

Fig 4. 1 *Original logistic function* $g(x) = 1/(1 + e^{-x})$

Fig 4. 2 *Revised logistic function (used in my model)* $g(x) = 2/(1 + e^{-x}) - 1$

4.3 Model Comparison

Table. 4: *Performance comparison of different methods with $K=5$*

Dimension/Class		K=5 with iters=200					
Datasets	Method	PMF	SVD++	SocialMF	TrustMF	My method	
						EqualK	CrossK
Epinions	MSE	0.23483158	0.16173242*	0.18266362	0.18999948	0.14333689	0.14883228
	MAE	0.42584385	0.35962470*	0.38678622	0.39465208	0.25085124	0.25654145
Ciao	MSE	0.21673937	0.14429018*	0.16576458	0.17501501	0.14709184	0.12089714
	MAE	0.40659948	0.33157452*	0.36231516	0.37368968	0.27133296	0.23042671
Douban	MSE	0.15907851	0.11074009*	0.13221966	0.14240553	0.11269564	0.11026557
	MAE	0.33619069	0.28095311*	0.31776733	0.33241728	0.26678029	0.26423214
FilmTrust	MSE	0.14578283	0.13415983*	0.15627394	0.16999252	0.16680258	0.54135712
	MAE	0.33422645	0.32329248*	0.35060106	0.36879902	0.30312452	0.73514765

Table. 5: *Performance comparison of different methods with $K=10$*

Dimension/Class		K=10 with iters=200					
Datasets	Method	PMF	SVD++	SocialMF	TrustMF	My method	
						EqualK	CrossK
Epinions	MSE	0.23477849	0.16172468*	0.18065236	0.18069996	0.14400779	0.14813879
	MAE	0.42577849	0.35962419*	0.38418322	0.38388306	0.25361219	0.25723211
Ciao	MSE	0.21697383	0.14422688*	0.16263202	0.16861856	0.14700474	0.11586314
	MAE	0.40694517	0.33153311*	0.35856699	0.36666530	0.27150504	0.22402393
Douban	MSE	0.15907851	0.11074965*	0.13215840	0.14950735	0.10446742	0.10267458
	MAE	0.33619069	0.28104240*	0.31781201	0.34201211	0.25394639	0.25222711
FilmTrust	MSE	0.14689665	0.13475211*	0.15512432	0.16992524	0.08368143	0.07517175
	MAE	0.33537589	0.32379326*	0.35060106	0.36879902	0.21207364	0.20103769

Firstly, I compare the results of my model with the baseline. Different from the traditional MSE and MAE used with the item $(R - U^T V)$, I use the constrained item $(\text{normalize}(R) - g(U^T V))$ to replace the original one in my model and in the benchmark approaches. In table 4 and 5, they show the performance of my model and the benchmark approaches in 4 real-life datasets with the preset parameters -- the number of latent features $K=5$ or 10 and the number of item classifications $K'=5$ or 10 . More specifically, the number of item classifications K' is chosen the same value as the number of latent features of users K . As demonstrated, my method improves the final performance mostly on 4 datasets excepting on FilmTrust dataset in the cross-item classification ($K'=5$) situation. I can see the loss is so large which is influenced by the numbers of iterations in the *Stochastic Gradient Method* with Momentum. “*” indicates the best performance of the four fundamental baselines. The red values show they are the best performance in five methods with the same parameters. In my method, the EqualK shows that the method using the K' item classifications without intersections, and the CrossK shows the method using the K' item classifications with intersections. What's more, in the situation of CrossK=5 in FilmTrust, MSE and MAE are higher than the other performance. When checking the generated data in the program, I found there are a mass of items crowded in the first classification. If the item classifications are quite imbalanced, it will lead an adverse affection on the performance. The idea of making cross item classifications is to handle with the problem of cold start. However, in that case, when making cross item classifications, the number of 1st class items increasing fast and the other class add a few items in their classes which made the problem more imbalanced. Therefore, how to deal with the imbalanced problem is still being studied.

Besides, when adopting my form of MSE and MAE, the method *SVD++* achieves the best performance among the four benchmark approaches and it also gets one global best performance in FilmTrust with $K=5$. The performance of *TrustMF* and *SocialMF* are worse than *PMF* and *SVD++* in FilmTrust. The reasons may be: 1) the latter two

methods use some hidden information such as implicit feedbacks and combine them into their models to improve the final performance. However, if the matrix is so sparse, the previous two methods cannot get enough information to enhance their results. 2) There is much noise in trust matrixes. 3) The previous assumption may not fit the model, because the trustor may not have the similar preference with the trustor.

4.4 Model Effectiveness and Sensitivity

I compare the effectiveness of original *PMF* with my method. *PMF* does not have the *SVR* components, which means it does not make the trust matrix decomposition. Then, fig 4.3 shows that my method with *SVR* components has better performance than *PMF* without *SVR* components when choosing $K=5$ or 10 . That implies when taking the trust information, it can greatly improve the effectiveness of the model. In addition, there is an exception of my model when choosing $K=5$ in FilmTrust has been explained in the previous section 4.3.

I also try to find the influence of parameter K and K' . More profoundly, my method improves the performance in different situations with different K and K' . Depending on the experience, when the number of item classification K' is large, I can easily find the trust decomposition in specific classifications and receive the accurate recommendation to reduce the loss. Meanwhile, depending on the researches, when the number of latent features K is small, it can reduce the performance of loss and reduce the noise. Therefore, the numbers of K and K' are complicated. How to choose the numbers of K' and K to balance the positive influence of item classifications and the negative influence of latent features is significant.

Fig 4.4 indicates the final performance by changing the parameter λ . In fig 4.4, the best performance is achieved when $\lambda=0.5$. In addition, $\lambda=0$ represents that the latent trust has no relation with users latent similarity and $\lambda=1$ represents that it use the users latent similarity as the trust value between users.

Fig 4. 3 The effectiveness of SVR methods used in my model on MSE and MAE with $K=5$ and 10

Fig 4. 4 Performance with different λ with $K=10$

Chapter 5

Conclusion

Based on the Probabilistic Matrix Factorization model, I try to present a model to relate the user-item rating matrix and user-user trust matrix. Also, when dealing with the trust information, I choose to decompose the trust information into several specific item classifications, which means the user's trust to another is divided into several specific classes. Then, combine these aspects into *SVR* to predict trust values in the network and extract the predicted trust values into *PMF* model. I choose four popular real-life datasets in recommender systems: Epinions, Ciao, FilmTrust, and Douban. The first three have trust network and the Douban has a social network. Finally, I compare the performance of my model with the other four performance-of-art methods: *PMF*, *SVD++*, *SocialMF*, and *TrustMF* on previous four datasets. The result shows that my method improves the performance of accuracy of rating prediction.

The contributions of my study are mainly composed of: 1) When I combine the rating information with trust information, I try to find the latent information of trust in different item classifications. 2) Use item classifications as trust aspects and remodel the rating prediction by combining trust aspects and user latent similarity. 3) The trust may have a more profound influence on the latent user preference which is thought affected by rating information in other models.

Future Work

Indeed, there are many things can be explored in my model. Based on the assumption and observations mentioned in the paper, the first complement is to add the seeds into the trust network. More specifically, there are not only commons in the social network but also the experts in the network. For example, the user who called Stan Lee, who is the father of Marvel and he is an expert on the action movies and adventure movies. If he expresses high ratings on some action or adventure movies, his ratings will potentially encourage the people who treat him as an expert or a friend to watch these movies. In other words, these users' influence is called reputation. So, I will separate the social network as two parts. The first one is the common social network, i.e. friend relationship, workmate relationship, family and so on. These relationships can be treated as private. The second one is the authority social network. For example, I follow or trust someone just because he/she is an expert in specific fields or known as public character. This relationship can be treated as public. Also, these experts and public character are chosen as seeds in social network or trust network.

The second complement is based on the assumption: the user trusts the others mostly influenced by one significant aspect. Therefore, when I use the *PMF* model to combine rating information with trust information, the conditions will be change based on the specific item classifications. This matches real-life better.

Finally, I want to figure out the problems mentioned in section 4.3. The first one is to find suitable K and K' to balance the positive influence of item classifications and the negative influence of user latent features. The second one is to find a suitable logistic function $g(x)$ to constrain the predicted values into $[0,1]$ which are more fit with truth value. The third one, which is the most important one, is to study more on how the classifications affect the users' latent preference and trust, and study on how to generate the classifications avoiding the imbalanced problem.

References

- [1] Alex J. SMOLA, BERNHARD SCHÖLKOPF. 2002 A tutorial on support vector regression. *Statistics and Computing* 14: 199-222, 2004
- [2] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, 2004. Evaluating collaborative filtering recommender systems, *ACM Transactions on Information Systems (TOIS)*, vol. 22, pp. 5-53.
- [3] F. Isinkaye, Y. Folajimi, and B. Ojokoh, 2015. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, vol. 16, pp. 261-273.
- [4] X. Su and T. M. Khoshgoftaar, 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, vol. 2009, p. 4.
- [5] P. B. Thorat, R. Goudar, and S. Barve, 2015. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, vol. 110.
- [6] A. Rafsanjani, N. Salim, A. R. Aghdam, and K. B. Fard, 2013. Recommendation Systems: a review. *International Journal of Computational Engineering Research*, vol. 3, pp. 47-52.
- [7] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, 2007. Collaborative filtering recommender systems. *The adaptive web*, ed: Springer, pp. 291-324.
- [8] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, 2001. Item-based collaborative filtering recommendation algorithms. *Proceedings of the 10th international conference on World Wide Web*, pp. 285-295.
- [9] C. A. Levinas, 2014. An Analysis of Memory Based Collaborative Filtering Recommender Systems with Improvement Proposals.
- [10] S. Girase and D. Mukhopadhyay, 2015. Role of Matrix Factorization Model in Collaborative Filtering Algorithm: A Survey. *arXiv preprint arXiv:1503.07475*.
- [11] Y. Koren, R. Bell, and C. Volinsky, 2009. Matrix factorization techniques for

recommender systems. *Computer*, vol. 42, pp. 30-37.

[12] M. J. Pazzani and D. Billsus, 2007. Content-based recommendation systems.

The adaptive web, ed: Springer, pp. 325-341.

[13] Mayer, R. C.; Davis, J. H.; and Schoorman, F. D. 1995. An integrative model of organizational trust. *Academy of Management Review* 20(3):709–734.

[14] McKnight, D. H., and Chervany, N. L. 2002. What trust means in e-commerce customer relationships: an interdisciplinary conceptual typology. *International Journal of Electronic Commerce* 6:35–60.

[15] McKnight, D. H.; Choudhury, V.; and Kacmar, C. 2002. Developing and validating trust measures for e-commerce: an integrative typology. *Information Systems Research* 13(3):334–359.

[16] Hui Fang, Yang Bao, Jie Zhang. 2014. Leveraging Decomposed Trust in Probabilistic Matrix Factorization For Effective Recommendation. *AAAI*

[17] Mnih, A., and Salakhutdinov, R. 2007. Probabilistic matrix factorization. In *Advances in neural information processing systems*, 1257-1264

[18] Ma, H.; King, I.; and Lyu, M. R. 2009. Learning to recommend with social trust ensemble. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 203–210. ACM.

[19] Guo, G.; Zhang, J.; and Thalmann, D. 2012. A simple but effective method to incorporate trusted neighbors in recommender systems. In *User Modeling, Adaptation, and Personalization (UMAP)*. Springer. 114–125.

[20] Ma, H.; Yang, H.; Lyu, M. R.; and King, I. 2008. SoRec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM)*, 931–940. ACM.

[21] Ma, H.; King, I.; and Lyu, M. R. 2009. Learning to recommend with social trust ensemble. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 203–210. ACM.

[22] Jamali, M., and Ester, M. 2010. A matrix factorization technique with trust

- propagation for recommendation in social networks. In Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys, 135–142. ACM.)
- [23] Yang, B.; Lei, Y.; Liu, D.; and Liu, J. 2013. Social collaborative filtering by trust. In Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI), 2747–2753. AAAI Press.
- [24] Cortes C. and Vapnik V. 1995. Support vector networks. *Machine Learning* 20: 273–297.
- [25] Tong Zhao, Chunping Li, Mengya Li, Qiang Ding and Li Li 2013. Social recommendation incorporating topic mining and social trust analysis. ACM
- [26] W. Yao, J. He, G. Huang, and Y. Zhang, 2014. Modeling dual role preferences for trust-aware recommendation. ACM.
- [27] Haifeng Liu, Zhuo Yang, Jun Zhang, Xiaomei Bai, Wei Wang, Feng Xi. 2015. Mining implicit correlations between users with the same role for trust-aware recommendation.
- [28] Hao Ma, Michael R. Lyu, and Irwin King. 2009. Learning to recommend with trust and distrust relationships. ACM.
- [29] G. Guo, J. Zhang, and N. Yorke Smith. 2015. TrustSVD: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings, in Proc. 29th AAAI Conf. Artif. Intell.
- [30] X. Yang, H. Steck, and Y. Liu. 2012. Circle based recommendation in online social networks, ACM.