Social Trust Prediction Using Heterogeneous Networks

JIN HUANG, FEIPING NIE, and HENG HUANG, University of Texas at Arlington
YI-CHENG TU, University of South Florida
YU LEI, University of Texas at Arlington

17

Along with increasing popularity of social websites, online users rely more on the trustworthiness information to make decisions, extract and filter information, and tag and build connections with other users. However, such social network data often suffer from severe data sparsity and are not able to provide users with enough information. Therefore, trust prediction has emerged as an important topic in social network research. Traditional approaches are primarily based on exploring trust graph topology itself. However, research in sociology and our life experience suggest that people who are in the same social circle often exhibit similar behaviors and tastes. To take advantage of the ancillary information for trust prediction, the challenge then becomes what to transfer and how to transfer. In this article, we address this problem by aggregating heterogeneous social networks and propose a novel joint social networks mining (JSNM) method. Our new joint learning model explores the user-group-level similarity between correlated graphs and simultaneously learns the individual graph structure; therefore, the shared structures and patterns from multiple social networks can be utilized to enhance the prediction tasks. As a result, we not only improve the trust prediction in the target graph but also facilitate other information retrieval tasks in the auxiliary graphs. To optimize the proposed objective function, we use the alternative technique to break down the objective function into several manageable subproblems. We further introduce the auxiliary function to solve the optimization problems with rigorously proved convergence. The extensive experiments have been conducted on both synthetic and real- world data. All empirical results demonstrate the effectiveness of our method.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Trust prediction, social network, transfer learning, nonnegative matrix factorization

ACM Reference Format:
Huang, J., Nie, F., Huang, H., Tu, Y.-C., and Lei, Y. 2013. Social trust prediction using heterogeneous networks. ACM Trans. Knowl. Discov. Data 7, 4, Article 17 (November 2013), 21 pages.
DOI: http://dx.doi.org/10.1145/2541268.2541270

1. INTRODUCTION

The ever-increasing popularity of social websites such as Facebook and LinkedIn has yielded complicated social networks and corresponding datasets with enormous sizes. With such a large amount of information about users' interaction activities, public profiles, and private content, the question of whom and what to trust has become an important challenge to users. Many online social networks allow users to explicitly express evaluations of other users, or the content they created. For example, Facebook

Corresponding author's address: H. Huang, Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX; email: heng@uta.edu.

DOI: http://dx.doi.org/10.1145/2541268.2541270

users may choose to accept or decline friend requests from strangers, based on their confidence in trustworthiness. Yahoo allows community members to rate any comment from another user as spam or regular message. Researchers generally adopt graphs to model such behaviors in social networks. Trust tags in a social network can be represented as a *trust graph*[1] $G = \langle V, E \rangle$, where V represents the collection of nodes (users) and an edge between node i and j denotes a trust vote from user i to user j. Because most users only know a very small fraction of the users and tag explicitly an even smaller number of such users, it is necessary to infer the missing values for trust prediction at the whole social network level.

Trust prediction is mainly concerned with predicting the unobserved relation between users [Leskovec et al. 2010], finding out who are friends and who are enemies in a social network. The accurate trust prediction using the available trust and distrust information observed from users is necessary to develop the social computing applications. For example, one important function for popular social websites is to suggest new relationships to a user based on the user's common friends, interests, or other properties. Since most users generally have pre-existing attitudes and opinions toward others who share certain characteristics, the online suggestion should be based on the existing evidence from the trust graph. However, trust prediction in real-world social networks is a very difficult task due to the severe data sparsity. There are two main reasons for such sparsity: the first one is the lack of diligence on the users' part and the other one is the users' concern of privacy in sharing trust links publicly. As a result, the general graph mining algorithms that rely on exploring the individual trust graph are not so well developed. In the literature, there are a few trust prediction papers using trust propagation [Guha et al. 2004; Kamvar et al. 2003]. The assumption for these methods is that users tend to trust each other given a trustable common friend. However, for popular social websites, only a small portion of entries in the trust graph are explicitly tagged. For instance, Facebook.com now has 800 million registered users according to Wikipedia, most users have at most a few thousand friends; therefore, the number of explicit connected edges is almost negligible considering the total number of potential users. The prospect of these approaches seems gloomy due to such severe sparsity of the trust graph.

It has been discovered in McPherson et al. [2001] that people who are in the same social circle often share similar behaviors and tastes. In Crandall et al. [2008], Crandall et al. give the following two main reasons. One is that people generally adopt behaviors exhibited by those they interact with. Such process is called social influence. The other more distinct reason is that people incline to form relationships with others who are already similar to them. Prior research works on inferring individual users' interests and attributes from their social neighbors [Bedi et al. 2007; Massa and Avesani 2007; Mislove et al. 2010; Wen and Lin 2010]. These articles show the possibility of improving the users' attributes prediction from the trust graph. However, a straightforward and interesting question can be raised here: is it possible to reverse the direction and explore the trust graph with the users' behavior information instead? Or is it possible to achieve an even more aggressive goal? That is, if we construct the auxiliary information graph where a large amount of entries are also missing, can we utilize all the available information and improve the predictions for *both* graphs?

In this article, we propose a joint social networks mining (JSNM) model to predict the trust and distrust in social networks by aggregating heterogeneous social networks from both the target trust domain and the auxiliary information domain. In this article, when we say two graphs are heterogenous, it implies they are from different

[1]In this article, the words *trust graph* and *trust matrix* are used interchangeably.

domains and have no apparent structural similarity and their entries generally have different scales. Because the rating information can also be formulated into a graph, our approach is to alleviate the sparsity problem in the trust graph by taking advantage of the supplementary knowledge about user behavior and discovering the implicit group-level similarity, which is jointly determined by the user-user trust graph matrix and user-item auxiliary graph matrix. This helps us find the optimal like-minded user groups across both domains. Moreover, we construct the individual affinity graphs to explore the individual geometric structures of the feature manifold to improve the prediction of the missing elements. In addition to the improvement in trust prediction accuracy, our model also helps predict the missing values in the auxiliary matrix. Meanwhile, our method can also be extended to the homogeneous datasets as a powerful collaborative filtering tool. The solution yielded by our algorithm is unique due to the orthonormal constraints and can be easily interpreted. Experimental evaluations have been carried out by using one synthetic dataset and two real-world datasets. All empirical results demonstrate that our proposed JSNM method outperforms the classic methods using a single social network graph.

The remainder of this article is organized as follows. In Section 2, we first do a brief literature review about the trust or link prediction in social networks. In Section 3, we describe the notations used in this article and formulate the new objective function. We derive our optimization method and provide the algorithm in Section 4. In Section 5, we prove the convergence of our new algorithm. We empirically validate the effectiveness of our method for trust prediction in Section 6 and conclude the article in Section 7.

2. RELATED WORK

Trust prediction can be viewed as a special case of the more general link prediction problem. There have been quite a few methods in link prediction from various perspectives, relational data modeling [Getoor and Diehl 2005], structural proximity measures [Liben-Nowell and Kleinberg 2003], and a more advanced stochastic relational model [Yu et al. 2006; Yu and Chu 2007; Yu et al. 2007]. As to the collaborative filtering methods, there are also a few classic ones, such as memory-based methods [Sarwar et al. 2001] to find k-nearest neighbors based on defined similarity measure, model-based methods [Hofmann and Puzicha 1999] to learn the preference models for similar users, and matrix factorization methods [Srebro and Jaakkola 2003; Salakhutdinov and Mnih 2007, 2008] to find a low-rank approximation for the user-item matrix. It is tempting to apply the previously mentioned collaborative filtering methods to solve the trust prediction problem; however, the trust graph has two structure properties different from the user-item matrix. The trust graph generally has transitivity and symmetric properties between a few nodes. Transitivity enables the trust propagation among users. Symmetry comes from the mutual trust between users in a social network. Such additional properties distinguish the trust graph from the typical user-item graphs where collaborative filtering methods are applicable.

Our work is more related to multirelational learning, where several relations are modeled jointly and their structures are captured simultaneously. Most methods express the given relations as a few related matrices where a row or column represents an entity. Several methods have been developed to share parameters or structure information by jointly factorizing related matrices so that knowledge can be transferred across different tasks. In Singh and Gordon [2008] and Zhu et al. [2007], an entity is represented by the same latent feature in different matrices. A few Bayesian models were also proposed, such as nonparametric latent variable models [Xu et al. 2009; Cao et al. 2010]. In particular, transfer learning has been applied to collaborative filtering

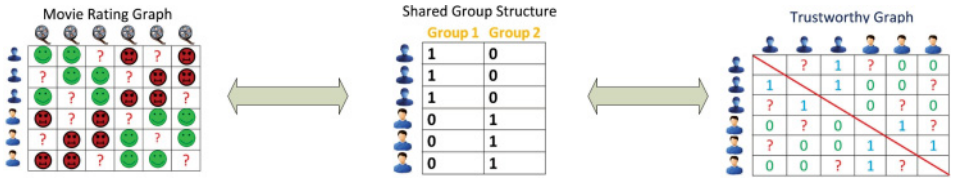


Fig. 1. A demonstration for our motivation and learning process. The shared group structure matrix is jointly determined by the rating graph and trust graph. The rating matrix contains two groups of users' reviews about movies, where a smile face represents satisfactory and an angry face represents unsatisfactory. The trust matrix contains users' trust evaluation toward other users, where 1 represents trust and 0 represents distrust. The question mark represents missing value in both graphs. The 1s in the cluster information matrix indicate users are in the corresponding group, while 0s represent users are not in that group.

[Pan et al. 2010], where Pan et al. proposed to take advantage of an auxiliary user-item rating matrix to help the prediction of the target user-item rating matrix. While this idea is intuitive and straightforward, such method is too idealistic to assume the existence of such a related and dense auxiliary rating matrix. Our recent work [Huang et al. 2012] was the first one utilizing the transfer learning between the trust graph and rating graph to simultaneously predict human social behaviors. This article is an extension of our previous work in Huang et al. [2012].

3. JOINT MANIFOLD FACTORIZATION

In this section, we will introduce our new JSNM objective function to aggregate the heterogeneous social networks. Prior to this, we will reveal the implicit connection between the target user-user trust graph and auxiliary user-item rating graph.[2]

As mentioned, trusted users in a social network often display similar behaviors and tastes. Meanwhile, social network users become friends due to the similar background and interest. Therefore, the trust graph and rating graph should contain some structure similarity in spite of the apparent difference, if the coincidence of the similar ratings contributes to such trust. As a result, the trust prediction accuracy can be improved with the aid of rating graph information and vice versa. In summary, we transfer the knowledge from different domains to circumvent the sparsity constraint and help predict the entries in both matrices. Figure 1 is a demonstration of our motivation.

In our proposed solution, we plan to share the implicit group structure between two graphs, which is jointly determined by the trust graph and rating graph. This answers the two most important questions for transfer learning: what to transfer and how to transfer [Pan et al. 2010].

3.1. Notations

We use boldface uppercase letters, such as $\mathbf{X}$, to denote matrices, and $\mathbf{X}_{i\cdot}$, $\mathbf{X}_{\cdot j}$, and X_{ij} to denote the ith row, jth column, and the entry located at (i, j) of $\mathbf{X}$, respectively. In our setting, for simplicity, we only discuss two matrices, $\mathbf{G}_1$ and $\mathbf{G}_2$ case, then extend the objective function to multiple matrices case. We further assume that $\mathbf{G}_1 \in \mathbb{R}^{n\times m_1}$ and $\mathbf{G}_2 \in \mathbb{R}^{n\times m_2}$ are the trust graph and rating graph, respectively, where n is the number of identical users in both domains, m_1 is the number of users who receive trust votes, and m_2 is the number of different items. $\Omega_1 \subset \mathbf{G}_1$ and $\Omega_2 \subset \mathbf{G}_2$ are entries known in corresponding graphs.

[2]We will use abbreviation trust graph and rating graph for the following context.

3.2. Objective Function Formulation

Inspired by the aforementioned assumption, we target the joint matrix factorization to find out the shared group structure between two graphs:

$$\min_{\mathbf{U},\mathbf{V}_1,\mathbf{V}_2,c} \left\|\mathbf{G}_1 - \mathbf{U}\mathbf{V}_1^T\right\|_F^2 + \left\|c\,\mathbf{G}_2 - \mathbf{U}\mathbf{V}_2^T\right\|_F^2. \tag{1}$$

Here $\mathbf{U} \in \mathbb{R}^{n\times l}$, $\mathbf{V}_1 \in \mathbb{R}^{m_1\times l}$, and $\mathbf{V}_2 \in \mathbb{R}^{m_2\times l}$, where l is the number of group parameter to be determined. $c > 0$ is a scalar adjusting the scale inconsistency between graphs since the two graphs are from different domains. Here $\mathbf{U}$ is jointly determined by the trust graph and rating graph structures; therefore, it provides the shared group structure for both graphs. Since rows represent users in both graphs, we could group users based on $\mathbf{U}$ and then conduct the trust and rating prediction with $\mathbf{V}_1$ and $\mathbf{V}_2$, respectively. It can be observed that $\mathbf{U}$ carries the knowledge of both trust graph and rating graph, and such framework becomes especially useful since both graphs usually have data sparsity issues for real datasets.

While the previous model takes into account the common row group structure in terms of both matrices, it fails to take into account the social network constraint. To overcome this drawback, we include the Laplacian regularity term [He and Niyogi 2003; Cai et al. 2008]. To be specific,

$$\begin{aligned}\min_{\mathbf{U},\mathbf{V}_1,\mathbf{V}_2,c} & \left\|\mathbf{G}_1 - \mathbf{U}\mathbf{V}_1^T\right\|_F^2 + \left\|c\,\mathbf{G}_2 - \mathbf{U}\mathbf{V}_2^T\right\|_F^2 \\ & + \lambda Tr\left(\mathbf{V}_1^T\mathbf{L}_1\mathbf{V}_1\right) + \lambda Tr\left(\mathbf{V}_2^T\mathbf{L}_2\mathbf{V}_2\right) \\ s.t.\quad & \mathbf{V}_1\mathbf{V}_1^T = \mathbf{I}, \mathbf{V}_2\mathbf{V}_2^T = \mathbf{I},\quad \mathbf{U} \geq 0, \mathbf{V}_1 \geq 0, \mathbf{V}_2 \geq 0.\end{aligned} \tag{2}$$

Here $\lambda > 0$ is a scalar parameter to be tuned; $\mathbf{L}_1$ and $\mathbf{L}_2$ are the Laplacian graphs based on the columns of $\mathbf{G}_1$ and $\mathbf{G}_2$, respectively; and Tr is the trace operation that yields the sum of diagonal elements of the matrix.

In our objective function, to incorporate the social network information, we add two graph Laplacian regularization terms. As the trust graph, the $\mathbf{G}_1$ graph explicitly shows the users' trust/friendship relations in social networks. As the rating graph, the $\mathbf{G}_2$ graph indicates the users' taste/hobby similarity, from which we can learn users' implicit relations. Thus, the graph Laplacian $\mathbf{L}_1$ and $\mathbf{L}_2$ represent the explicit and implicit social relations of users. When we predict the users' trust relations, these existing social relations between users should be preserved. Thus, we add two graph Laplacian regularization terms as in Equation (2).

The details of $\mathbf{L}_1$ and $\mathbf{L}_2$ constructions are given in the next section. We impose the orthogonal constraints on $\mathbf{V}_1$ and $\mathbf{V}_2$ to ensure the uniqueness of the solution. Suppose $\mathbf{U}^*$, $\mathbf{V}_1^*$, and $\mathbf{V}_2^*$ are the solutions to Equation (2); then for any given nonzero constant $c_1 > 1$, $c_1\mathbf{U}^*$ and $\mathbf{V}_1^*/c_1$ would give the same value in the first term and a lower value for the third term. This is true no matter if $\mathbf{U}^*$ and $\mathbf{V}_1^*$ are local or global optimum solutions. The same reasoning applies to $\mathbf{V}_2$. In other words, the optimal solution to Equation (2) does not exist without the constraints. With the orthogonal and nonnegative constraints for $\mathbf{V}_1$ and $\mathbf{V}_2$, our solution is the unique local optimum solution for the nonconvex objective function 2.

3.3. General Formulation

There are a few possible generalizations to Equation (2) we want to point out.

First, it can be easily extended to the multiple matrices case. The objective function would then be

$$\min_{\mathbf{U},\mathbf{V}_1\ldots\mathbf{V}_n} \sum_{i=1}^{n} \left\| c_i\mathbf{G}_i - \mathbf{U}\mathbf{V}_i^T \right\|_F^2 + \lambda \sum_{i=1}^{n} Tr\left(\mathbf{V}_i^T\mathbf{L}_i\mathbf{V}_i\right).$$
$$s.t. \quad \mathbf{V}_i^T\mathbf{V}_i = \mathbf{I}, \mathbf{V}_i \geq 0, \ i = 1, \ldots, n \tag{3}$$

The U here would then contain the common information among multiple matrices.

Second, although our motivation is to capture the shared pattern among users, it could be used as a powerful collaborative filtering tool. For example, our framework can also be applied to the item-user case (Section 6.3), where the reviews are from users in different domains.

4. OPTIMIZATION AND ALGORITHM

In the following, we will derive a solution to Equation (2). As we see, minimizing Eqquation (2) is with respect to $\mathbf{U}$, $\mathbf{V}_1$, $\mathbf{V}_2$, and c, and we cannot give a closed-form solution. We will present an alternating scheme to optimize the objective; this procedure repeats until convergence.

4.1. Initialization

As mentioned in the introduction, the social graphs generally have a large number of missing values; therefore, the initialization is almost necessary in trust prediction to replace those missing values for methods that require similarity calculation or structure exploration. In this article, for any missing entry $\mathbf{G}_{ij}$, we use the mean of the available entries in the corresponding row and column to impute this. For a user-item rating matrix, such initialization combines the available information for both the individual user rating habit and other users' ratings on a particular item. For a user-user trust matrix, such initialization considers both user i's and user j's social circle influence.

After the initial imputation, we construct the Laplacian graphs of both social networks. As mentioned, the main purpose of the Laplacian terms is to incorporate the data geometric information, because it is found that many real-world data distribute on low-dimensional manifold embedded in the high-dimensional ambient space [Roweis and Saul 2000]. The Laplacian graph is to discretely approximate the manifold, whose vertices correspond to the data samples, while the edge weight represents the affinity between the data points. One common assumption about the affinity between data points is the cluster assumption [Chapelle et al. 2006], which claims if two data samples are close to each other in the input space, then they are also close to each other in the embedding space. This assumption has been widely used in spectral clustering [von Luxburg 2006; Shi and Malik 2000; Ng et al. 2001]. To be specific, in this paper, we define the edge weight matrix $\mathbf{W}$ as follows:

$$\mathbf{W}_{ij} = \begin{cases} 1 : \mathbf{G}_{i.} \in N_k(\mathbf{G}_{.j}) \ or \ \mathbf{G}_{j.} \in N_k(\mathbf{G}_{.i}), \\ 0 : otherwise \end{cases}$$

where $N_k(\mathbf{G}_{i.})$ denotes the set of k nearest neighbors of $\mathbf{G}_{i.}$. We calculate the Euclidean distances between users for each graph, then construct the corresponding $\mathbf{W}$s based on the top k similar users for each user. It is easy to see $\mathbf{W}$s are symmetric. Let graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is a diagonal matrix whose entries are column sums of $\mathbf{W}$, $\mathbf{D}_{ii} = \sum_j \mathbf{W}_{ij}$. Corresponding to trust graph $\mathbf{G}_1$ and rating graph $\mathbf{G}_2$, we construct $\mathbf{L}_1$ and $\mathbf{L}_2$.

After that, we construct $\mathbf{V}_1$ and $\mathbf{V}_2$ based on k-means on columns for $\mathbf{G}_1$ and $\mathbf{G}_2$, respectively. For the ith row of $\mathbf{V}_1$, if this row belongs to the jth cluster, then $\mathbf{V}_1(i, j) = 1$, and all other elements in the ith row are 0. $\mathbf{V}_2$ is initialized in the same manner.

Now we come to the optimization of our objective function. When we optimize the objective function Equation (2), we iteratively solve $\mathbf{U}$, $\mathbf{V}_1$, $\mathbf{V}_2$, and c in an alternating manner. In other words, we will optimize the objective with respect to one variable while fixing the other variables. Such process repeats until convergence.

4.2. Computation of U

Optimizing Equation (2) with respect to $\mathbf{U}$ is equivalent to optimizing

$$\begin{aligned} &J_1 = \left\|\mathbf{G}_1 - \mathbf{U}\mathbf{V}_1^T\right\|_F^2 + \left\|c\,\mathbf{G}_2 - \mathbf{U}\mathbf{V}_2^T\right\|_F^2 \\ &s.t.\quad \mathbf{V}_1^T\mathbf{V}_1 = \mathbf{I},\ \ \mathbf{V}_2^T\mathbf{V}_2 = \mathbf{I}, \mathbf{V}_1 \geq 0, \mathbf{V}_2 \geq 0. \end{aligned} \tag{4}$$

Setting $\frac{\partial J_1}{\partial \mathbf{U}} = 0$ leads to the following updating formula:

$$\mathbf{U} = \frac{\mathbf{G}_1\mathbf{V}_1 + c\,\mathbf{G}_2\mathbf{V}_2}{2}. \tag{5}$$

4.3. Computation of $\mathbf{V}_1$

Optimizing Equation (2) with respect to $\mathbf{V}_1$ is equivalent to optimizing

$$\begin{aligned} &J_2 = \left\|\mathbf{G}_1 - \mathbf{U}\mathbf{V}_1^T\right\|_F^2 + \lambda Tr\big(\mathbf{V}_1^T\mathbf{L}_1\mathbf{V}_1\big). \\ &s.t.\quad \mathbf{V}_1^T\mathbf{V}_1 = \mathbf{I}, \mathbf{V}_1 \geq 0 \end{aligned} \tag{6}$$

For the constraint $\mathbf{V}_1^T\mathbf{V}_1 = \mathbf{I}$, we cannot get a closed-form solution of $\mathbf{V}_1$. Therefore, we will present an iterative multiplicative updating algorithm. We introduce the Lagrangian multiplier $\boldsymbol{\alpha} \in \mathbb{R}^{l\times l}$, and the corresponding Lagrangian function is

$$L(\mathbf{V}_1) = \left\|\mathbf{G}_1 - \mathbf{U}\mathbf{V}_1^T\right\|_F^2 + \lambda Tr\big(\mathbf{V}_1^T\mathbf{L}_1\mathbf{V}_1\big) - Tr\big(\boldsymbol{\alpha}\big(\mathbf{V}_1^T\mathbf{V}_1 - \mathbf{I}\big)\big). \tag{7}$$

Setting $\frac{\partial L(\mathbf{V}_1)}{\partial \mathbf{V}_1} = 0$ and using the orthogonal constrain $\mathbf{V}_1^T\mathbf{V}_1 = \mathbf{I}$, we obtain

$$\begin{aligned} &-\mathbf{G}_1^T\mathbf{U} + \lambda\mathbf{L}_1\mathbf{V}_1 - \mathbf{V}_1\boldsymbol{\alpha} = 0 \\ &\Rightarrow \boldsymbol{\alpha} = -\mathbf{V}_1^T\mathbf{G}_1^T\mathbf{U} + \lambda\mathbf{V}_1^T\mathbf{L}_1\mathbf{V}_1. \end{aligned} \tag{8}$$

Using the Karush-Kuhn-Tucker condition [Boyd and Vandenberghe 2004] $\boldsymbol{\alpha}\cdot\mathbf{V}_1 = 0$, where $\cdot$ is the element-wise product operator and thereafter, we get

$$\big(-\mathbf{V}_1^T\mathbf{G}_1^T\mathbf{U} + \lambda\mathbf{V}_1^T\mathbf{L}_1\mathbf{V}_1\big)\cdot\mathbf{V}_1 = 0. \tag{9}$$

Introducing $\mathbf{L}_1 = \mathbf{L}_1^+ - \mathbf{L}_1^-$, $\mathbf{V}_1 = \mathbf{V}_1^+ - \mathbf{V}_1^-$, and $\mathbf{U} = \mathbf{U}^+ - \mathbf{U}^-$ where $U_{ij}^+ = (|U_{ij}| + U_{ij})/2$ and $U_{ij}^- = (|U_{ij}| - U_{ij})/2$ [Ding et al. 2010] and $\mathbf{L}_1$,$\mathbf{V}_1$ defined in a similar fashion, we obtain

$$\big(\mathbf{G}_1^T\mathbf{U}^- + \lambda\mathbf{L}_1^+\mathbf{V}_1 + \mathbf{V}_1\boldsymbol{\alpha}^- - \mathbf{G}_1^T\mathbf{U}^+ - \lambda\mathbf{L}_1^-\mathbf{V}_1 - \mathbf{V}_1\boldsymbol{\alpha}^+\big)\cdot\mathbf{V}_1 = 0. \tag{10}$$

Equation (10) leads to the following updating formula:

$$(\mathbf{V}_1)_{ij} \leftarrow (\mathbf{V}_1)_{ij}\sqrt{\frac{\big[\mathbf{G}_1^T\mathbf{U}^+ + \lambda\mathbf{L}_1^-\mathbf{V}_1 + \mathbf{V}_1\boldsymbol{\alpha}^+\big]_{ij}}{\big[\mathbf{G}_1^T\mathbf{U}^- + \lambda\mathbf{L}_1^+\mathbf{V}_1 + \mathbf{V}_2\boldsymbol{\alpha}^-\big]_{ij}}}. \tag{11}$$

4.4. Computation of $\mathbf{V}_2$

Optimizing Equation (2) with respect to $\mathbf{V}_2$ is equivalent to optimizing

$$\begin{aligned} &J_3 = \left\| c\,\mathbf{G}_2 - \mathbf{U}\mathbf{V}_2^T \right\|_F^2 + \lambda Tr\left(\mathbf{V}_2^T \mathbf{L}_2 \mathbf{V}_2\right). \\ &s.t. \quad \mathbf{V}_2^T \mathbf{V}_2 = \mathbf{I}, \mathbf{V}_2 \geq 0 \end{aligned} \tag{12}$$

The optimization with the previous equation is almost identical to the previous subsection,

$$L(\mathbf{V}_2) = \left\| c\,\mathbf{G}_2 - \mathbf{U}\mathbf{V}_2^T \right\|_F^2 + \lambda Tr\left(\mathbf{V}_2^T \mathbf{L}_2 \mathbf{V}_2\right) - Tr\left(\boldsymbol{\beta}\left(\mathbf{V}_2^T \mathbf{V}_2 - \mathbf{I}\right)\right). \tag{13}$$

Setting $\frac{\partial L(\mathbf{V}_2)}{\partial \mathbf{V}_2} = 0$ and using the orthogonal constrain $\mathbf{V}_2^T \mathbf{V}_2 = \mathbf{I}$, we obtain

$$\begin{aligned} &-c\,\mathbf{G}_2^T \mathbf{U} + \lambda \mathbf{L}_2 \mathbf{V}_2 - \mathbf{V}_2 \boldsymbol{\beta} = 0 \\ &\Rightarrow \boldsymbol{\beta} = -c\,\mathbf{V}_2^T \mathbf{G}_2^T \mathbf{U} + \lambda \mathbf{V}_2^T \mathbf{L}_2 \mathbf{V}_2. \end{aligned} \tag{14}$$

Using the Karush-Kuhn-Tucker condition [Boyd and Vandenberghe 2004] $\boldsymbol{\beta} \cdot \mathbf{V}_2 = 0$, we get

$$\left(-c\,\mathbf{V}_2^T \mathbf{G}_2^T \mathbf{U} + \lambda \mathbf{V}_2^T \mathbf{L}_2 \mathbf{V}_2\right) \cdot \mathbf{V}_2 = 0. \tag{15}$$

Introducing $\mathbf{L}_2 = \mathbf{L}_2^+ - \mathbf{L}_2^-$, $\mathbf{V}_2 = \mathbf{V}_2^+ - \mathbf{V}_2^-$, and $\mathbf{U} = \mathbf{U}^+ - \mathbf{U}^-$ where $U_{ij}^+ = \left(\left|U_{ij}\right| + U_{ij}\right)/2$ and $U_{ij}^- = \left(\left|U_{ij}\right| - U_{ij}\right)2$ [Ding et al. 2010] and $\mathbf{L}_2$,$\mathbf{V}_2$ defined in a similar fashion, we obtain

$$\left(c\,\mathbf{G}_2^T \mathbf{U}^- + \lambda \mathbf{L}_2^+ \mathbf{V}_2 + \mathbf{V}_2 \boldsymbol{\beta}^- - c\,\mathbf{G}_2^T \mathbf{U}^+ - \lambda \mathbf{L}_2^- \mathbf{V}_2 - \mathbf{V}_2 \boldsymbol{\beta}^+\right) \cdot \mathbf{V}_2 = 0. \tag{16}$$

Equation (16) leads to the following updating formula:

$$(\mathbf{V}_2)_{ij} \leftarrow (\mathbf{V}_2)_{ij} \sqrt{\frac{\left[c\mathbf{G}_2^T \mathbf{U}^+ + \lambda \mathbf{L}_2^- \mathbf{V}_2 + \mathbf{V}_2 \boldsymbol{\beta}^+\right]_{ij}}{\left[c\mathbf{G}_2^T \mathbf{U}^- + \lambda \mathbf{L}_2^+ \mathbf{V}_2 + \mathbf{V}_2 \boldsymbol{\beta}^-\right]_{ij}}}. \tag{17}$$

4.5. Computation of c

Optimizing Equation (2) with respect to c is equivalent to optimizing

$$J_4 = \left\| c\,\mathbf{G}_2 - \mathbf{U}\mathbf{V}_2^T \right\|_F^2. \tag{18}$$

The above task is equivalent to

$$\min_c Tr\left(c\,\mathbf{G}_2 - \mathbf{U}\mathbf{V}_2^T\right)\left(c\,\mathbf{G}_2 - \mathbf{U}\mathbf{V}_2^T\right)^T.$$

This can be written as

$$\min_c Ac^2 - 2Bc + D,$$

where $A = Tr(\mathbf{G}_2\mathbf{G}_2^T)$, $B = Tr(\mathbf{U}\mathbf{V}_2^T\mathbf{G}_2^T)$, $D = Tr(\mathbf{U}\mathbf{V}_2^T\mathbf{V}_2\mathbf{U}^T)$. It is a quadratic equation in c, and the solution is then

$$c = \frac{Tr\left(\mathbf{U}\mathbf{V}_2^T\mathbf{G}_2^T\right)}{Tr\left(\mathbf{G}_2\mathbf{G}_2^T\right)}. \tag{19}$$

In summary, we present the iterative multiplicative updating algorithm of optimizing Equation (2) in Algorithm 1. Because the targeted problem is a nonconvex one, there

ALGORITHM 1: Joint Manifold Factorization Algorithm

Input: $\mathbf{G}_1$,$\mathbf{G}_2$, maximum number of iterations T
Output: Converged $\mathbf{U}$, $\mathbf{V}_1$, and $\mathbf{V}_2$
Initialize missing entries in $\mathbf{G}_1$ and $\mathbf{G}_2$ using the row-column average.
Initialize V_1 and V_2 using k-means clustering, initialize c according to scale discrepancy between graphs.
Construct Laplacian graphs $\mathbf{L}_1$ and $\mathbf{L}_2$.
while *not converged and iteration t less than T* **do**

Compute $U = \frac{\mathbf{G}_1\mathbf{V}_1 + c\,\mathbf{G}_2\mathbf{V}_2}{2}$

Compute $(\mathbf{V}_1)_{ij} \leftarrow (\mathbf{V}_1)_{ij}\sqrt{\frac{\left[\mathbf{G}_1^T\mathbf{U}^+ + \lambda\mathbf{L}_1^-\mathbf{V}_1 + \mathbf{V}_1\boldsymbol{\alpha}^+\right]_{ij}}{\left[\mathbf{G}_1^T\mathbf{U}^- + \lambda\mathbf{L}_1^+\mathbf{V}_1 + \mathbf{V}_2\boldsymbol{\alpha}^-\right]_{ij}}}$

Compute $(\mathbf{V}_2)_{ij} \leftarrow (\mathbf{V}_2)_{ij}\sqrt{\frac{\left[c\mathbf{G}_2^T\mathbf{U}^+ + \lambda\mathbf{L}_2^-\mathbf{V}_2 + \mathbf{V}_2\boldsymbol{\beta}^+\right]_{ij}}{\left[c\mathbf{G}_2^T\mathbf{U}^- + \lambda\mathbf{L}_2^+\mathbf{V}_2 + \mathbf{V}_2\boldsymbol{\beta}^-\right]_{ij}}}$

Compute $c = \frac{Tr(\mathbf{U}\mathbf{V}_2^T\mathbf{G}_2^T)}{Tr(\mathbf{G}_2\mathbf{G}_2^T)}$

end

is no guarantee that Algorithm 1 will converge to the global optimum. However, the orthogonal constraints in objective function ensure the yielded solution is unique.

The convergence criterion here is the relative change of the object function value at the consecutive steps is less than 10^{-4}. The previously mentioned loop always exits within 20 iterations for the subsequent experiments.

4.6. Algorithm Complexity Analysis

In this part, we want to analyze the time complexity of our algorithm. We would analyze the cost for each phase separately. Let us assume $n \geq max(m_1, m_2)$ to keep the notations simple.

For the missing values initialization, each missing entry needs to calculate its row and column average, of order $O(n + m_1)$ and $O(n + m_2)$, respectively. Therefore, the initialization cost would be $O(n^2m_1)$ and $O(n^2m_2)$, respectively, and the total cost would be $O(n^2(m_1 + m_2))$.

Now it comes to the $\mathbf{V}_1$ and $\mathbf{V}_2$ initialization. k-means of $\mathbf{G}_1$ takes $O(knm_1)$ and k-means of $\mathbf{G}_2$ takes $O(knm_2)$; therefore, the total cost would be $O(kn(m_1 + m_2))$.

The last step of the initialization is to construct the Laplacian graphs. It takes $O(knm_1)$ and $O(knm_2)$ to construct the k-nearest neighbor graphs for $\mathbf{G}_1$ and $\mathbf{G}_2$, respectively. The total cost would then be $O(kn(m_1 + m_2))$.

Now it comes to the computation of $\mathbf{U}$, $\mathbf{V}_1$, and $\mathbf{V}_2$. We focus on the discussion of the tth iteration.

From the $\mathbf{U}$ updating formula Equation (5), it takes at most $O(m_1^3 + m_2^3)$; however, since $\mathbf{V}_1$ was initialized to have only one nonzero element in each row and in general sparse during the updating process, indeed it could be reduced to $O(m_1^2 + m_2^2)$ [Yuster and Zwick 2005].

For the update of $\mathbf{V}_1$, since $\mathbf{V}_1$ is sparse, it takes $O(k^2m_1^2)$ to calculate $\boldsymbol{\alpha}$; as Eq. (11) is an element-wise operation, it takes $O(nkm_1)$ to update $\mathbf{V}_1$.

For the update of $\mathbf{V}_2$, again since $\mathbf{V}_2$ is sparse, it takes $O(k^2m_2^2)$ to calculate $\boldsymbol{\beta}$; as Eq. (17) is an element-wise operation, it takes $O(nkm_2)$ to update $\mathbf{V}_2$.

For the update of c, it takes $O(k^2m_2^2)$ to calculate A, $O(m_2^3)$ to calculate B, and the total cost would then be $O(m_2^3)$.

Therefore, the total cost for one iteration is $O(n^2(m_1 + m_2))$. As specified, our algorithm usually converges in a few iterations independent of matrix size, and the total

multiplicative update process takes $O(n^2(m_1 + m_2))$. The total complexity of our algorithm is then $O(n^2(m_1 + m_2))$.

5. CONVERGENCE ANALYSIS

In this section, we will prove the convergence of Algorithm 1. We use the classic auxiliary function approach used in Lee and Seung [2000].

Definition 5.1 *(Auxiliary Function) [Lee and Seung 2000].* $Z(h, h')$ is an auxiliary function for $F(h)$ if the conditions

$$Z(h, h') \geq F(h), Z(h, h) = F(h)$$

are satisfied.

LEMMA 5.2 [LEE AND SEUNG 2000]. *If Z is an auxiliary function for F, then F is nonincreasing under the update*

$$h^{(t+1)} = \arg\min_h Z(h, h^{(t)}).$$

PROOF. $F(h^{(t+1)}) \leq Z(h^{(t+1)}, h^{(t)}) \leq Z(h^{(t)}, h^{(t)}) = F(h^{(t)})$. □

LEMMA 5.3 [DING ET AL. 2010]. *For any nonnegative matrices $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{k\times k}$, $\mathbf{S} \in \mathbb{R}^{n\times k}$, $\mathbf{S}' \in \mathbb{R}^{n\times k}$, and $\mathbf{A}$, $\mathbf{B}$ are symmetric, and then the following inequality holds:*

$$\sum_{i=1}^{n}\sum_{p=1}^{k} \frac{(\mathbf{AS}'\mathbf{B})_{ip}\mathbf{S}_{ip}^2}{\mathbf{S}'_{ip}} \geq Tr(\mathbf{S}^T\mathbf{ASB}).$$

THEOREM 5.4. *Let*

$$J(\mathbf{V}_1) = Tr\big(\lambda\mathbf{V}_1^T\mathbf{L}_1\mathbf{V}_1 - 2\mathbf{G}_1^T\mathbf{U}\mathbf{V}_1^T + \boldsymbol{\alpha}\mathbf{V}_1^T\mathbf{V}_1\big). \tag{20}$$

Then the following function

$$\begin{aligned} Z(\mathbf{V}_1, \mathbf{V}'_1) \;=\; & \lambda\sum_{ij}\frac{(\mathbf{L}_1^+\mathbf{V}'_1)_{ij}\mathbf{V}_{1,ij}^2}{\mathbf{V}'_{1,ij}} - \lambda\sum_{ijk}(\mathbf{L}_1^-)_{jk}\mathbf{V}'_{1,ji}\mathbf{V}'_{1,ki}\left(1+\log\frac{\mathbf{V}_{1,ji}\mathbf{V}_{1,ki}}{\mathbf{V}'_{1,ji}\mathbf{V}'_{1,ki}}\right) \\ & - 2\sum_{ij}\mathbf{G}_1^T\mathbf{U}^+\mathbf{V}'_{1,ij}\left(1+\log\frac{\mathbf{V}_{1,ij}}{\mathbf{V}'_{1,ij}}\right) + 2\sum_{ij}\mathbf{G}_1^T\mathbf{U}^-\frac{\mathbf{V}_{1,ij}^2+\mathbf{V}_{1,ij}'^2}{2\mathbf{V}'_{1,ij}} \\ & + \sum_{ij}\boldsymbol{\alpha}^+\mathbf{V}_{1,ij}^2 - \sum_{ijk}\boldsymbol{\alpha}^-\mathbf{V}'_{1,ij}\mathbf{V}'_{1,ik}\left(1+\log\frac{\mathbf{V}_{1,ij}\mathbf{V}_{1,ik}}{\mathbf{V}'_{1,ij}\mathbf{V}'_{1,ik}}\right) \end{aligned}$$

is an auxiliary function for $J(\mathbf{V}_1)$. Furthermore, it is a convex function in $\mathbf{V}_1$ and its global minimum is

$$(\mathbf{V}_1)_{ij} \leftarrow (\mathbf{V}_1)_{ij}\sqrt{\frac{\big[\mathbf{G}_1^T\mathbf{U}^+ + \lambda\mathbf{L}_1^-\mathbf{V}_1 + \mathbf{V}_1\boldsymbol{\alpha}^+\big]_{ij}}{\big[\mathbf{G}_1^T\mathbf{U}^- + \lambda\mathbf{L}_1^+\mathbf{V}_1 + \mathbf{V}_2\boldsymbol{\alpha}^-\big]_{ij}}}. \tag{21}$$

PROOF. See Appendix A. □

THEOREM 5.5. *Updating $\mathbf{V}_1$ using Equation (11) will monotonically decrease the value of the objective in Eq. (2); hence, it converges.*

PROOF. By Lemma 5.2 and Theorem 5.4, we can get that $J(\mathbf{V}_1^0) = Z(\mathbf{V}_1^0, \mathbf{V}_1^0) \geq Z(\mathbf{V}_1^1, \mathbf{V}_1^0) \geq J(\mathbf{V}_1^1) \geq \cdots$ so $J(\mathbf{V}_1)$ is monotonically decreasing. As $J(\mathbf{V}_1)$ is nonnegative (i.e, bounded below), the theorem is self-evident. □

THEOREM 5.6. *Let*

$$J(\mathbf{V}_2) = Tr\big(\lambda \mathbf{V}_2^T \mathbf{L}_2 \mathbf{V}_2 - 2c\,\mathbf{G}_2^T \mathbf{U} \mathbf{V}_2^T + \boldsymbol{\beta} \mathbf{V}_2^T \mathbf{V}_2\big). \tag{22}$$

Then the following function

$$\begin{aligned} Z(\mathbf{V}_2, \mathbf{V}_2') \;=\; & \lambda \sum_{ij} \frac{(\mathbf{L}_2^+ \mathbf{V}_2')_{ij} \mathbf{V}_{2,ij}^2}{\mathbf{V}_{2,ij}'} - \lambda \sum_{ijk} (\mathbf{L}_2^-)_{jk} \mathbf{V}_{2,ji}' \mathbf{V}_{2,ki}' \left(1 + \log \frac{\mathbf{V}_{2,ji} \mathbf{V}_{2,ki}}{\mathbf{V}_{2,ji}' \mathbf{V}_{2,ki}'}\right) \\ & - 2 \sum_{ij} c\,\mathbf{G}_2^T \mathbf{U}^+ \mathbf{V}_{2,ij}' \left(1 + \log \frac{\mathbf{V}_{2,ij}}{\mathbf{V}_{2,ij}'}\right) + 2 \sum_{ij} c\,\mathbf{G}_2^T \mathbf{U}^- \frac{\mathbf{V}_{2,ij}^2 + \mathbf{V}_{2,ij}'^2}{2\mathbf{V}_{2,ij}'} \\ & + \sum_{ij} \boldsymbol{\beta}^+ \mathbf{V}_{2,ij}^2 - \sum_{ijk} \boldsymbol{\beta}^- \mathbf{V}_{2,ij}' \mathbf{V}_{2,ik}' \left(1 + \log \frac{\mathbf{V}_{2,ij} \mathbf{V}_{2,ik}}{\mathbf{V}_{2,ij}' \mathbf{V}_{2,ik}'}\right) \end{aligned}$$

is an auxiliary function for $J(\mathbf{V}_2)$. Furthermore, it is a convex function in $\mathbf{V}_2$ and its global minimum is

$$(\mathbf{V}_2)_{ij} \leftarrow (\mathbf{V}_2)_{ij} \sqrt{\frac{\left[c\mathbf{G}_2^T \mathbf{U}^+ + \lambda \mathbf{L}_2^- \mathbf{V}_2 + \mathbf{V}_2 \boldsymbol{\beta}^+\right]_{ij}}{\left[c\mathbf{G}_2^T \mathbf{U}^- + \lambda \mathbf{L}_2^+ \mathbf{V}_2 + \mathbf{V}_2 \boldsymbol{\beta}^-\right]_{ij}}}. \tag{23}$$

PROOF. See Appendix B. □

6. EXPERIMENTS

In this article, we will compare the prediction performance with other methods on both the trust graph and rating graph. The competitive methods include average filling (AF); k-nearest neighbors (KNN) using Jaccard's coefficient, which is based on node similarities; SimRank [Jeh and Widom 2002], which is based on path ensembles; and SVD approximation [Billsus and Pazzani 1998] and matrix completion via trace norm (MC) [Candes and Recht 2012], which are based on the global graph structure.

We are going to give a brief description about MC since this is a relatively new technique in missing value imputation. MC seeks a lower-rank matrix, as SVD does. The key difference between MC and SVD is that MC tries to minimize the nuclear norm of the matrix (sum of singular values of the matrix); therefore, its *convex* objective function guarantees its global optimum solution. On the other hand, SVD is often stuck at the local optimum. MC is generally more robust to outliers than SVD. In this article, we stack the trust graph and rating graph using common users (movie titles) for the matrix completion method in this section in the form of $\mathbf{M} = [\mathbf{G}_1, \mathbf{G}_2]$; to be specific, it attempts to find X such that

$$\begin{aligned} &\min_{\mathbf{X}} \|\mathbf{X}\|_* \\ &s.t.\;\; \mathbf{X}_\Omega = \mathbf{M}_\Omega, \end{aligned}$$

where $\mathbf{M}_\Omega$ is the subset of the observed elements and $\|\mathbf{X}\|_*$ is the trace norm of X. Researchers also relax the constraints and optimize the following one:

$$\min_X \left\|\mathbf{X}_\Omega - \mathbf{M}_\Omega\right\|_F^2 + \varsigma \|X\|_*,$$

where ς is the regularity coefficient. This would serve as the benchmark transfer learning method in comparison to JSNM. It might be expected that the trust graph and rating graph also share their structures with this method; however, as we will demonstrate in the experiment part, such a naive idea does not work well.

For KNN, we search k in the list $\{1, 2, \ldots, 9\}$ to impute the missing value using the node with the highest Jaccard similarity score. For the SimRank method, we set the parameters using the default value suggested by the author. For SVD, we choose the rank from the list $(\frac{R}{10}, \frac{2R}{10} \ldots, R)$, where $R = \min(n, m)$, the minimum of the number of rows and columns. For MC, ς is tuned from the list $\{10^{-2}, 10^{-1}, 1, 10\}$.

6.1. Evaluations on Synthetic Data

In this part, we first do the experiments on a synthetic dataset, which consists of the MovieLens100K rating graph [Herlocker et al. 1999] and the synthetic trust graph we would construct.

MovieLens100K consists of 100,000 ratings (from 1 to 5) from 943 users on 1,682 movies; here each user rated at least 20 movies. Since this dataset has around 94% missing values, we first fill in the missing values with the mean of the available information in that row. Then we construct the Laplacian graph $\mathbf{W}$ based on users with parameter settings as follows: Euclidean distance as metric measure, heat kernel with scale parameter 5, and number of neighborhoods $k = 100$. After that, we normalize each column into an ℓ_2 unit vector. At last, we construct the trust graph $\mathbf{T}$ based on the threshold θ, which is set at 0.01, $\mathbf{T}(i, j) = 1$ if $\mathbf{W}(i, j) > \theta$, and 0 otherwise. Via the aforementioned setting, the two users get 1 mutually (trust each other) if their reviews on items are similar. We find by such procedure that the ratio of 1s in the trust graph is about 12%.

Due to the lack of ground truth for unobservable rating entries, we have to hide existing rating entries to simulate missing ones; here we randomly leave half of them available (about 3%) and mask half of them for the test. Since the trust graph is constructed from the Laplacian graph of the rating graph, our evaluation would be limited to the rating graph in this subsection. Note that the trust graph is constructed from very limited rating entries; nevertheless, we show that with the auxiliary trust information, the accuracy of rating graph imputation is better than classical imputation methods that explore the rating graph alone.

We adopt two evaluation metrics: Mean Absolute Error (MAE) and Root Mean Square Error (RMSE):

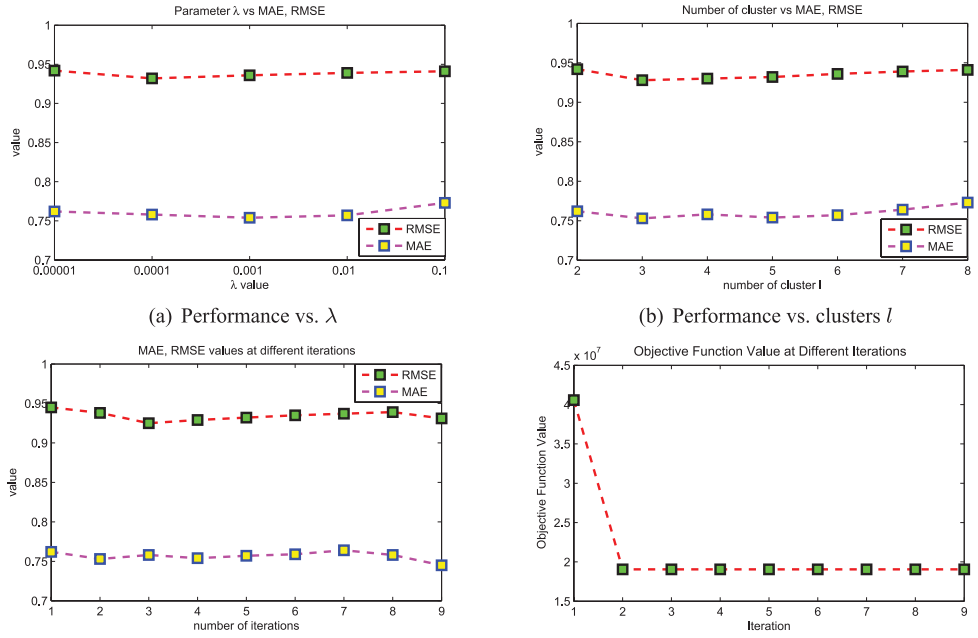
$$\begin{aligned} \text{MAE} &= \sum_{\mathbf{R}_{ij} \in T_E} |\mathbf{R}_{ij} - \hat{\mathbf{R}}_{ij}| / |T_E| \\ \text{RMSE} &= \sqrt{\sum_{\mathbf{R}_{ij} \in T_E} (\mathbf{R}_{ij} - \hat{\mathbf{R}}_{ij})^2 / |T_E|}, \end{aligned} \tag{24}$$

where $\mathbf{R}_{ij}$ and $\hat{\mathbf{R}}_{ij}$ are the true and predicted ratings, respectively, and $|T_E|$ is the number of test ratings. In all experiments, we run 10 random trials when generating the missing and observed ratings, use AF methods to initialize missing values, and do the imputation with all the methods in the twofold cross-validation process. The averaged results are reported in Table I.

It can be observed that our method consistently outperforms other methods in terms of MAE and RMSE, and it successfully incorporates the auxiliary information in the trust graph. We now want to investigate the influence of parameters for our method. First, we set $l = 3$ and maximum iteration $T = 20$ and vary the value of λ; the MAE and RMSE results are shown in Figure 2(a).

Table I. Prediction Result for MovieLens

Prediction Measure	Methods	Result
MAE	AF	0.802 ± 0.005
	KNN	0.812 ± 0.006
	SimRank	0.814 ± 0.006
	SVD	0.962 ± 0.007
	MC	0.826 ± 0.005
	JSNM	$\mathbf{0.745} \pm 0.004$
RMSE	AF	0.996 ± 0.004
	KNN	1.019 ± 0.004
	SimRank	1.024 ± 0.004
	SVD	1.183 ± 0.008
	MC	1.032 ± 0.004
	JSNM	$\mathbf{0.931} \pm 0.003$



(a) Performance vs. λ (b) Performance vs. clusters l

(c) Performance vs. iterations (d) Objective function value vs. iterations

Fig. 2. Investigation of parameters in our method.

Next, with $\lambda = 10^{-3}$ and $T = 20$, we plot the MAE and RMSE curves when the number of clusters l varies in Figure 2(b).

The MAE and RMSE results for each iteration have been displayed in Figure 2(c) with $l = 3$ and $\lambda = 10^{-3}$.

It can be observed that our method is generally robust to the choice of the parameter λ, the number of clusters, and maximum iterations. For the subsequent experiments, unless otherwise specified, we set $\lambda = 10^{-3}$, $l = 3$, and $T = 20$.

We provided the theoretical proof about the monotone decrease of our objective function in the preceding section. To give a concrete example, we also include the objective function value plot using the previously mentioned default setting. From Figure 2(d), we can observe that our objective function is very stable as the iteration increases.

This synthetic dataset demonstrates that with a synthetic auxiliary trust graph, our method has better performance than other classical methods. Our next real dataset shows that such transfer learning process is mutual beneficial; it improves the prediction for both the trust graph and the rating graph.

6.2. Evaluations on Real Data

In this part, we will compare our method with other methods on trust prediction using the Epinions dataset. This dataset was collected by Paolo Massa [Massa and Avesani 2009] in a five-week crawl from Epinions.com. It consists of two parts: one is the ratings part; the other is the trust part. The Epinions dataset consists of 49,290 users, 139,738 items, 664,824 reviews from users to items, and 487,181 trust statement between users. Users express their web of trust, that is, reviewers whose reviews and ratings they have consistently found to be valuable and offensive [Massa and Avesani 2009]. Therefore, it is reasonable to assume that most individual users tend to cast trust votes toward other users if the users have similar rating patterns toward those items. As a result, the rating matrix and trust matrix could have similar row structure given common users.

Inspired by the previous observation, we design the experiments as follows: we select the top 2,000 users with the highest degrees (who cast and receive the most votes), and then we select items with more than 68 ratings from the previously mentioned selected users. The resulting trust graph $\mathbf{G}_1$ of size $2{,}000 \times 2{,}000$ has 149,146 trust votes (represented by 1), which consists of 3.73% of all possible votes; those distrust or unknown votes are represented by 0. The rating graph $\mathbf{G}_2$ of size $2{,}000 \times 96$ has 10,225 ratings (from 1 to 5), which consists of 5.33% of all possible ratings; those missing ratings are represented by 0. Among those available ratings, the number of ratings 1, 2, and 3 are roughly equal; 4 is twice as many as 1; and 5 is about four times as many as 1; such a skewed distribution might be due to users' reluctance to give low ratings for unsatisfactory items.

Evaluation Metric. Since the binary trust votes have a very skewed distribution, precision and recall are more suitable than receiver operating characteristic (ROC) [Davis and Goadrich 2006]. The precision and recall of the evaluation metric are defined as follows:

$$recall = \frac{TP}{TP+FN}, \quad precision = \frac{TP}{TP+FP}$$
$$F1 = \frac{2 \times recall \times precision}{recall + precision}, \tag{25}$$

where TP, FN, and FP are numbers of true positives, false negatives, and false positives, respectively. Since the predicted values for the trust graph are generally not 0/1 integers for most methods, here we must decide the transformation criterion. The simplest one is probably the threshold method: if the predicted value is less than the threshold θ, we decide it is 0; otherwise, it is 1.

We still hide half of the available entries and conduct the prediction via twofold cross-validation as in the previous subsection. To evaluate the prediction result in a comprehensive manner, we calculate the recall and precision values for both trust and distrust, as $the\theta$ value varies from 0 to 1 with step 0.01. We can then compute the corresponding AUC values for all methods for both trust and distrust predictions together with the F1 score values. From Table II(a), JSNM has better performance than other methods except the F1 score for trust links, where AF shows some slight advantage. Note that it is impractical and time consuming to tune the threshold for real application; therefore, our method still shows better performance in trust link prediction than

Table II. Evaluation Result for Trust-Rating Dataset

(a) Recall-Precision Curve Evaluations

Link	Methods	AUC	F1
Trust	AF	0.207	**0.223**
	KNN	0.183	0.218
	SimRank	0.185	0.218
	SVD	0.123	0.160
	MC	0.075	0.122
	JSNM	**0.215**	0.221
Distrust	AF	0.914	0.977
	KNN	0.916	0.977
	SimRank	0.916	0.977
	SVD	0.583	0.971
	MC	0.972	0.981
	JSNM	**0.992**	**0.991**

(b) Rating Graph Evaluation Results

Prediction Measure	Methods	Result
MAE	AF	0.864 ± 0.003
	KNN	0.839 ± 0.004
	SimRank	0.832 ± 0.005
	SVD	0.924 ± 0.006
	MC	0.828 ± 0.005
	JSNM	$\mathbf{0.772} \pm 0.003$
RMSE	AF	1.062 ± 0.006
	KNN	1.045 ± 0.003
	SimRank	1.034 ± 0.003
	SVD	1.263 ± 0.012
	MC	1.024 ± 0.004
	JSNM	$\mathbf{0.963} \pm 0.004$

the AF method considering the significant AUC advantage. We can conclude that our method has the best performance in trust prediction in all the methods we listed in terms of trust links and distrust links. Table II(b) lists all methods' optimal value in terms of MAE and RMSE for the rating graph. Again JSNM has the best MAE and RMSE results. Based on Tables II(a) and II(b), we can conclude that transfer learning does provide the bridge for the trust graph and rating graph to share the valuable information with each other. This helps alleviate the common data sparsity issue in social network data. On the other hand, as we have shown, the naive transfer learning MC method does not work very well here; the MC method fails to extract the common row structure with matrices stacked.

6.3. Application to Homogeneous Dataset

The previous two datasets both deal with the trust graph and the rating graph, which are heterogeneous in terms of domain and scale. There are also cases in which homogeneous social graph inference is desired. One example is to predict user preferences on books and movies. In this subsection, we want to demonstrate that our framework also applies to such homogeneous-type data, two movie rating datasets. Note that for homogeneous datasets, we would drop the scale adjusting parameter c in objective Equation (2), which is a special case of our framework.

In our experiment, two movie rating datasets used are the Netflix training set and MovieLens [Herlocker et al. 1999]. The Netflix rating data contains more than 10^8 ratings with values from 1 to 5, which are given by around 500,000 users on around 20,000 movies. The MovieLens rating data contains more than 10^7 ratings with values from 1 to 5 and a scale of 0.5. We construct the dataset used in this experiment as follows: first we extract common movie titles that have at least 100 ratings in both datasets; after that we select the first 100 users each. Via this way, we get both matrices with size $1{,}381 \times 100$. Next, we randomly split available ratings into 20 parts. Each time preserving one part and masking all others,[3] we do the prediction and evaluate the performance of all methods. We calculate the average MAE and RMSE for these 20 experiments. Such process is repeated 10 times to calculate the mean and standard deviation.

From Tables III(a) and III(b), we can observe that our method still outperforms other methods in terms of MAE and RMSE. Meanwhile, the performance of all methods

[3]For performance purposes, we do the row sampling based on movies and ensure each movie has a few available ratings.

Table III. Netflix-MovieLens Evaluation

(a) Prediction Result for Netflix

Prediction Measure	Methods	Result
MAE	AF	0.942 ± 0.007
	KNN	0.928 ± 0.008
	SimRank	0.925 ± 0.008
	SVD	1.724 ± 0.014
	MC	0.983 ± 0.008
	JSNM	$\mathbf{0.903} \pm 0.005$
RMSE	AF	1.234 ± 0.005
	KNN	1.189 ± 0.005
	SimRank	1.164 ± 0.005
	SVD	1.924 ± 0.011
	MC	1.162 ± 0.006
	JSNM	$\mathbf{1.072} \pm 0.005$

(b) Prediction Result for MovieLens

Prediction Measure	Methods	Result
MAE	AF	0.802 ± 0.005
	KNN	0.812 ± 0.006
	SimRank	0.814 ± 0.006
	SVD	1.843 ± 0.007
	MC	1.045 ± 0.006
	JSNM	$\mathbf{0.764} \pm 0.004$
RMSE	AF	1.023 ± 0.004
	KNN	1.043 ± 0.004
	SimRank	1.037 ± 0.004
	SVD	2.046 ± 0.008
	MC	1.173 ± 0.005
	JSNM	$\mathbf{0.987} \pm 0.003$

Table IV. Manifold Term Investigation

(a) Trust Graph Evaluation for Epinion

Link	Methods	AUC	F1
Trust	JSNM	$\mathbf{0.214} \pm 0.003$	$\mathbf{0.220} \pm 0.002$
	DGF	0.209 ± 0.003	0.213 ± 0.004
Distrust	JSNM	$\mathbf{0.992} \pm 0.002$	$\mathbf{0.992} \pm 0.003$
	DGF	0.982 ± 0.004	0.984 ± 0.003

(b) Rating Graph Evaluation for Various Datasets

Data	Methods	MAE	RMSE
Epinion	JSNM	$\mathbf{0.772} \pm 0.004$	$\mathbf{0.963} \pm 0.005$
	DGF	0.802 ± 0.011	1.004 ± 0.007
Netflix	JSNM	$\mathbf{0.904} \pm 0.004$	$\mathbf{1.074} \pm 0.006$
	DGF	0.918 ± 0.009	1.136 ± 0.011
MovieLens	JSNM	$\mathbf{0.766} \pm 0.005$	$\mathbf{0.987} \pm 0.004$
	DGF	0.783 ± 0.009	1.012 ± 0.012

have decreased quite significantly compared with rating graph results in the previous subsection. One possible reason is that since these two matrices are now made up of common movies but different users, the ratings for any movie have more variability than the data in the proceeding section. We would conduct more investigations in our future research for such type of data.

6.4. Social Network Regularization Effect Investigation

In this section, we look into the effect of the manifold term in objective function Eq. (2). We compare the performance of our framework (JSNM) with the objective function without the manifold terms. We call the new method dual graph factorization (DGF). We still set $\lambda = 10^{-3}$ in JSNM and then repeat the same experiment procedure in the previous subsections. We summarize the results in Table IV. In terms of trust prediction evaluation, JSNM improves the DGF result 2% to 3% based on the DGF result for the trust link and 1% for the distrust link. As to the rating evaluation, JSNM improves 3.7% and 4% on Epinion for both MAE and RMSE, 1.2% and 1.5% on Netflix, 5% and 2% on MovieLens, respectively. From the table, we can conclude that the social network regularity term plays a role in our framework.

7. CONCLUSION

In this article, we developed the joint social network mining (JSNM) method to perform the trust prediction with the ancillary rating matrix. We transfer the common group structure knowledge between two related matrices and simultaneously explore the individual matrix geometric structure. With publicly available datasets, our method shows its advantage over classical trust prediction methods for both the trust matrix and rating matrix. Furthermore, our method can also be applied to homogeneous-type data and yield similar improvement in the prediction.

Although most websites do not have (publish) both trust graph and rating graph datasets, we believe our method provides many websites a new perspective to improve their service. Taking amazon.com and facebook.com for example, users may consent to information sharing between these two sites, as their friends lists and purchase histories generally cause no severe privacy leakage. Amazon may recommend users items their friends purchased to boost their sale; on the other hand, Facebook users could have the opportunity to link to other users and make new friends who purchased similar topics of books or styles of music and demonstrated same interest. In future work, we will investigate the effectiveness of our framework applying to more general related dual graphs.

APPENDIXES

A. PROOF OF THEOREM 5.4

PROOF. We rewrite Equation (20) as

$$\begin{aligned} J(\mathbf{V}_1) \;=\; & Tr\big(\lambda \mathbf{V}_1^T \mathbf{L}_1^+ \mathbf{V}_1 - \lambda \mathbf{V}_1^T \mathbf{L}_1^- \mathbf{V}_1 - 2\mathbf{G}_1^T \mathbf{U}^+ \mathbf{V}_1 \\ & + 2\mathbf{G}_1^T \mathbf{U}^- \mathbf{V}_1 + \boldsymbol{\alpha}^+ \mathbf{V}_1^T \mathbf{V}_1 - \boldsymbol{\alpha}^- \mathbf{V}_1^T \mathbf{V}_1\big). \end{aligned}$$

According to Lemma 5.2, we have

$$\begin{aligned} Tr\big(\mathbf{V}_1^T \mathbf{L}_1^+ \mathbf{V}_1\big) \;&\le\; \sum_{ij} \frac{(\mathbf{L}_1^+ \mathbf{V}_1')_{ij} \mathbf{V}_{1,ij}^2}{\mathbf{V}_{1,ij}'} \\ \boldsymbol{\alpha}^+ Tr\big(\mathbf{V}_1^T \mathbf{V}_1\big) \;&\le\; \boldsymbol{\alpha}^+ \sum_{ij} \mathbf{V}_{1,ij}^2. \end{aligned}$$

Meanwhile, by the inequality $x \le \frac{(x^2+y^2)}{2y}, \forall y > 0$, we have

$$\begin{aligned} Tr\big(\mathbf{G}_1^T \mathbf{U}^- \mathbf{V}_1^T\big) \;&=\; \sum_{i,j} \big(\mathbf{G}_1^T \mathbf{U}^-\big)_{ij} V_{1,ij} \\ &\le\; \sum_{i,j} \big(\mathbf{G}_1^T \mathbf{U}^-\big)_{ij} \frac{\mathbf{V}_{1,ij}^2 + \mathbf{V}_{1,ij}'^2}{\mathbf{V}_{1,ij}'}. \end{aligned}$$

On the other hand, to get the lower bound for the remaining terms, we employ the inequality $z \ge 1 + \log z, \forall z > 0$, and then

$$\begin{aligned} Tr\big(\mathbf{G}_1^T \mathbf{U}^+ \mathbf{V}_1^T\big) \;&\ge\; \sum_{ij} \big(\mathbf{G}_1^T \mathbf{U}^+\big) \mathbf{V}_{1,ij}' \left(1 + \log \frac{\mathbf{V}_{1,ij}}{\mathbf{V}_{1,ij}'}\right) \\ Tr\big(\mathbf{V}_1^T \mathbf{L}_1^- \mathbf{V}_1\big) \;&\ge\; \sum_{ijk} (\mathbf{L}_1^-)_{jk} \mathbf{V}_{ji}' \mathbf{V}_{ki}' \left(1 + \log \frac{\mathbf{V}_{1,ji} \mathbf{V}_{1,ki}}{\mathbf{V}_{1,ji}' \mathbf{V}_{1,ki}'}\right) \\ Tr\big(\mathbf{V}_1^T \mathbf{V}_1\big) \;&\ge\; \sum_{ijk} \mathbf{V}_{1,ij}' \mathbf{V}_{1,ik}' \left(1 + \log \frac{\mathbf{V}_{1,ij} \mathbf{V}_{1,ik}}{\mathbf{V}_{1,ij}' \mathbf{V}_{1,ik}'}\right). \end{aligned}$$

By summing over all the bounds, we get $Z(\mathbf{V}_1, \mathbf{V}_1')$ and it is easy to conclude that

$$Z(\mathbf{V}_1, \mathbf{V}_1') \ge J(\mathbf{V}_1), \quad Z(\mathbf{V}_1, \mathbf{V}_1) = J(\mathbf{V}_1).$$

To find the minimum of $Z(\mathbf{V}_1, \mathbf{V}'_1)$, we take derivative with respect to $\mathbf{V}_{1,ij}$,

$$\begin{aligned}\frac{\partial Z(\mathbf{V}_1, \mathbf{V}'_1)}{\partial \mathbf{V}_{1,ij}} &= 2\lambda \frac{(\mathbf{L}_1^+ \mathbf{V}'_1)_{ij} \mathbf{V}_{1,ij}}{\mathbf{V}'_{1,ij}} - 2\lambda (\mathbf{L}_1^- \mathbf{V}'_1)_{ij} \frac{\mathbf{V}'_{1,ij}}{\mathbf{V}_{1,ij}} \\ &\quad - 2\big(\mathbf{G}_1^T \mathbf{U}^+\big)_{ij} \frac{\mathbf{V}'_{1,ij}}{\mathbf{V}_{1,ij}} + 2\big(\mathbf{G}_1^T \mathbf{U}^-\big)_{ij} \frac{\mathbf{V}_{1,ij}}{\mathbf{V}'_{1,ij}} \\ &\quad + 2\boldsymbol{\alpha}^+ \mathbf{V}_{1,ij} - 2\boldsymbol{\alpha}^- \frac{\mathbf{V}'_{1,ij}}{\mathbf{V}_{1,ij}},\end{aligned}$$

and the Hessian matrix of $Z(\mathbf{V}_1, \mathbf{V}'_1)$,

$$\begin{aligned}\frac{\partial^2 Z(\mathbf{V}_1, \mathbf{V}'_1)}{\partial \mathbf{V}_{1,ij} \partial \mathbf{V}_{1,kl}} &= \delta_{ik}\delta_{jl}\Bigg(2\lambda \frac{(\mathbf{L}_1^+ \mathbf{V}'_1)_{ij}}{\mathbf{V}'_{1,ij}} + 2\lambda (\mathbf{L}_1^- \mathbf{V}'_1)_{ij} \frac{\mathbf{V}'_{1,ij}}{\mathbf{V}^2_{1,ij}} \\ &\quad + 2\big(\mathbf{G}_1^T \mathbf{U}^+\big)_{ij} \frac{\mathbf{V}'_{1,ij}}{\mathbf{V}^2_{1,ij}} + 2\frac{(\mathbf{G}_1^T U^-)_{ij}}{\mathbf{V}'_{1,ij}} \\ &\quad + 2\boldsymbol{\alpha}^+ + 2\boldsymbol{\alpha}^- \frac{\mathbf{V}'_{1,ij}}{\mathbf{V}^2_{1,ij}}\Bigg),\end{aligned}$$

is a diagonal matrix with positive elements due to $\mathbf{V}_1$ initialization and update rule; δ_{ik} is the delta function, $\delta_{ik} = 1$ if $i = k$, and 0 otherwise. Therefore, $Z(\mathbf{V}_1, \mathbf{V}'_1)$ is a convex function of $\mathbf{V}_1$, and we can obtain the global minimum of $Z(\mathbf{V}_1, \mathbf{V}'_1)$ by setting $\frac{\partial Z(\mathbf{V}_1, \mathbf{V}'_1)}{\partial \mathbf{V}'_{1,ij}} = 0$ and solving for $\mathbf{V}_1$, and we can get Equation (21). □

B. PROOF OF THEOREM 5.6

PROOF. We rewrite Equation (22) as

$$\begin{aligned}J(\mathbf{V}_2) &= Tr\big(\lambda \mathbf{V}_2^T \mathbf{L}_2^+ \mathbf{V}_2 - \lambda \mathbf{V}_2^T \mathbf{L}_2^- \mathbf{V}_2 - 2c\, \mathbf{G}_2^T \mathbf{U}^+ \mathbf{V}_2 \\ &\quad + 2c\, \mathbf{G}_2^T \mathbf{U}^- \mathbf{V}_2 + \boldsymbol{\beta}^+ \mathbf{V}_2^T \mathbf{V}_2 - \boldsymbol{\beta}^- \mathbf{V}_2^T \mathbf{V}_2\big).\end{aligned}$$

According to Lemma 5.2, we have

$$\begin{aligned}Tr\big(\mathbf{V}_2^T \mathbf{L}_2^+ \mathbf{V}_2\big) &\le \sum_{ij} \frac{(\mathbf{L}_2^+ \mathbf{V}'_2)_{ij} \mathbf{V}^2_{2,ij}}{\mathbf{V}'_{2,ij}} \\ \boldsymbol{\beta}^+ Tr\big(\mathbf{V}_2^T \mathbf{V}_2\big) &\le \boldsymbol{\beta}^+ \sum_{ij} \mathbf{V}^2_{2,ij}.\end{aligned}$$

Meanwhile, by the inequality $x \le \frac{(x^2+y^2)}{2y}$, $\forall y > 0$, we have

$$\begin{aligned}Tr\big(\mathbf{G}_2^T \mathbf{U}^- \mathbf{V}_2^T\big) &= \sum_{i,j} \big(\mathbf{G}_2^T \mathbf{U}^-\big)_{ij} V_{2,ij} \\ &\le \sum_{i,j} \big(\mathbf{G}_2^T \mathbf{U}^-\big)_{ij} \frac{\mathbf{V}^2_{2,ij} + \mathbf{V}'^2_{2,ij}}{\mathbf{V}'_{2,ij}}.\end{aligned}$$

On the other hand, to get the lower bound for the remaining terms, we employ the inequality $z \geq 1 + \log z, \forall z > 0$, and then

$$Tr(\mathbf{G}_2^T \mathbf{U}^+ \mathbf{V}_2^T) \geq \sum_{ij} (\mathbf{G}_2^T \mathbf{U}^+) \mathbf{V}'_{2,ij} \left(1 + \log \frac{\mathbf{V}_{2,ij}}{\mathbf{V}'_{2,ij}}\right)$$

$$Tr(\mathbf{V}_2^T \mathbf{L}_2^- \mathbf{V}_2) \geq \sum_{ijk} (\mathbf{L}_2^-)_{jk} \mathbf{V}'_{ji} \mathbf{V}'_{ki} \left(1 + \log \frac{\mathbf{V}_{2,ji} \mathbf{V}_{2,ki}}{\mathbf{V}'_{2,ji} \mathbf{V}'_{2,ki}}\right)$$

$$Tr(\mathbf{V}_2^T \mathbf{V}_2) \geq \sum_{ijk} \mathbf{V}'_{2,ij} \mathbf{V}'_{2,ik} \left(1 + \log \frac{\mathbf{V}_{2,ij} \mathbf{V}_{2,ik}}{\mathbf{V}'_{2,ij} \mathbf{V}'_{2,ik}}\right).$$

By summing over all the bounds, we get $Z(\mathbf{V}_2, \mathbf{V}'_2)$ and it is easy to conclude that

$$Z(\mathbf{V}_2, \mathbf{V}'_2) \geq J(\mathbf{V}_2), \quad Z(\mathbf{V}_2, \mathbf{V}_2) = J(\mathbf{V}_2).$$

To find the minimum of $Z(\mathbf{V}_2, \mathbf{V}'_2)$, we take derivative with respect to $\mathbf{V}_{2,ij}$,

$$\begin{aligned}\frac{\partial Z(\mathbf{V}_2, \mathbf{V}'_2)}{\partial \mathbf{V}_{2,ij}} = {} & 2\lambda \frac{(\mathbf{L}_2^+ \mathbf{V}'_2)_{ij} \mathbf{V}_{2,ij}}{\mathbf{V}'_{2,ij}} - 2\lambda (\mathbf{L}_2^- \mathbf{V}'_2)_{ij} \frac{\mathbf{V}'_{2,ij}}{\mathbf{V}_{2,ij}} \\ & - 2\big(c\, \mathbf{G}_2^T \mathbf{U}^+\big)_{ij} \frac{\mathbf{V}'_{2,ij}}{\mathbf{V}_{2,ij}} + 2\big(c\, \mathbf{G}_2^T \mathbf{U}^-\big)_{ij} \frac{\mathbf{V}_{2,ij}}{\mathbf{V}'_{2,ij}} + 2\boldsymbol{\beta}^+ \mathbf{V}_{2,ij} - 2\boldsymbol{\beta}^- \frac{\mathbf{V}'_{2,ij}}{\mathbf{V}_{2,ij}},\end{aligned}$$

and the Hessian matrix of $Z(\mathbf{V}_2, \mathbf{V}'_2)$,

$$\begin{aligned}\frac{\partial^2 Z(\mathbf{V}_2, \mathbf{V}'_2)}{\partial \mathbf{V}_{2,ij} \partial \mathbf{V}_{2,kl}} = {} & \delta_{ik} \delta_{jl} \Bigg(2\lambda \frac{(\mathbf{L}_2^+ \mathbf{V}'_2)_{ij}}{\mathbf{V}'_{2,ij}} + 2\lambda (\mathbf{L}_2^- \mathbf{V}'_2)_{ij} \frac{\mathbf{V}'_{2,ij}}{\mathbf{V}^2_{2,ij}} \\ & + 2\big(c\, \mathbf{G}_2^T \mathbf{U}^+\big)_{ij} \frac{\mathbf{V}'_{2,ij}}{\mathbf{V}^2_{2,ij}} + 2\frac{\big(c\, \mathbf{G}_2^T U^-\big)_{ij}}{\mathbf{V}'_{2,ij}} + 2\boldsymbol{\beta}^+ + 2\boldsymbol{\beta}^- \frac{\mathbf{V}'_{2,ij}}{\mathbf{V}^2_{2,ij}}\Bigg),\end{aligned}$$

is a diagonal matrix with positive elements due to $\mathbf{V}_2$ initialization and update rule; δ_{ik} is the delta function, $\delta_{ik} = 1$ if $i = k$, and 0 otherwise. Therefore, $Z(\mathbf{V}_2, \mathbf{V}'_2)$ is a convex function of $\mathbf{V}_2$, and we can obtain the global minimum of $Z(\mathbf{V}_2, \mathbf{V}'_2)$ by setting $\frac{\partial Z(\mathbf{V}_2, \mathbf{V}'_2)}{\partial \mathbf{V}'_{2,ij}} = 0$ and solving for $\mathbf{V}_2$, and we can get Equation (23). □

ACKNOWLEDGMENTS

The authors would like to thank the authors of Epinions and Movielens for graciously making data available for this study. We also appreciate the data from Netflix. This research was partially supported by NSF IIS-1117965. Y. Tu is supported by NIH grant R01GM086707.

REFERENCES

Bedi, P., Kaur, H., and Marwaha, S. 2007. Trust based recommender system for semantic web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, San Francisco, CA, 2677–2682.

Billsus, D. and Pazzani, M. J. 1998. Learning collaborative information filters. In *Proceedings of the 15th International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 46–54.

Boyd, S. and Vandenberghe, L. 2004. *Convex Optimization*. Cambridge University Press, Cambridge.

Cai, D., He, X., Wu, X., and Han, J. 2008. Non-negative factorization on manifold. In *Proceedings of the 8th IEEE International Conference on Data Mining*. IEEE, Los Alamitos, CA, 63–72.

Candes, E. and Recht, B. 2012. Exact matrix completion via convex optimization. *Comm. ACM 55,* 6, 111–119.

Cao, B., Liu, N., and Yang, Q. 2010. Transfer learning for collective link prediction in multiple heterogenous domains. In *Proceedings of the 27th International Conference on Machine Learning*. ACM, New York, 159–166.

Chapelle, O., Scholkopf, B., and Zien, A. 2006. *Semi-Supervised Learning*. MIT Press, Cambridge, MA.

Crandall, D., Cosley, D., Huttenlocher, D., Kleinberg, J., and Suri, S. 2008. Feedback effects between similarity and social influence in online communities. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, 160–168.

Davis, J. and Goadrich, M. 2006. The relationship between precision-recall and roc curves. In *Proceedings of the 15th International Conference on Machine Learning*. ACM, New York, 233–240.

Ding, C., Li, T., and Jordan, M. 2010. Convex and semi-nonnegative matrix factorization. *IEEE Trans. Pattern Anal. Mach. Intell. 32,* 1, 45–55.

Getoor, L. and Diehl, C. 2005. Link mining: A survey. *ACM SIGKDD Explorations Newsletter 7,* 2, 3–12.

Guha, R., Kumar, R., Raghavan, P., and Tomkins, A. 2004. Propagation of trust and distrust. In *Proceedings of 11th International Conference on World Wide Web*. ACM, New York, 403–412.

He, X. and Niyogi, P. 2003. Locality preserving projections. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 153–160.

Herlocker, J., Konstan, J., Borchers, A., and Riedl, J. 1999. An algorithmic framework for performing collaborative filtering. In *Proceedings of the Conference on Research and Development in Information Retrieval*. ACM, New York, 230–237.

Hofmann, T. and Puzicha, J. 1999. Latent class models for collaborative filtering. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*. ACM, New York, 688–693.

Huang, J., Nie, F., Huang, H., and Tu, Y.-C. 2012. Trust prediction via aggregating heterogeneous social networks. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. ACM, New York, NY, 1774–1778.

Jeh, G. and Widom, J. 2002. A measure of structural-context similarity. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, 538–543.

Kamvar, S., Schlosser, M., and Garcia-Molina, H. 2003. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th International Conference on World Wide Web*. ACM, New York.

Lee, D. and Seung, H. 2000. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 556–562.

Leskovec, J., Huttenlocher, D., and Kleinberg, J. 2010. Predicting positive and negative links in online social networks. In *Proceedings of the 19th International Conference on World Wide Web*. ACM, New York.

Liben-Nowell, D. and Kleinberg, J. 2003. The link prediction problem for social networks. In *Proceedings of the 12th International Conference on Information and Knowledge Management*. ACM, New York, 556–559.

Massa, P. and Avesani, P. 2007. Trust-aware recommender systems. In *Proceedings of the 2007 ACM Conference on Recommender Systems*. ACM, New York, 17–24.

Massa, P. and Avesani, P. 2009. Trust metrics in recommender systems. *Comput. Social Trust*, 259–285.

McPherson, M., Smith-Lovin, L., and Cook, J. 2001. Birds of a feather: Homophily in social networks. *Ann. Rev. Sociol. 27*, 415–444.

Mislove, A., Viswanath, B., Gummadi, P., and Druschel, P. 2010. You are who you know: Inferring user profiles in online social networks. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*. ACM, New York, NY, 251–260.

Ng, A., Jordan, M., and Weiss, Y. 2001. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 849–856.

Pan, W., Xiang, W., Liu, N., and Yang, Q. 2010. Transfer learning in collaborative filtering for sparsity reduction. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. AAAI Press, Palo Alto, CA, 230–235.

Roweis, S. and Saul, L. K. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science 290,* 5500, 2323–2326.

Salakhutdinov, R. and Mnih, A. 2007. Probabilitistic matrix factorization. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 1257–1264.

Salakhutdinov, R. and Mnih, A. 2008. Bayesian probabilistic matrix factorization using Marknov chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine learning*. ACM, New York, 880–887.

SARWAR, B., KARYPIS, G., KONSTAN, J., AND RIEDL, J. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*. ACM, New York, NY, 285–295.

SHI, J. AND MALIK, J. 2000. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell. 22,* 8, 888–905.

SINGH, A. AND GORDON, G. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, 650–658.

SREBRO, N. AND JAAKKOLA, T. 2003. Weighted low-rank approximations. In *Proceedings of the 20th Annual International Conference on Machine Learning*. AAAI Press, Palo Alto, CA, 720–727.

VON LUXBURG, U. 2006. A tutorial on spectral clustering. *Stat. Comput. 17,* 4, 395–416.

WEN, Z. AND LIN, C. 2010. On the quality of inferring interests from social neighbors. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, 373–382.

XU, Z., KERSTING, K., AND TRESP, V. 2009. Multi-relational learning with gaussian process. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, San Francisco, CA, 1309–1314.

YU, K. AND CHU, W. 2007. Gaussian process models for link analysis and transfer learning. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 1657–1664.

YU, K., CHU, W., YU, S., TRESP, V., AND ZHAO, X. 2006. Stochastic relational models for discriminative link prediction. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 333–340.

YU, K., LAFFERTY, J., ZHU, S., AND GONG, Y. 2007. Large-scale collaborative prediction using a nonparametric random effects model. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, New York, 1185–1192.

YUSTER, R. AND ZWICK, U. 2005. Fast sparse matrix multiplication. *ACM Trans. Algorithms 1,* 1, 2–13.

ZHU, S., YU, K., CHI, Y., AND GONG, Y. 2007. Combining content and link for classification using matrix factorization. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, 487–494.

Received August 2012; revised November 2012; accepted March 2013