Transactions on Knowledge Discovery in Data

Semi-supervised Learning Meets Factorization: Learning to Recommend with Chain Graph Model

Journal:	*Transactions on Knowledge Discovery from Data*
Manuscript ID	TKDD-2017-07-0127
Manuscript Type:	Regular full paper submission
Date Submitted by the Author:	21-Jul-2017
Complete List of Authors:	Chen, Chaochao; Zhejiang University; University of Illinois at Urbana-Champaign Chang, Kevin; University of Illinois at Urbana-Champaign Li, Qibing; Zhejiang University ZHENG, Xiaolin; Zhejiang University, College of Computer Science
Keywords:	Semi-supervised Learning, Latent Factor Model, Chain Graph Model, Data Sparsity

Semi-supervised Learning Meets Factorization: Learning to Recommend with Chain Graph Model

CHAOCHAO CHEN, Zhejiang University and University of Illinois at Urbana-Champaign
KEVIN CHEN-CHUAN CHANG, University of Illinois at Urbana-Champaign
QIBING LI and XIAOLIN ZHENG*, Zhejiang University

Recently *latent factor model* (LFM) has been drawing much attention in recommender systems due to its good performance and scalability. However, existing LFMs predict missing values in a user-item rating matrix only based on the known ones, and thus the *sparsity* of the rating matrix always limits their performance. Meanwhile, *semi-supervised learning* (SSL) provides an effective way to alleviate the label (i.e., rating) sparsity problem by performing label propagation, which is mainly based on the *smoothness* insight on affinity graphs. However, graph-based SSL suffers serious scalability and graph unreliable problems when directly being applied to do recommendation. In this paper, we propose a novel probabilistic *chain graph model* (CGM) to marry SSL with LFM. The proposed CGM is a combination of *Bayesian network* and *Markov random field*. The Bayesian network is used to model the rating generation and regression procedures, and the Markov random field is used to model the confidence-aware smoothness constrain between the generated ratings. Experimental results show that our proposed CGM significantly outperforms the state-of-the-art approaches in terms of four evaluation metrics, and with a larger performance margin when data sparsity increases.

CCS Concepts: • **Information systems** → **Recommender systems**; *Probabilistic retrieval models*;

Additional Key Words and Phrases: Semi-supervised Learning, Latent Factor Model, Chain Graph Model, Data Sparsity

ACM Reference format:
Chaochao Chen, Kevin Chen-Chuan Chang, Qibing Li, and Xiaolin Zheng*. 2017. Semi-supervised Learning Meets Factorization: Learning to Recommend with Chain Graph Model. *ACM Trans. Knowl. Discov. Data.* 1, 1, Article 1 (July 2017), 23 pages.
https://doi.org/0000001.0000001

1 INTRODUCTION

As e-commerce platforms and online social networks provide customers and users myriad products and massive information, a *recommender system* (RS) has become a useful and even indispensable facility for information filtering. In fact, RS is now part of our daily activities, as it has been widely adopted by internet service leaders, such as Amazon, Youtube, and Facebook [35].

[*] Corresponding author.

This work was supported in part by the National Natural Science Foundation of China (Nos. 61379034 and U1509221), and the National Key Technology R&D Program (Nos. 2014BAH28F05 and 2015BAH07F01).

1556-4681/2017/7-ART1 $15.00
https://doi.org/0000001.0000001

Data sparsity challenge. As *users* have actions (e.g., rate and buy) on some *items*, RS aims to predict the users' unknown actions on other items. The tendency of a user's action on an item can be indicated by a real-valued number, i.e., *rating* or *label*. Thus, RS is also known as the *unknown ratings prediction* problem [37]. In practice, however, many RSs have to evaluate very large user and item sets, where the user-item (U-I) matrix is extremely sparse– such *data sparsity* has always been the main challenge of RS [38].

So far, two techniques have been popularly used to alleviate the data sparsity problem of RS, i.e., *semi-supervised learning* (SSL) and *latent factor model* (LFM).

Semi-supervised learning. SSL uses unlabeled data to either modify or reprioritize hypotheses obtained from labeled data alone, and thus can alleviate the label sparsity problem [36]. Towards effective SSL, affinity graph-based smoothness approaches have attracted much research interests, which follow the *smoothness* insight: *close nodes on an affinity graph have similar labels*. Graph-based SSL is appealing recently because it is easy to implement and gives rise to closed-form solutions [8, 10, 13, 41, 42, 46].

Latent factor model. LFM increases the rating density by reducing the dimensionality of U-I rating matrix, and thus is able to alleviate the data sparsity problem. LFM uses a low dimensional user and item latent factors to represent the characteristics of each user and each item, and uses the product of them to represent the user's rating on the item. LFM has drawn much attention recently due to its good performance and scalability [1, 4, 16, 21, 23, 25, 30, 31, 33, 39, 40, 43].

Although SSL and LFM have their own merits to alleviate the data sparsity problem, they have their own disadvantages. Graph-based SSL directly predicts the unknown ratings in the original U-I matrix, and thus suffers from the scalability problem. Meanwhile, LFM focuses on regressing the rare labelled data, and fails to capture the smoothness insight between ratings, unlike SSL, and thus its performance is further limited under data sparsity scenario [1].

As a key insight of this paper, we identify the marriage of SSL and LFM. The main insights of SSL (i.e., *smoothness*) and LFM (i.e., *dimensionality reduction*) are orthogonal, and they are likely to benefit from each other. However, surprisingly, the synergy between SSL and LFM has not been explored. On one hand, the smoothness idea of SSL, i.e., similar users or items should give or receive similar ratings– across the entire U-I matrix of ratings, known or unknown, can mitigate the data sparsity problem of LFM. On the other hand, the dimensionality reduction idea of LFM can solve the scalability problem of SSL, since the prediction is done in a low rank matrix instead of the original high dimensional matrix.

Challenges of marrying SSL with LFM. However, marrying SSL with LFM to do recommendation is nontrivial. We summarize the main challenges as follows:

Challenge $\mathcal{I}$: Disparate unification. SSL predicts unknown ratings through rating propagation on affinity graphs. I.e., SSL captures the correlation dependency between ratings. LFM predicts unknown ratings through learning user and item latent factors by regressing known ratings. I.e., LFM captures the causal dependency between latent factors and ratings. Therefore, we aim to propose a principled framework to unify SSL and LFM so that such different dependency between different variables can be captured.

Challenge $\mathcal{I}\mathcal{I}$: Expensive graph construction. In a RS scenario, each data object (u_i, v_j) is a U-I pair, with a rating R_{ij} as its label, and thus, to adopt SSL for rating prediction in RS, rating smoothness should exist among U-I pairs. Suppose we have I users and $\mathcal{J}$ items, building such a U-I pairwise affinity graph needs to compute the similarity between each of $I \times J$ U-I pairs, with a time complexity of $O(I \times J)^2$, and thus, cannot scale to large datasets. Thus, we aim to realize smoothness in a more efficient manner.

Challenge III: Unreliable affinity. In a traditional SSL scenario, the affinity graphs are built based on the characteristics of the data itself, e.g., the pixel data of a scanned digit. In contrast, in a RS scenario, the affinity graphs are usually built based on user social relationships or ratings [9, 10, 15, 26, 31, 41, 42]. As a result, the affinity graphs are unreliable due to the sparsity of such user social relationships or ratings. Thus, we aim to alleviate the unreliable affinity problem in a robust way.

Our proposal: Learning to recommend with CGM. Our proposal is based on the following insights.

Insight I: Principled unification. To address Challenge I, we propose a novel chain graph model (CGM) to marry SSL with LFM. As far as we know, this is the first attempt in the literature to adopt CGM in RS. A CGM is a combination of Bayesian network and Markov random field, which has the ability to capture different kinds of dependency (i.e., correlation and causal dependency) between different kinds of variables (i.e., latent factors and ratings) [5, 24, 27]. Thus, CGM is an ideal model to solve Challenge I.

Insight II: Efficient smoothness. To address Challenge II, we develop a novel "joint smoothness" framework to realize SSL on a pair of decomposed user and item affinity graphs. Since a rating is given from a user to an item, smoothness exists in two dimensions, i.e., user and item. We term it *joint-smoothness*, which enables us to decrease the time complexity to build affinitive graphs to $O(I^2 + J^2)$.

Insight III: Selective smoothness. To address Challenge III, we propose a confidence-aware smoothness approach. Different from the traditional SSL which performs rating propagation (i.e., smoothness constraint) everywhere on the affinity graphs, we choose to perform smoothness selectively. Specifically, we propose a smoothness confidence decay model to control the hops of rating propagation length.

Solutions. First, we propose a novel CGM (Section 4), which is a combination of Bayesian network and Markov random field. Second, we propose a joint-smoothness objective function. Instead of building a U-I pairwise affinity graph, we build two decomposed user and item affinity graphs. Third, we propose a confidence-aware smoothness approach (Section 5.3). This selective smoothness approach not only alleviates the graph unreliable problem in RS scenarios, but also saves huge computation. Finally, we present the model learning method based on coordinate descent (Section 6).

Results. We concretely realize our solutions of marrying SSL with two kinds of popular LFMs, and conduct comprehensive experiments in Section 7. Our experiments are conducted on *three* popular datasets, with *nine* state-of-the-art comparison approaches. We use *four* metrics to evaluate model performance. The experimental results show that our approach significantly outperforms the state-of-the-art methods, especially in data sparsity scenarios.

Contributions. We summarize the main contributions in this paper as follows:

- We propose a novel CGM to marry SSL with LFM for alleviating the data sparsity problem of RS, which we believe is the first attempt in the literature.
- We propose to perform joint-smoothness instead of pairwise smoothness, which has better efficiency.
- We propose a confidence-aware smoothness approach to alleviate the unreliable graph problem in RS scenario. To the best of our knowledge, it is also the first attempt.
- Our model scales linearly with the observed data size, since we adopt dimensionality reduction technique and confidence-aware smoothness approach.

The rest of the paper is organized as follows. In Section 2, we review related work. In Section 3, we describe the popular realizations of SSL and LFM. In Section 4, we propose a novel probabilistic CGM to marry SSL with LFM. In Section 4, we present the confidence-aware joint-smoothness energy function. In Section 6, we present the model learning approach based on gradient descent. In Section 7, we present the experimental results and analysis. Finally, we conclude the paper in Section 8.

2 RELATED WORK

As described in Section 1, semi-supervised learning (SSL) and latent factor model ïijĹLFMïijĽ both have their advantages and shortcomings when making rating prediction. LFR attempts to adopt the idea of SSL in LFM, but it can not deal with Challenges $\mathcal{I}$, $\mathcal{II}$, and $\mathcal{III}$. Since we want to marry SSL with LFM by using chain graph model (CGM), in this section, we review literature on SSL, LFM, LFR, and CGM, respectively.

Semi-supervised learning. Graph-based SSL is proposed to alleviate data sparsity problem by performing label propagation on the affinity graphs [36], and its main insight is graph-based smoothness [13, 46]. In the literature, several different objective functions are proposed to realize graph-based smoothness, e.g., Harmonic Function (HF) [46] and Green's Function (GF) [10]. Take HF for example, it minimizes the rating difference between close nodes and thus achieves smoothness on $\mathcal{G}$, which is the same as propagate the known labels to the unknown ones on the affinity graph [45]. So far, there are several research directly adopting SSL to do rating prediction [10, 41, 42]. However, as described in Section 1, directly adopting them in RS suffers from scalability and unreliable affinity problems.

Latent factor model. Generally, existing LFMs can be divided into two types: basic matrix factorization (MF) that only uses U-I rating matrix to do prediction, and side information aided MF that uses other side information besides the U-I rating matrix. Here, we introduce both of them.

Basic matrix factorization. MF has drawn much attention recently since it was adopted in the Netflix competition [23], and its main insight is the dimensionality reduction technique [34]. The most basic MF model, known as probabilistic matrix factorization (PMF) or single value decomposition (SVD), factorizes a U-I matrix into a low rank user feature matrix and item feature matrix, and then uses their product to predict unknown ratings [4, 14, 18, 25, 30, 39, 43]. Other promotions of SVD include SVDB and SVD++ [23].

Matrix factorization with side information. Generally, side information can be divided into three categories: content information, social information, and other context information, e.g., user attributes. These three kinds of information have been proven efficient to improve recommendation performance [1, 2, 2, 9, 16, 19, 20, 28, 33, 40, 44].

First, regression-based latent factor model (RLFM) [1] and factorization machines [33] were proposed to incorporate context information to improve recommendation performance. However, these context information are not always available and hard to obtain, and thus they are out of the scope of this paper. Second, fLDA [2] further incorporates item content information into RLFM Third, Wang et al. propose collaborative topic regression (CTR) [40], which systematically combines PMF and latent Dirichlet allocation (LDA) [6]. CTR uses PMF to factorize U-I rating information and uses LDA to mine item content information. It has been proven that CTR outperforms fLDA in a similar setting, since fLDA largely ignores the other users ratings [40]. Later, CTR-SMF [31] was proposed to factorize not only rating information but also user social information to make a better recommendation.

However, existing LFMs only fit the model by minimizing the difference between the rare known ratings and the predicted ratings, and do not consider rating smoothness nature between similar U-I pairs.

Latent factor restriction. The most similar works to ours are LFRs [9, 15, 26, 32]. They first use rating or side information to build user or item affinity graphs, and then constrain latent factor smoothness on the graphs. They assume that connected user or item on the affinity graphs should have similar latent factors. We divide the existing LFRs into two types, i.e, user latent factor restriction (ULFR) [9, 26] and user-item latent factor restriction (UILFR) [15], which means that only user latent factor and both user and item latent factors are restricted, respectively.

LFR is actually *not* the smoothness insight as captured in SSL, i.e., rating smoothness. LFR is much stronger than rating smoothness constrain, because LFR leads to rating smoothness, but not the other way around. Take an item affinity graph for example: based on the assumption of LFR, all the ratings of the connected items should be similar, since neighbors have similar latent factors. That is, LFR indicates smoothness exists everywhere on an affinity graph. Our experiments in Section 7 will demonstrate that this overly strong assumption will fail particularly in data sparsity scenarios.

Chain Graph Models. CGM is a probabilistic model that combines *Bayesian network* and *Markov random field*. Bayesian network is useful to express causal relationships between random variables [5], and it is popularly used in the existing LFMs [1, 30, 40], which use Bayesian network to express the rating generation procedure. Markov random field is suited to express soft constraints between random variables [5], and its applications also include recommender systems [12]. Chain graph models contain both Bayesian network and Markov random field, and thus can represent a broader class of distributions, and it has been used in applications include image de-noising [27], while, surprisingly, not in RS so far.

In this paper, we propose a novel CGM to marry LFM with SSL and combine the merits of both of them, and we also propose a confidence-aware approach to solve the overly strong assumption of LFR.

3 PRELIMINARY

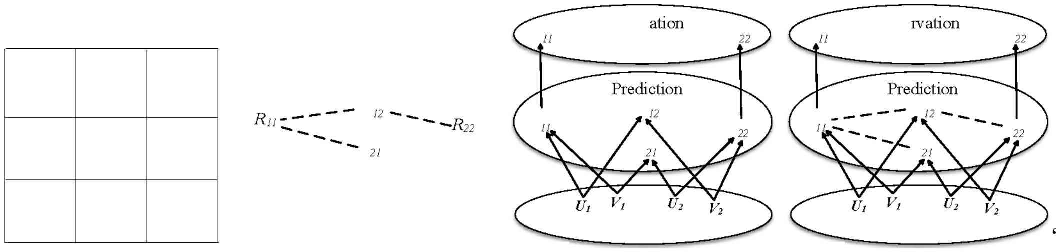
Since both SSL and LFM can be used to alleviate the data sparsity problem of RS, in this section, we will present the popular approaches of both of them.

Problem setting. In RS, each data object, i.e., a U-I pair, is related to two elements, i.e., user and item. The data label, i.e., U-I rating, is generated from a user to an item. Suppose there is a U-I *pairwise affinity graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ denote data objects, edges $\mathcal{E}$ denote the affinitive relationship between nodes, and weights of edges P denote the affinitive relation strength between nodes. Let $\boldsymbol{U} \in \mathbb{R}^{K \times I}$ and $\boldsymbol{V} \in \mathbb{R}^{K \times J}$ be the user and item latent feature matrixes, with their column vectors $\boldsymbol{U_i}$ and V_j representing the K-dimensional latent vectors of u_i and v_j respectively.

Semi-supervised learning approaches. Graph-based SSL mainly alleviates the data sparsity problem by realizing the smoothness insight on affinity graphs, i.e., close nodes on an affinity graph should have similar labels [13, 46]. In the literature, several different objective functions are proposed to realize graph-based smoothness, e.g., Harmonic Function (HF) [46] and Green's Function (GF) [10]. Take HF for example, its energy function on a pairwise affinity graph $\mathcal{G}$ is

$$\mathcal{L}_P = \frac{\lambda_P}{2} \sum_{i=1}^{I} \sum_{k=1}^{I} \sum_{j=1}^{J} \sum_{o=1}^{J} P_{ij,ko} \left(r_{ij} - r_{ko} \right)^2, \quad (1)$$

where $\{i, j\}$ and $\{k, o\}$ are two nodes on $\mathcal{G}$ and $P_{ij,ko}$ is the weight between them. λ_P controls the global smoothness degree on $\mathcal{G}$ and a bigger λ_P corresponding to a higher rating smoothness degree



(a) U-I rating matrix (b) SSL example (c) LFM example (d) CGM example

Fig. 1. Toy example.

on $\mathcal{G}$. Eq. (9) minimize the rating difference between close nodes and thus achieves smoothness on $\mathcal{G}$, which is the same as propagate the known labels to the unknown ones on the affinity graph [45].

Latent factor models. As we described in Section 2, two promising kinds of LFMs are the basic MF and side information-aided MF. The basic MF approach uses $r_{ij} = \boldsymbol{U_i}^T\boldsymbol{V_j}$ to capture u_i's overall interests in v_j's characteristics, that is the predicted rating of u_i on v_j [30]. Its object is to minimize the difference between the observed ratings and the predicted ratings:

$$\arg\min_{\boldsymbol{U},\boldsymbol{V}} \sum_{i=1}^{I}\sum_{j=1}^{J} I_{ij}^{R}\left(R_{ij}-r_{ij}\right)^{2} + \lambda_U \sum_{i=1}^{I} ||\boldsymbol{U_i}||_F^2 + \lambda_V \sum_{j=1}^{J} ||\boldsymbol{V_j}||_F^2, \qquad (2)$$

where I_{ij}^R is an indicator function that equals to 1 if u_i rated v_j, 0 otherwise, $||\cdot||_F^2$ denotes the Frobenius norm, and λ_U and λ_V are regularization parameters to avoid overfitting.

The side information-aided MF incorporate other information, e.g., item content and context information, to learn a better prior for user and item latent factors. For example, CTR [40] additionally includes a topic proportion which learned from item's content using topic modeling when modeling item latent factor. For its model details, please refer to [40].

From the above explanation of the existing LFMs, we can see that they only focus on fitting the model by regressing the rare known ratings, and neglect the rating smoothness insight between similar users and items.

4 PROPOSED CHAIN GRAPH MODEL

In this section, we propose a novel probabilistic CGM to marry SSL with LFM, which is Insight $\mathcal{I}$ (i.e., principled unification).

4.1 Problem Statement

We first define our recommendation problem. Given a user set $\mathbb{U} = \{u_1, ..., u_I\}$ and an item set $\mathbb{V} = \{v_1, ..., v_J\}$, there are totally $I \times J$ ratings, among which only $\mathscr{L}$ ratings are known, and our recommendation task is to predict the unknown ratings, with the size of $\mathscr{U} = I \times J - \mathscr{L}$.

As described in Section 1, SSL and LFM are popularly used to solve the above recommendation problem, however, they have their own merits and disadvantages. Since we want to marry SSL with LFM, we first use a toy example to illustrate how SSL and LFM make recommendation separately. Let $\boldsymbol{R}$ be the U-I rating matrix, with each element R_{ij} denoting the rating that u_i gives to v_j. Let $\boldsymbol{r}$ be the predicted U-I rating matrix, with its real-valued rating r_{ij} denoting the predicted rating of u_i on v_j. Figure 1 (a) shows a U-I rating matrix example, where we have two users (u_1 and u_2), two

Table 1. Hyperparameters notations in our model

Notation	**Meaning**
C^L	label confidence matrix
μ^U	user prior mean
μ^V	item prior mean
λ_U	user prior confidence
λ_V	item prior confidence
λ_F	global smoothness degree on user graph
λ_G	global smoothness degree on item graph
α	smoothness confidence decay parameter

items (v_1 and v_2), and two known U-I rating pairs (R_{11} and R_{22}). Our mission is to predict the two unknown U-I pairs (r_{12} and r_{21}).

Graph-based SSL makes recommendation mainly based on the smoothness idea on affinity graphs. Since each data object in RS is a U-I rating pair, to make recommendation, SSL first needs to build a U-I pairwise affinity graph, and then propagates the known U-I rating pairs to the unknown ones. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a U-I *pairwise affinity graph* with nodes $\mathcal{V}$ denoting data objects, edges $\mathcal{E}$ denoting the affinitive relationship between nodes, and weight matrix between edges P denoting the affinitive relation strength between nodes. Figure 1 (b) shows how to adopt SSL to do recommendation, where the affinity graph $\mathcal{G}$ is shown in dash lines, and we do not show the edge weight for conciseness. Predictions will be made by propagating R_{11} and R_{22} to r_{12} and r_{21}. That is, the purpose of SSL is rating smoothness on affinity graphs, and SSL captures the mutual dependency between ratings.

LFM makes recommendation mainly based on the dimensionality reduction idea. Let $\boldsymbol{U} \in \mathbb{R}^{K \times I}$ and $\boldsymbol{V} \in \mathbb{R}^{K \times J}$ be the user and item latent feature matrixes, with their column vectors $\boldsymbol{U_i}$ and $\boldsymbol{V_j}$ representing the K-dimensional latent vectors of u_i and v_j respectively. Figure 1 (c) shows how to use LFM to do recommendation. LFM predicts the unknown ratings by learning user and item latent factors through regressing on the known ones. In this example, LFM predicts r_{12} and r_{21} by learning U_1, U_2, V_1, and V_2 through regressing on R_{11} and R_{22}. That is, the purpose of LFM is rating regression, and LFM captures the causal dependency between latent factors and ratings.

Although LFM and SSL have difference purpose, they both are actually used to capture different dependency between different variables. To marry LFM with SSL, we need a model so that such different dependency between different variables can be captured.

4.2 Regression and Smoothness Based Chain Graph Model

We identify probabilistic CGM, a powerful tool for representing the relationship between variables, to be ideal for marrying LFM with SSL. CGM is a combination of Bayesian network and Markov random field [22, 29]. First, Bayesian network, i.e, directed probability graphical model, is commonly used to model the causal dependency between variables. Second, Markov random field, i.e., undirected probability graphical model, is used to model the mutual dependency between variables.

To marry LFM with SSL, the proposed CGM should serve for two goals, as we described in the above example: (1) *Rating regression.* The predicted rating should be close to the observed rating; (2) *Rating smoothness.* The predicted ratings should be smooth between close U-I pairs. Thus, we call our model **R**egression and **S**moothness-based **C**hain **G**raph **M**odel (RSCGM).

Our proposed RSCGM combines Bayesian network and Markov random field. Figure 1 (d) shows an our proposed CGM example, where we use the directed graph (shows with solid arrows) to model rating generation and known rating regression procedures, and use the undirected graph (shows with dash lines) to model the rating smoothness constrain on an affinity graph. RSCGM is a three layer probabilistic graphical model. The first layer is the *latent factor layer*, and its nodes are latent variables, i.e., $\boldsymbol{U}$ and $\boldsymbol{V}$. The second layer is the *prediction layer*, and its nodes are the predicted ratings, i.e., $\boldsymbol{r}$. The third layer is the *observation layer*, and its nodes are the observed data, i.e., $\boldsymbol{R}$. From the latent factor layer to the prediction layer, it is a Bayesian network which denotes the rating generation procedure; I.e., we use $r_{ij} = \boldsymbol{U_i}^T\boldsymbol{V_j}$ as the prediction of a U-I pair. The prediction layer is a Markov random field which denotes the prediction smoothness constrain. From the prediction layer to the observation layer, it is another Bayesian network which denotes the rating regression objective, i.e., the predicted rating should be close to the observed rating.

Joint distribution over all the variables. We first give the joint distribution over all the variables in all the three layers. The Markov property of chain graph model [24], i.e., conditional independence relations between variables, indicates that the probability of a node on a CGM only depends on its directed neighbors. Thus, we factorize the joint distribution over all the variables as:

$$\begin{aligned}&P(\boldsymbol{U},\boldsymbol{V},\boldsymbol{r},\boldsymbol{R}|C^{\boldsymbol{L}},\boldsymbol{\mu}^{\boldsymbol{U}},\boldsymbol{\mu}^{\boldsymbol{V}},\lambda_U,\lambda_V)\\ \propto &P(\boldsymbol{R}|\boldsymbol{r},C^{\boldsymbol{L}})P(\boldsymbol{r}|\boldsymbol{U},\boldsymbol{V})P\left(\boldsymbol{U}|\boldsymbol{\mu}^{\boldsymbol{U}},\lambda_U\right)P\left(\boldsymbol{V}|\boldsymbol{\mu}^{\boldsymbol{V}},\lambda_V\right).\end{aligned} \tag{3}$$

We then derive the conditional probabilities of variables in each layer.

Latent factor layer. Our model starts with latent factors which is also the start of the Bayesian network. We place Gaussian priors on user latent factor:

$$P\left(\boldsymbol{U}|\boldsymbol{\mu}^{\boldsymbol{U}},\lambda_U\right) = \prod_{i=1}^{I}\mathcal{N}(\boldsymbol{U_i}|\boldsymbol{\mu_i^U},\lambda_U^{-1}\boldsymbol{I_K}), \tag{4}$$

where $\boldsymbol{I_K}$ is a K-dimensional identity matrix, and $\mathcal{N}(\mu,\lambda)$ is the probability density function of the Gaussian distribution with mean μ and variance λ. Note that $\boldsymbol{\mu}^{\boldsymbol{U}}$ is the user prior mean matrix with each column $\boldsymbol{\mu_i^U}$ denoting the mean of each user latent factor. $\boldsymbol{\mu}^{\boldsymbol{U}}$ can be obtained from additional information. For example, [19] takes the average preference of a user's friends as the mean of his prior. Similarly, we place another Gaussian prior on item factor:

$$P\left(\boldsymbol{V}|\boldsymbol{\mu}^{\boldsymbol{V}},\lambda_V\right) = \prod_{j=1}^{J}\mathcal{N}(\boldsymbol{V_j}|\boldsymbol{\mu_j^V},\lambda_V^{-1}\boldsymbol{I_K}), \tag{5}$$

where $\boldsymbol{\mu}^{\boldsymbol{V}}$ is the item prior mean matrix with each column $\boldsymbol{\mu_j^V}$ denoting the mean of each item latent factor, which can also be obtained from additional information. For example, CTR [40] takes the topic allocation learned from the content information of an item as the mean of its prior.

Prediction layer. The prediction $\boldsymbol{r}$ comes from the latent factor layer, and has smoothness constrain between themselves. As a result, the probability of $\boldsymbol{r}$ depends on two independent parts: (1) variables from the first layer, i.e., $\boldsymbol{U}$ and $\boldsymbol{V}$; (2) affinitive neighbors on user and item affinity graphs. The first part corresponds to the directed graph that comes from the latent factor layer to the prediction layer in Figure 1. The second part corresponds to the undirected graph in the second layer of Figure 1. Based on the Markov property of chain graph model [24], we have

$$P(\boldsymbol{r}|\boldsymbol{U},\boldsymbol{V}) = \frac{1}{Z}\phi_1(\boldsymbol{U},\boldsymbol{V},\boldsymbol{r})\phi_2(\boldsymbol{r}) \propto \phi_1(\boldsymbol{U},\boldsymbol{V},\boldsymbol{r})\phi_2(\boldsymbol{r}), \tag{6}$$

where Z is a normalizer that makes sure the probability equals to 1.

In Eq. (6), we define the first term $\phi_1(\boldsymbol{U},\boldsymbol{V},\boldsymbol{r}) = \delta(\boldsymbol{r} - \boldsymbol{U}^T\boldsymbol{V})$ with $\delta()$ denoting the Dirac delta function [11], and the property of Dirac delta function indicates that integrating out $\boldsymbol{r}$ is equivalent to replacing $\boldsymbol{r}$ with $\boldsymbol{U}^T\boldsymbol{V}$, which is the rating generation procedure. The second term $\phi_2(\boldsymbol{r}) = exp\{-E\}$ is probability that constrains rating smoothness with E denoting the rating smoothness energy function on affinity graphs, i.e., the confidence-aware joint-smoothness objective function that we will present in Section 5.

Observation layer. Each node in the observation layer is an observed rating of an prediction from the prediction layer, and it corresponds to the rating regression part from the prediction layer to the observation layer in Figure 1. As a result, the probability of $\boldsymbol{R}$ is conditional on $\boldsymbol{r}$ in the prediction layer. We adopt a Gaussian prior here, the same as the existing LFM [30]:

$$P\left(\boldsymbol{R}|\boldsymbol{r},\boldsymbol{C}^{\boldsymbol{L}}\right) = \prod_{i=1}^{I}\prod_{j=1}^{J}\mathcal{N}(R_{ij}|r_{ij}, {C_{ij}^{L}}^{-1}), \tag{7}$$

where $\boldsymbol{C}^{\boldsymbol{L}} \in \mathbb{R}^{I\times J}$ is a label confidence matrix with each element C_{ij}^{L} denoting the label confidence for each U-I pair, and more details refer to [40].

Posterior distribution over latent factors. Finally, we use maximum a posteriori (MAP) probability to learn the best $\boldsymbol{U}$ and $\boldsymbol{V}$. Based on Eq. (3), Eq. (4), Eq. (5), Eq. (6), and Eq. (7), we have the following posterior distribution over user and item latent factors by using Bayes' theorem,

$$\begin{aligned}
&P(\boldsymbol{U},\boldsymbol{V}|\boldsymbol{r},\boldsymbol{R},\boldsymbol{C}^{\boldsymbol{L}},\boldsymbol{\mu}^{\boldsymbol{U}},\boldsymbol{\mu}^{\boldsymbol{V}},\lambda_U,\lambda_V)\\
\propto &P\left(\boldsymbol{U}|\boldsymbol{\mu}^{\boldsymbol{U}},\lambda_U\right)P\left(\boldsymbol{V}|\boldsymbol{\mu}^{\boldsymbol{V}},\lambda_V\right)\\
&\delta(\boldsymbol{r}-\boldsymbol{U}^T\boldsymbol{V})\phi_2(\boldsymbol{r})P(\boldsymbol{R}|\boldsymbol{r},\boldsymbol{C}^{\boldsymbol{L}}).
\end{aligned} \tag{8}$$

We will present how to learn the MAP distribution over the user and item latent factors in Section 6.

5 REALIZING SMOOTHNESS

In this section, we present the confidence-aware joint-smoothness energy function, which will enable us to obtain $\phi_2(\boldsymbol{r})$ shown in Eq. (6).

5.1 From Pairwise to Joint Smoothness

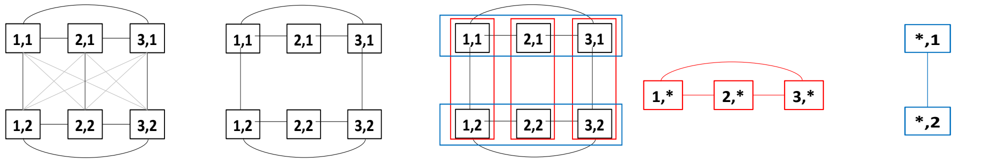
Pairwise smoothness has severe scalability problem. As Section 1 mentioned, building such an affinity graph requires $O(I \times J)^2$ for I users and J items, i.e., Challenge $\mathcal{I}\mathcal{I}$. To solve it, we propose an efficient way to perform smoothness.

In RS, each data object connects two elements, i.e., user and item. Thus, rating smoothness implies smoothness exists on two elements, i.e., user and item, and we term it "joint smoothness". We try to find the relationship between pairwise and joint smoothness. Figure 2 shows a derivation example from pairwise to joint smoothness, where we have three users and two items, and we simply use (i, j) to denote (u_i, v_j) pair. For any two pairs on $\mathcal{G}$, e.g., (u_i, v_j) and (u_k, v_o) ($i \neq j$ or $k \neq o$), based on HF [46], pairwise objective energy function can be shown as:

$$\mathcal{L}_P = \frac{\lambda_P}{2}\sum_{i=1}^{I}\sum_{k=1}^{I}\sum_{j=1}^{J}\sum_{o=1}^{J}P_{ij,ko}\left(r_{ij}-r_{ko}\right)^2, \tag{9}$$

which can be further divided into the following three terms based on different (u_i, v_j) and (u_k, v_o) combinations:

$$\mathcal{L}_P = \mathcal{L}'_U + \mathcal{L}'_V + \mathcal{L}'_P, \tag{10}$$



(a) pairwise graph (b) remove diagonal edges (c) merge nodes (d) user graph (e) item graph

Fig. 2. Pairwise to joint smoothness example.

where

$$
\begin{aligned}
\mathcal{L}'_U &= \frac{\lambda_P}{2}\sum_{i=1}^{I}\sum_{k=1}^{I}\sum_{j=1}^{J} P_{ij,kj}\Big(r_{ij}-r_{kj}\Big)^2, i \neq k \\
\mathcal{L}'_V &= \frac{\lambda_P}{2}\sum_{k=1}^{I}\sum_{j=1}^{J}\sum_{o=1}^{J} P_{kj,ko}\Big(r_{kj}-r_{ko}\Big)^2, j \neq o \\
\mathcal{L}'_P &= \frac{\lambda_P}{2}\sum_{i=1}^{I}\sum_{k=1}^{I}\sum_{j=1}^{J}\sum_{o=1}^{J} P_{ij,ko}\Big(r_{ij}-r_{ko}\Big)^2, i \neq k, j \neq o.
\end{aligned}
\tag{11}
$$

First, we remove the diagonal edges from the pairwise graph, i.e., remove $\mathcal{L}'_P$ from Eq. (10). Figure 2(b) shows the result after we remove the diagonal edges from Figure 2(a). We use δ_U to denote the difference between r_{ij} and r_{kj}, i.e., $\delta_U = |r_{ij} - r_{kj}|$, δ_V to denote the difference between r_{kj} and r_{ko}, i.e., $\delta_V = |r_{kj} - r_{ko}|$, and δ_P to denote the difference between r_{ij} and r_{ko}, i.e., $\delta_P = |r_{ij} - r_{ko}|$. We know that the difference between r_{ij} and r_{ko} has a upper bound of $\delta_u + \delta_v$, since $|r_{ij} - r_{ko}| = |r_{ij} - r_{kj} + r_{kj} - r_{ko}| \leq |r_{ij} - r_{kj}| + |r_{kj} - r_{ko}|$, i.e, $\delta_P \leq \delta_U + \delta_V$. In other words, the rating smoothness between (u_i, v_j) and (u_k, v_j) and the rating smoothness between (u_k, v_j) and (u_k, v_o) will result in the rating smoothness between (u_i, v_j) and (u_k, v_o). Thus, we can still achieve diagonal rating smoothness after we remove diagonal edges.

Second, we compute the edge affinity weight. The U-I pairwise affinity is determined by both user and item affinity, and thus, we take pairwise affinity as a function of user affinity and item affinity. Specifically, we assume $P_{ij,ko} = F_P(F_U(u_i, u_k), F_V(v_j, v_o))$, where $0 \leq F_U(u_i, u_k) \leq 1$ is a function of the similarity between u_i and u_k, $0 \leq F_V(v_j, v_o) \leq 1$ is a function of the similarity between v_j and v_o, and $0 \leq F_P(F_U, F_V) \leq 1$ is a function of the similarity between (u_i, v_j) and (u_k, v_o). We also assume $F_U(u_i, u_i) = 1$ for all the users and $F_V(v_j, v_j) = 1$ for all the items, which means that the similarity between a node itself is 1. For any v_j, since $P_{ij,kj} = F_P(F_U(u_i, u_k), 1)$ holds, i.e., the weight between any (u_i, v_j) and (u_k, v_j) pairs is the same regardless of v_j, we can represent all such weights with one, i.e., $W_{ik} = P_{ij,kj}$ which is the similarity between u_i and u_k. Similarly, we use $S_{jo} = P_{kj,ko}$ to represent the similarity between v_j and v_o.

Next, since $W_{ik} = P_{ij,kj}$ and $S_{jo} = P_{kj,ko}$, we can merge nodes, as shown in Figure 2(c). Specifically, we merge nodes (u_i, v_j) for the v_j into one node u_i which has J labels, and we merge nodes (u_i, v_j) for all the u_i into one node v_j which has I labels, as shown in Figures 2(d) and 2(e). That is, we decompose the pairwise graph into user and item joint graphs, and decrease the time complexity to build joint affinity graphs to $O(I^2 + J^2)$ by computing the similarity between users and items separately. Thus, $\mathcal{L}'_U$ and $\mathcal{L}'_V$ change to:

$$
\mathcal{L}_U = \frac{\lambda_P}{2}\sum_{i=1}^{I}\sum_{k=1}^{I}\sum_{j=1}^{J} W_{ik}\Big(r_{ij}-r_{kj}\Big)^2, \tag{12}
$$

$$
\mathcal{L}_V = \frac{\lambda_P}{2}\sum_{j=1}^{J}\sum_{o=1}^{J}\sum_{k=1}^{I} S_{jo}\Big(r_{kj}-r_{ko}\Big)^2, \tag{13}
$$

Finally, to achieve joint smoothness, we need to leverage the effect of user and item rating smoothness. To do this, we use two parameters, i.e., λ_F and λ_G, to control the global smoothness degree on $\mathcal{G}_1$ and $\mathcal{G}_2$. Combining Eq. (12) and Eq. (13), we define joint smoothness objective as:

$$\mathcal{L}_J = \frac{\lambda_F}{2}\mathcal{L}_U + \frac{\lambda_G}{2}\mathcal{L}_V, \tag{14}$$

where a bigger λ_F corresponds to a higher user rating smoothness degree on $\mathcal{G}_1$ and a bigger λ_G a higher item rating smoothness degree on $\mathcal{G}_2$. We will perform experiments to compare pairwise and joint smoothness in Section 7.4.

5.2 User and Item affinity Graphs

We now explain how to build user and item affinity graphs.

User affinity graph. A user affinity graph should capture affinitive relationships between users, so that the ratings of close users will be similar, as we define below:

Definition 1: A *user affinity graph* is an undirected weighted graph $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$, where (1) $\mathcal{V}_1 = \mathbb{U}$ is the node set; that is, each node is a user with J labels, corresponding to the ratings each user gives to all them items; (2) $\mathcal{E}_1$ is the edge set, with W_{ik} denoting the weight of edge $\mathcal{E}_{ik}$; that is, W_{ik} is the relationship strength between nodes u_i and u_k, which is symmetric, i.e., $W_{ik} = W_{ki}$.

We suggest to build user affinity graph using whatever information that is available to capture the affinitive relationships between users, e.g., ratings and user social relationships. For example, user social relationships also indicate users' common interests, and thus can be taken as the user affinity graph.

Item affinity graph. Similarly, an item affinity graph should capture affinitive relationships between items, so that the ratings of close items will be similar, as defined as below:

Definition 2: An *item affinity graph* is an undirected weighted graph $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$, where (1) $\mathcal{V}_2 = \mathbb{V}$ is the node set; that is, each node is an item with I labels, corresponding to the ratings it receives from all the users; (2) $\mathcal{E}_2$ is the edge set, with S_{jo} denoting the weight between of edge $\mathcal{E}_{jo}$; that is, S_{jo} is the relationship strength between nodes v_j and v_o, which is symmetric, i.e., $S_{jo} = S_{oj}$.

One can also build item affinity graph using whatever information that is available to capture the affinitive relationships between items, e.g., ratings and item content information. Frequently, cosine similarity, Jaccard's coefficient (JC), and Pearson correlation coefficient (PCC) are used to measure rating similarity between items [26].

The joint affinity graphs we build are unreliable, since we use rare existing information. Thus, we propose a confidence-aware approach to realize smoothness on them.

5.3 Confidence-aware Joint-smoothness

We finally present a confidence-aware approach to realize smoothness, which solves Challenge III (i.e., unreliable affinity).

Confidence-aware user rating smoothness. User rating smoothness on $\mathcal{G}_1$ constrains that close users on $\mathcal{G}_1$ have similar ratings on items. This is consistent with the reality: e.g., Alice is wondering which movie to watch, and she finds her close friends gave high ratings to "The Dark Knight", and she is likely to watch "The Dark Knight".

The build affinity graphs are unreliable, and thus full smoothness, i.e., assuming smoothness exists everywhere, will be an overly strong assumption. We identify to solve this by using selective smoothness approach. As described in Section 2, rating smoothness can be viewed as propagating known ratings to unknown ones. Since the user affinity graph is unreliable, intuitively, smoothness confidence will decrease with the rating propagation length.

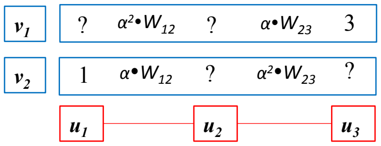

v_1
? $\alpha^2 \bullet W_{12}$? $\alpha \bullet W_{23}$ 3
v_2
1 $\alpha \bullet W_{12}$? $\alpha^2 \bullet W_{23}$?
u_1
u_2
u_3

Fig. 3. Confidence-aware user rating smoothness.

To achieve confidence-aware user rating smoothness, we propose a confidence decay parameter, i.e., $0 \leq \alpha \leq 1$, to control rating propagation length. Figure 3 shows a user affinity graph, where we have three users and two items. Although we use only one edge W_{ik} to express the similarity for (u_i, v_j) and (u_k, v_j) pairs, there is actually a smoothness confidence between them. From Figure 3, we can see how the rating smoothness confidence decreases with propagation length. For example, the smoothness confidence between (u_3, v_1) and (u_2, v_1) should be bigger than that between (u_2, v_1) and (u_1, v_1), since the label of (u_3, v_1), i.e., R_{31}, will first be propagated to (u_2, v_1) and then to (u_1, v_1).

In general, suppose we have two users (u_i and u_k) and an item (v_j), we define the smoothness confidence between (u_i, v_j) and (u_k, v_j) pairs as:

$$C_{i,j,k} = \alpha^{|d_{i,j,k}|+1}, \tag{15}$$

where α is the confidence decay parameter that ranges in $[0, 1]$, and $|d_{i,j,k}| = min\{|d_i|, |d_k|\}$ with $|d_i|$ and $|d_k|$ denoting the distances from u_i and u_k to a user whose rating on v_j is given, respectively.

The proposed confidence-aware smoothness approach, on one hand, alleviates the overly strong assumption of fully smoothness, and, on the other hand, reduces computation. For u_i and u_k on $\mathcal{G}_1$ and any v_j, given smoothness confidence in Eq. (15) and following Eq. (12), we define confidence-aware user rating smoothness energy function as:

$$E_U = \underset{\boldsymbol{U},\boldsymbol{V}}{\arg\min} \sum_{i=1}^{I} \sum_{k=1}^{I} \sum_{j=1}^{J} C_{i,j,k} W_{ik} \left(r_{ij} - r_{kj}\right)^2. \tag{16}$$

Confidence-aware item rating smoothness. Similarly, item rating smoothness on $\mathcal{G}_2$ restricts that close items have similar ratings. This is also consistent with reality: e.g., Alice has watched movie “The Dark Knight” and loves it so much. She then finds “Batman Begins” is a similar movie, and she will probably likes “Batman Begins” as well.

Similar to user rating smoothness, for v_j and v_o on $\mathcal{G}_2$ and any u_k, following Eq. (13), we define confidence-aware item rating smoothness energy function as,

$$E_V = \underset{\boldsymbol{U},\boldsymbol{V}}{\arg\min} \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{o=1}^{J} C_{k,j,o} S_{jo} \left(r_{kj} - r_{ko}\right)^2, \tag{17}$$

where $C_{k,j,o} = \alpha^{|d_{k,j,o}|+1}$ has the similar meaning to Eq. (15).

Following Eq. (14), combining Equations (16) & (17), we define confidence-aware joint smoothness energy function as:

$$E_J = \frac{\lambda_F}{2} E_U + \frac{\lambda_G}{2} E_V. \tag{18}$$

Having E_J will enable us to obtain $\phi_2(\boldsymbol{r})$ shown in Eq. (6).

Algorithm 1: Learning RSCGM

```
Input: ratings in training set $R$, user affinity graph $\mathcal{G}_1$, item affinity graph $\mathcal{G}_2$, $\alpha$, $\lambda_U$, $\lambda_V$, $\lambda_G$, $\lambda_F$
Output: user latent profile matrix $U$ and item latent profile matrix $V$
initialize $U$ and $V$
repeat
    for $r_{ij} \in R$ do
        update $U_i$ based on Eq. (21)
        update $V_j$ based on Eq. (22)
until convergence;
return $U$ and $V$
```

6 MODEL PARAMETER LEARNING

Based on Equations (4), (5), (6), (7), and (18), the posterior distribution over the user and item latent factors in Eq. (8) becomes

$$\begin{aligned}
&P(\boldsymbol{U},\boldsymbol{V}|\boldsymbol{r},\boldsymbol{R},\boldsymbol{C^L},\boldsymbol{\mu^U},\boldsymbol{\mu^V},\lambda_U,\lambda_V)\\
&\propto P\left(\boldsymbol{U}|\boldsymbol{\mu^U},\lambda_U\right) P\left(\boldsymbol{V}|\boldsymbol{\mu^V},\lambda_V\right) \delta(\boldsymbol{r}-\boldsymbol{U}^T\boldsymbol{V})\phi_2(\boldsymbol{r})P(\boldsymbol{R}|\boldsymbol{r},\boldsymbol{C^L})\\
&=\prod_{i=1}^{I}\mathcal{N}(\boldsymbol{U_i}|\boldsymbol{\mu_i^U},\lambda_U^{-1}\boldsymbol{I_K})\times\prod_{j=1}^{J}\mathcal{N}(\boldsymbol{V_j}|\boldsymbol{\mu_j^V},\lambda_V^{-1}\boldsymbol{I_K})\\
&\times exp\{-\frac{\lambda_F}{2}\sum_{i=1}^{I}\sum_{k=1}^{I}\sum_{j=1}^{J}\alpha^{|d_{i,j,k}|+1}W_{ik}\left(r_{ij}-r_{kj}\right)^2 \qquad (19)\\
&-\frac{\lambda_G}{2}\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{o=1}^{J}\alpha^{|d_{k,j,o}|+1}S_{jo}\left(r_{ij}-r_{kj}\right)^2\}\\
&\times\prod_{i=1}^{I}\prod_{j=1}^{J}\mathcal{N}(R_{ij}|r_{ij},C_{ij}^{L\,-1}).
\end{aligned}$$

Maximizing the log of the posterior probability is equivalent to minimizing the following objective function:

$$\begin{aligned}
\mathcal{L}=&\mathop{\arg\min}_{\boldsymbol{U},\boldsymbol{V}}\frac{1}{2}\sum_{i=1}^{I}\sum_{j=1}^{J}C_{ij}^{L}\left(R_{ij}-\boldsymbol{U_i}^T\boldsymbol{V_j}\right)^2\\
&+\frac{\lambda_F}{2}\sum_{i=1}^{I}\sum_{k=1}^{I}\sum_{j=1}^{J}\alpha^{|d_{i,j,k}|+1}W_{ik}\left(\boldsymbol{U_i^T}\boldsymbol{V_j}-\boldsymbol{U_k^T}\boldsymbol{V_j}\right)^2\\
&+\frac{\lambda_G}{2}\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{o=1}^{J}\alpha^{|d_{k,j,o}|+1}S_{jo}\left(\boldsymbol{U_i^T}\boldsymbol{V_j}-\boldsymbol{U_i^T}\boldsymbol{V_o}\right)^2 \qquad (20)\\
&+\frac{\lambda_U}{2}\sum_{i=1}^{I}\left(\boldsymbol{U_i}-\boldsymbol{\mu_i^U}\right)^T\left(\boldsymbol{U_i}-\boldsymbol{\mu_i^U}\right)\\
&+\frac{\lambda_V}{2}\sum_{j=1}^{J}\left(\boldsymbol{V_j}-\boldsymbol{\mu_j^V}\right)^T\left(\boldsymbol{V_j}-\boldsymbol{\mu_j^V}\right),
\end{aligned}$$

where we replace r_{ij} with $\boldsymbol{U_i}^T\boldsymbol{V_j}$. We minimize Eq. (20) by performing coordinate descent approach, that is, we fix the hyper-parameters, and iteratively optimize the user and item latent factors $\boldsymbol{U_i}$ and $\boldsymbol{V_j}$.

Specifically, we take the gradient of $\mathcal{L}$ with respect to $\boldsymbol{U_i}$ and $\boldsymbol{V_j}$, set it to zero and get,

$$\begin{aligned}\boldsymbol{U_i} \leftarrow &\left\{\sum_{j=1}^{J} \boldsymbol{V_j} C_{ij}^{L} \boldsymbol{V_j}^T + \lambda_U \boldsymbol{I_K} + \lambda_F \sum_{k=1}^{I}\sum_{j=1}^{J} \alpha^{|d_{i,j,k}|+1} \boldsymbol{V_j} W_{ik} \boldsymbol{V_j}^T \right.\\ &\left. + \lambda_G \sum_{j=1}^{J}\sum_{o=1}^{J} \alpha^{|d_{k,j,o}|+1} (\boldsymbol{V_j} - \boldsymbol{V_o}) S_{jo} (\boldsymbol{V_j} - \boldsymbol{V_o})^T \right\}^{-1} \\ &\{\sum_{j=1}^{J} C_{ij}^{L} R_{ij} \boldsymbol{V_j}^T + \lambda_U \boldsymbol{\mu_i^U} + \lambda_F \sum_{k=1}^{I}\sum_{j=1}^{J} \alpha^{|d_{i,j,k}|+1} \boldsymbol{V_j} W_{ik} \boldsymbol{V_j}^T \boldsymbol{U_k}\},\end{aligned} \tag{21}$$

$$\begin{aligned}\boldsymbol{V_j} \leftarrow &\left\{\sum_{i=1}^{I} \boldsymbol{U_i} C_{ij}^{L} \boldsymbol{U_i}^T + \lambda_V \boldsymbol{I_K} + \lambda_G \sum_{i=1}^{I}\sum_{o=1}^{J} \alpha^{|d_{k,j,o}|+1} \boldsymbol{U_i} S_{jo} \boldsymbol{U_i}^T \right.\\ &\left. + \lambda_F \sum_{i=1}^{I}\sum_{k=1}^{I} \alpha^{|d_{i,j,k}|+1} (\boldsymbol{U_i} - \boldsymbol{U_k}) W_{ik} (\boldsymbol{U_i} - \boldsymbol{U_k})^T \right\}^{-1} \\ &\{\sum_{i=1}^{I} C_{ij}^{L} R_{ij} \boldsymbol{U_i}^T + \lambda_V \boldsymbol{\mu_j^V} + \lambda_G \sum_{i=1}^{I}\sum_{o=1}^{J} \alpha^{|d_{k,j,o}|+1} \boldsymbol{U_i} S_{jo} \boldsymbol{U_i}^T \boldsymbol{V_o}\}.\end{aligned} \tag{22}$$

We summarize the learning of RSCGM in Algorithm 1. After the optimal $\boldsymbol{U_i}$ and $\boldsymbol{V_j}$ are learned, they can be used to make predictions through $r_{ij} = (\boldsymbol{U_i^*})^T \boldsymbol{V_j^*}$.

Computation Analysis. We now analysis the time complexity of our model inference. From Eq. (20), we can see that the *time complexity* of realizing our model is $O(\rho_R(\sum \overline{F} + \sum \overline{G})K)$, where ρ_R is the size of observed data, $\sum \overline{F}$ and $\sum \overline{G}$ are the average number of edges we need to do rating smoothness (propagation) in $\mathcal{G}_1$ and $\mathcal{G}_2$.

Due to the data sparsity problem, the best value of α is usually very small (about 0.5 in our experiments), which indicates that the known ratings will only be propagated to their close U-I pairs. With a big d, the smoothness degree will decay by α^d which will be so small that can be neglected. Due to the data sparsity problem, the average number of neighbors on $\mathcal{G}_1$ and $\mathcal{G}_2$ are usually very small, which indicates that $\sum \overline{F} \ll \mathscr{L}$ and $\sum \overline{G} \ll \mathscr{L}$. Besides, since $K \ll \mathscr{L}$, our algorithm scales linearly with the observed data size $\mathscr{L}$.

7 EXPERIMENTS

In this section, we present the comprehensive experiments that aimto answer five key questions: (1) How well does our approach handle the data sparsity problem when comparing with the state-of-the-art approaches? (2) What is the performance difference between pairwise smoothness and joint smoothness? (3) How do parameters λ_F, λ_G, and α affect our model performance? (4) What is the time complexity of our approach?

7.1 RSCGM Realizations

Our proposed RSCGM is a general model to marry SSL with LFM. As described in Section 2, there are mainly two kinds of LFMs in existing work. Thus, we apply RSCGM into both of them, i.e., a basic MF (BMF) [30] and a content-based MF (CMF) [40]. BMF assumes zero mean Gaussian on both user and item latent factors, i.e., $\mu_i^U = \mu_j^V = 0$, while CMF assumes zero mean Gaussian on user latent factor and an item topic allocation mean Gaussian on item latent factor, i.e., $\mu_i^U = 0$

Table 2. Dataset description

Dataset	users	items	tags	user social relations	U-I ratings	rating density
MovieLens	71,567	10,681	–	–	10,000,054	1.31%
Netflix	480,189	17,770	–	–	100,480,507	1.18%
Delicious	1,867	69,226	53,388	15,328	104,799	0.08%
Lastfm	1,892	17,632	11,946	25,434	92,834	0.28%

and $\mu_j^V = \theta_j$, where θ_j is the topic allocation learned from item content information using topic modeling technique [40].

7.2 Experiment Setting

Datasets. To study how RSCGM behaves under the BMF scenario, we use Movielens-10M dataset (*MovieLens*) [17] and the classic *Netflix* dataset [3]. Both datasets and popular and famous, and the ratings of which are integers that rage from 0 to 5. To study how RSCGM behaves under the CMF scenario, we use hetrec2011-delicious-2k dataset (*Delicious*) and dataset hetrec2011-lastfm-2k (*Lastfm*) [7]. These datasets, as described in Table 2, have been popularly used [31]. For *MovieLens* and *Netflix*, we compute the PCC similarity between users and items to build user and item affinity graphs. For *Delicious* and *Lastfm*, we consider the rating of a user on an item as '1' if this user has bookmarked (or listened) this item, '0' otherwise, and we take the tags of each item as its content information. We use user social information to build user affinity graph, and compute the JC similarity between items to build item affinity graph.

Baseline methods. For BMF scenario, we compare RSCGM with the following state-of-the-art methods:

- **ICF** [35] is an item-based collaborative filtering approach.
- **SSL** [10, 41] is a directly application of SSL, and we use it to do rating propagation on our built item affinity graph.
- **BMF** [30] is a classic MF approach.
- **BMF-ULFR** [26] adds user latent factor restriction (ULFR) on BMF.
- **BMF-UILFR** [15] adds both user and item latent factor restriction (UILFR) on a weighted nonnegative MF approach. To make a fair comparison, we replace the weighted nonnegative MF approach with BMF.

For CMF scenario, we compare RSCGM with the following state-of-the-art methods:

- **CMF** [40] is a popular content information aided MF approach.
- **CMF-SMF** [31] adds user social factorization in CMF.
- **CMF-ULFR** [9] adds ULFR on CMF.
- **CMF-UILFR** [15] adds UILFR on a weighted nonnegative MF approach. To make a fair comparison, we also replace the weighted nonnegative MF approach with CMF.

Evaluation metrics. Since ratings range from '1' to '5' on *MovieLens*, we use Mean Absolute Error (*MAE*) and Root Mean Square Error (*RMSE*) to evaluate prediction performance [9, 26]:

$$MAE = \frac{\sum\limits_{(i,j)\in\tau} |R_{ij} - r_{ij}|}{|\tau|}, \tag{23}$$

$$RMSE = \sqrt{\frac{1}{|\tau|} \sum_{(i,j)\in\tau} (R_{ij} - r_{ij})^2}, \tag{24}$$

where $|\tau|$ is the number of predictions in the test dataset τ.

Since ratings range in {'0', '1'} on *Delicious* and *Lastfm*, we use *Precision* and *Recall* to evaluate prediction performance [31, 40]. For each user, precision and recall are defined as follows:

$$Precision@|M| = \frac{Number\ of\ items\ the\ user\ likes\ in\ M}{|M|},$$

$$Recall@|M| = \frac{Number\ of\ items\ the\ user\ likes\ in\ M}{Total\ number\ of\ items\ the\ user\ likes},$$

where M is the returned items. We compute the average of all the users' precision and recall in the test dataset as the final result.

Parameter setting: Before comparison, we first use fivefold cross validation to find the best values of λ_U and λ_V, and then we set accordingly for other models. For parameter λ_q in CMF-SMF, λ_f in BMF-ULFR and CMF-ULFR, λ and μ in BMF-UILFR and CMF-UILFR, and λ_F and λ_G in our model, we find their best values in $[10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2, 10^3]$. We also search the best value of α in our model in [0, 1]. All the experiments are done on a PC by using OpenMP* to perform parallel computing.

7.3 Comparison Results and Analysis

We compare our proposed model, i.e., RSCGM, with the state-of-the-art approaches on all the four dataset to prove the effectiveness of our approach. Our experiments are divided into two parts: comparison with the existing BMF-based models and comaprision with the existing CMF-based models.

7.3.1 Comparison with BMF-based models. First, we report the experiments of comparing RSCGM and the existing BMF models conducted on *MovieLens* and *Netflix* datasets. Since our key insight of this paper is to use the idea of SSL to alleviate the data sparsity problem of LFM, we focus on evaluating each model's performance under different data sparsity scenerios. We use two strategies to generate datasets with different spasity: (1) rating sample, that is, we randomly sample some ratings and remove them from the original *MovieLens* and *Netflix* datasets, and (2) user sample, that is, we remove the users whose ratings are bigger than a certain threshold. Through rating sample strategy, we get several sub-datasets with diffferent sparsity, i.e., *MovieLens-x%*, which means we randomly remove *x%* of the ratings from the original dataset. Similarly, through user sample strategy, we get several sub-datasets with diffferent sparsity, i.e., *MovieLens-y*, which means we remove the users whose rating numbers are bigger than y. Rating sample and user sample strategies can also be used together, and *MovieLens-y-x%* means we first use user sample strategy to remove the users whose rating numbers are bigger than y and then randomly remove *x%* of the ratings from the rest dataset.

During our experiments, we use five-fold cross validation method. Table 3 shows the *MAE* and *RMSE* comparison result on different rating sample *MovieLens* datasets. Table 4 shows the *MAE* and *RMSE* comparison result on different user sample *MovieLens* datasets. Table 5 shows the *MAE* and *RMSE* comparison result on different user and rating sample *MovieLens* datasets. From them, we have the following observations:

- The latent factor models significantly outperforms SSL and ICF, due to its dimensionality reduction technique.

[*] http://openmp.org/

Table 3. Performance comparison on rating sample *MovieLens* datasets

Datasets	MovieLens		MovieLens-70%		MovieLens-80%		MovieLens-85%		MovieLens-90%		MovieLens-95%	
Sparsity	1.31%		0.39%		0.26%		1.97%		0.14%		0.07%	
Metrics	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
SSL	0.7848	1.1315	0.7987	1.1434	0.8100	1.1548	0.8273	1.1788	0.8606	1.2173	1.0407	1.4385
ICF	0.7318	0.9494	0.7966	1.0319	0.8211	1.0619	0.8398	1.0889	0.8613	1.1136	0.9087	1.1681
BMF	0.6356	0.8290	0.6780	0.8831	0.6934	0.9022	0.7072	0.9244	0.7446	0.9681	0.8254	1.0785
BMF-ULFR	0.6351	0.8277	0.6767	0.8805	0.6933	0.9014	0.7020	0.9178	0.7446	0.9679	0.8023	1.0376
BMF-UILFR	**0.6351**	0.8277	0.6752	0.8762	0.6916	0.8948	0.7000	0.9098	0.7415	0.9586	0.7958	1.0317
RSCGM	0.6352	**0.8240**	**0.6702**	**0.8678**	**0.6818**	**0.8810**	**0.6889**	**0.8928**	**0.7268**	**0.9385**	**0.7899**	**1.0166**
improv. vs. BMF	0.07%	0.5%	1.15%	1.74%	1.67%	2.35%	2.58%	3.42%	2.39%	3.05%	4.31%	5.73%

Table 4. Performance comparison on user sample *MovieLens* datasets

Datasets	Sparsity	Metrics	SSL	ICF	BMF	BMF-ULFR	BMF-UILFR	RSCGM	improv. vs. BMF
MovieLens-100	0.44%	MAE	0.8362	0.8192	0.7409	0.7250	0.7232	**0.7172**	3.20%
		RMSE	1.2115	1.0680	0.9677	0.9435	0.9422	**0.9331**	3.58%
MovieLens-75	0.37%	MAE	0.8706	0.8316	0.7715	0.7531	0.7481	**0.7442**	3.53%
		RMSE	1.2606	1.0850	1.0070	0.9844	0.9737	**0.9555**	5.11%
MovieLens-50	0.30%	MAE	0.8958	0.8616	0.8105	0.7781	0.7724	**0.7649**	5.62%
		RMSE	1.2895	1.1244	1.0543	1.0118	1.0029	**0.9858**	6.50%

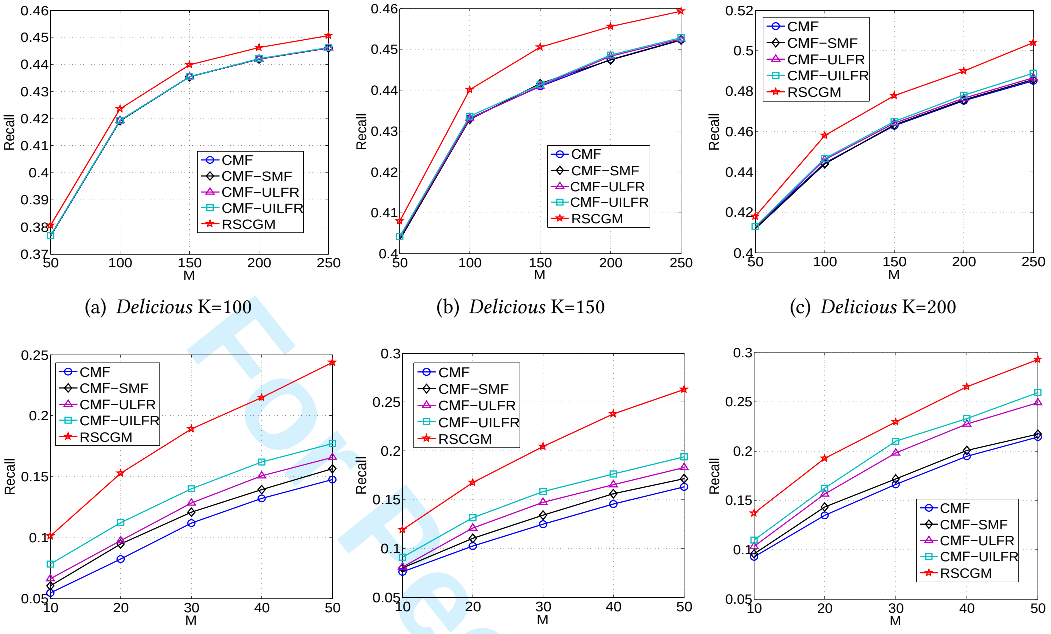
Table 5. Performance comparison on user and rating sample *Netflix* datasets

Datasets	Sparsity	Metrics	SSL	ICF	BMF	BMF-ULFR	BMF-UILFR	RSCGM	improv. vs. BMF
Netflix-200-70%	0.14%	MAE	0.8846	0.8639	0.7573	0.7569	0.7532	**0.7477**	1.27%
		RMSE	1.2968	1.1147	0.9856	0.9827	0.9710	**0.9534**	3.26%
Netflix-200-80%	0.09%	MAE	0.9187	0.8879	0.7738	0.7708	0.7673	**0.7625**	1.45%
		RMSE	1.3418	1.1463	1.0058	0.9975	0.9814	**0.9691**	3.65%
Netflix-200-90%	0.05%	MAE	1.0264	0.9131	0.7943	0.7939	0.7882	**0.7750**	2.43%
		RMSE	1.4781	1.1758	1.0347	1.0118	1.0009	**0.9842**	4.89%

- With the increase of the data sparsity degree, the performance of all the models decrease. This is the standard data sparsity problem.
- our approach always achieves the best performance. Besides, no matter which kind of sample strategy we use, either rating sample, user sample, or both user and rating sample, there is an obvious trend: the sparser the dataset is, the bigger improvement of our model against the comparison methods. Take the experimental results on different sampled *Netflix* datasets for example, the *RMSE* improvement of our approach over BMF are 3.26%, 3.65%, and 4.89% on *Netflix-200-70%*, *Netflix-200-80%*, and *Netflix-200-90%* respectively.

7.3.2 Comparison with CMF-based models. Next, we report the experiments of comparing RSCGM and the existing CMF models conducted on *Delicious* and *Lastfm* datasets. Note that the original *Delicious* dataset is already quite sparse (0.08%) and *Lastfm* dataset is relative dense. We use rating sample strategy on *Lastfm* to generate datasets with different sparsity degress, and we get *Lastfm20* and *Lastfm50* whose sparsity are 0.22% and 0.14%, respectively. Figure 4 shows the *Recall* performance on *Delicious* and *Lastfm*, and we omit the *Precision* performance since it is similar to *Recall*. Figure 5 shows the *Precision* performance for each model on different sparsity of datasets, and here we omit the *Recall* performance since it is similar to *Precision*. From it, we get:

- With the increase of the data sparsity degree, the performance of all the models decrease. This is the standard data sparsity problem.



(a) *Delicious* K=100 (b) *Delicious* K=150 (c) *Delicious* K=200

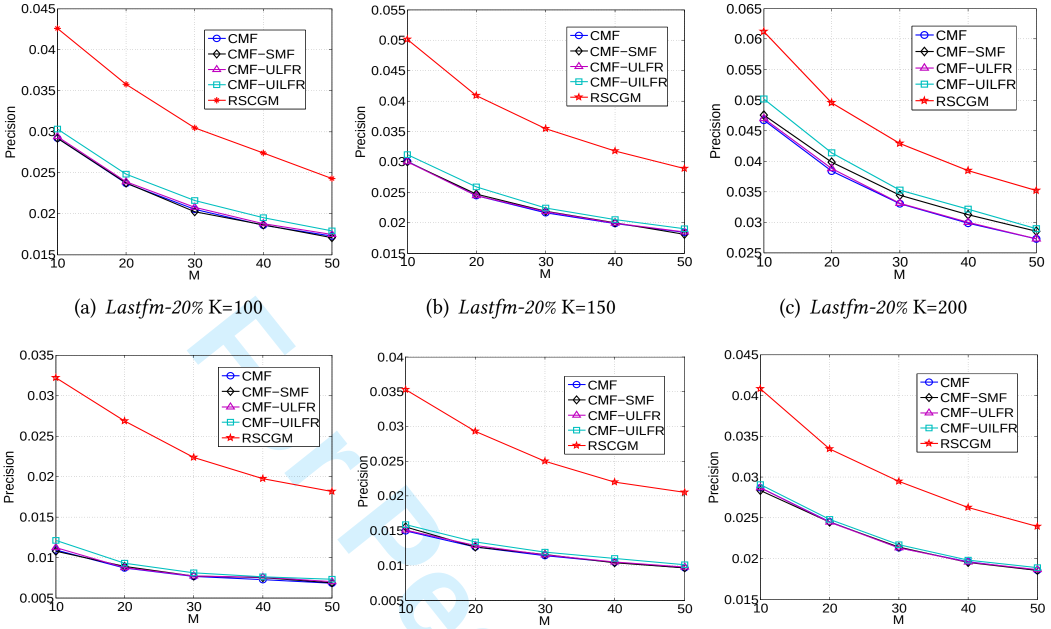
(d) *Lastfm* K=100 (e) *Lastfm* K=150 (f) *Lastfm* K=200

Fig. 4. *Recall* comparison of each method on *Delicious* and *Lastfm*.

- With the increase of the latent factor dimensionality K, the performances of all the models increase, since a bigger K will represent a better latent factor. However, as we will analysis later, it comes with price of higher model training time.
- CMF-SMF and CMF-ULFR slightly outperform CMF due to the additional user social information. Meanwhile, CMF-UILFR adopts additional user and item graph information and behaves as a constant runner-up.
- With the increase of the data sparsity degree, CMF, CMF-SMF, and LFRs (CMF-ULFR and CMF-UILFR) tend to have more similar performance. This is because LFRs' performance are limited due to their overly strong assumption that connected users or items tend to share similar latent factors. This overly strong assumption of LFR fails particularly when affinity graphs are unreliable in the data sparsity scenario.
- Our model (RSCGM) significantly outperforms CMF-UILFR (e.g., 7.04% on *Lastfm*) and consistently achieves the best performance among all the approaches on all the datasets. Because the marriage of SSL and LFM and the realization of the confidence-aware joint-smoothness.
- The sparser the dataset is, the bigger improvement of our model against the three comparison methods. Take $K = 200$ for example, the average *Precision* improvement of our approach over other models are 9.82%, 26.12%, and 35.58% on *Lastfm*, *Lastfm20*, and *Lastfm50* respectively. This is because our approach benefits from confidence-aware rating smoothness we use on the affinity graphs, which alleviates the overly strong assumption of LFR.

7.4 Compare Pairwise and Joint Smoothness

We then study the runtime performance (including graph build time and model inference time) and prediction performance between pairwise and joint smoothness. As analysied in Section 5.1,



(a) *Lastfm-20%* K=100 (b) *Lastfm-20%* K=150 (c) *Lastfm-20%* K=200

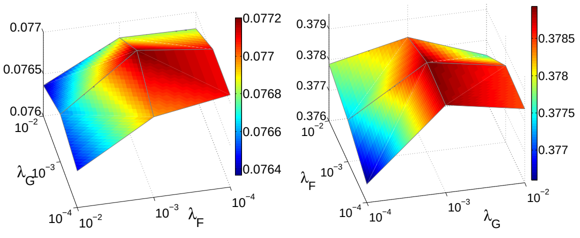
(d) *Lastfm-50%* K=100 (e) *Lastfm-50%* K=150 (f) *Lastfm-50%* K=200

Fig. 5. *Precision* comparison of each method on *Lastfm-20%* and *Lastfm-50%*.

Table 6. Performance comparison between pairwise and joint smoothness on *Movielens2K*.

Metric	MAE	RMSE	runtime (seconds)
pairwise1	0.7071	0.9170	337
pairwise2	0.6889	0.8916	157
joint	**0.6185**	**0.8188**	**5**

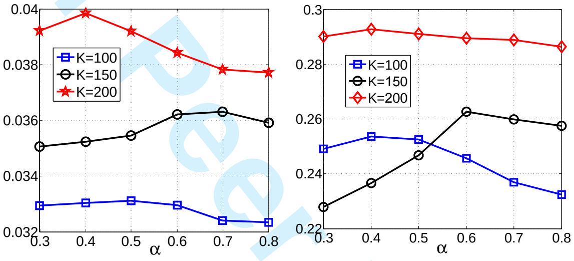
building a pairwise affinity graph is quite time-consuming, i.e., Challenge $\mathcal{II}$. To compare pairwise and joint smoothness, we use hetrec2011-movielens-2k dataset (*MovieLens2K*) [7], which is a relative small dataset. *MovieLens2K* contains 2,113 users, 10,197 items, and 855,598 U-I rating pairs. We then use different smoothness objective function, i.e., the pairwise smoothness shown in Eq. (9) and the joint smoothness shown in Eq. (14), in our proposed model. We term joint smoothness "joint" here for simplification. We also use two kinds of approaches to build the pairwise affinity graph, i.e., (1) "pairwise1": $P_{ij,ko} = W_{ij} * S_{ko}$, and (2) "pairwise2": $P_{ij,ko} = min\{W_{ij}, S_{ko}\}$. We finally compare their runtime and prediction differences on *MovieLens2K*. The results are shown in Table 6, where we set $K = 6$. As we analyzed in Section 5.1, the runtime of joint smoothness is significantly shorter than pairwise smoothness. Besides, the prediction performance of joint smoothness also outperforms pairwise smoothness. This is because joint smoothness uses two parameters (λ_F and λ_G) to control the global smoothness degree on $\mathcal{G}_1$ and $\mathcal{G}_2$ separately. In contrast, pairwise smoothness uses only one parameter (λ_P) to control the global smoothness degree on $\mathcal{G}$. Thus, joint smoothness can leverage smoothness degree more delicately on affinity graphs.



(a) *Precision@50* (b) *Recall@50*

Fig. 6. Effect of smoothness degree parameters λ_F and λ_G on RSCGM. Dataset used: *Delicious*.

7.5 Effect of Model Parameters



(a) *Precision@50* (b) *Recall@50*

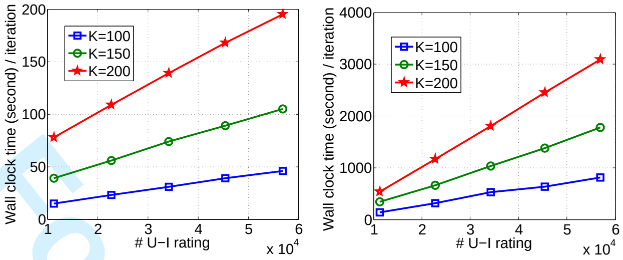
Fig. 7. Effect of smoothness confidence decay parameters α on RSCGM. Dataset used: *Lastfm*.

We first study the effect of the smoothness degree parameters, i.e., λ_F and λ_G, on our model performance. By doing this, we set $\alpha = 1$, $d = 0$, which means that we only propagate the known ratings to their direct U-I pairs. Figure 6 shows the effect of λ_F and λ_G on RSCGM, where we set $K = 100$. We can see that RSCGM achieves the best *Precision* and *Recall* performance when $\lambda_F = \lambda_G = 0.001$ on *Delicious*. The results indicate that smoothness on both user and item affinity graphs contribute to model performance, which proves the effectiveness of our joint-smoothness idea.

We then study the effect of the smoothness confidence decay parameter, i.e., α, on our model performance. By doing this, we set λ_F and λ_G to the best values obtained above. Figure 7 shows the effect of α on RSCGM. From it, we can see that, with the increase of α, the performance first increases, and then decreases after a certain threshold. The explanation is: (1) when α is small, the smoothness confidence $\alpha^{|d|+1}$ will be too small to propagate ratings. That is, known ratings will only be propagated to short path neighbors and this may cause information loss; (2) On the contrary, with a big α, known ratings will be propagated to their long path neighbors, and this may cause data noise due to the unreliable affinity graphs.

7.6 Computation Time

We run our model on a PC with 2.33 GHz Intel(R) Xeon(R) CPU and 8Gb RAM. Figure 8 shows the wall-clock running time per iteration of our proposed model with different size of training data on the *Lastfm*. The same as we analyzed in Section 6, the runtime does increase linearly with the observed data size. Besides, further observation shows that, the bigger K is, the bigger the runtime increase rate is. This is because the runtime can be seen as a linear function of the observed data size, and its slope is $(\sum \overline{F} + \sum \overline{G})K$. With a relative stable value of $\sum \overline{F} + \sum \overline{G}$, the bigger K is, the bigger the increase slope is.



(a) $\alpha = 0.3$

(b) $\alpha = 0.4$

Fig. 8. Wall-clock time with different size of training data on *Lastfm*.

8 CONCLUSION

In this paper, we have proposed a probabilistic chain graph models to marry SSL with LFM to improve recommendation performance by alleviating the data sparsity problem. The proposed CGM is a combination of Bayesian network and Markov random field. We use the dimensionality reduction idea of LFMs in the Bayesian network, and use the smoothness idea of SSL in Markov random field. We have proposed to perform joint smoothness instead of pairwise smoothness to save affinity graph build time. We also have proposed a confidence-aware approach to realize joint-smoothness to address the affinity unreliable problem in RS. Our proposed approach realized the ideas of both SSL and LFM, and addresses the challenges of adopting SSL in RS, and thus possesses the merits of both LFM and SSL. We have conducted experiments on three popular real world datasets, and the experimental results showed that our approach significantly outperforms the state-of-the-art recommendation approaches, especially in data sparsity scenarios.

REFERENCES

[1] Deepak Agarwal and Bee-Chung Chen. 2009. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 19–28.

[2] Deepak Agarwal and Bee-Chung Chen. 2010. fLDA: matrix factorization through latent dirichlet allocation. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*. 91–100.

[3] James Bennett and Stan Lanning. 2007. The netflix prize. In *Proceedings of KDD Cup and Workshop 2007 Aug 12*, Vol. 2007. 35.

[4] Alex Beutel, Ed H Chi, Zhiyuan Cheng, Hubert Pham, and John Anderson. 2017. Beyond Globally Optimal: Focused Learning for Improved Recommendations. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 203–212.

[5] Christopher M Bishop. 2006. *Pattern recognition and machine learning.*

[6] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3 (2003), 993–1022.

[7] Ivan Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2011. Second workshop on information heterogeneity and fusion in recommender systems (HetRec2011).. In *In Proceedings of the 5th ACM conference on Recommender Systems*. 387–388.

[8] Olivier Chapelle, Bernhard Schölkopf, Alexander Zien, et al. 2006. *Semi-supervised learning.* MIT press Cambridge.

[9] Chaochao Chen, Xiaolin Zheng, Yan Wang, Fuxing Hong, and Zhen Lin. 2014. Context-aware Collaborative Topic Regression with Social Matrix Factorization for Recommender Systems. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. 9–15.

[10] Chris Ding, Horst D Simon, Rong Jin, and Tao Li. 2007. A learning framework using Green's function and kernel regularization with application to recommender system. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 260–269.

[11] Paul Dirac. 1958. *The principles of quantum mechanics*. Oxford university press.

[12] Pedro Domingos and Matt Richardson. 2001. Mining the network value of customers. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 57–66.

[13] Yuan Fang, Kevin Chen-Chuan Chang, and Hady Wirawan Lauw. 2014. Graph-based Semi-supervised Learning: Realizing Pointwise Smoothness Probabilistically. In *Proceedings of the 31st International Conference on Machine Learning*.

[14] Shanshan Feng, Jian Cao, Jie Wang, and Shiyou Qian. 2017. Recommendations Based on Comprehensively Exploiting the Latent Factors Hidden in ItemsâĂŹ Ratings and Content. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 11, 3 (2017), 35.

[15] Quanquan Gu, Jie Zhou, and Chris HQ Ding. 2010. Collaborative Filtering: Weighted Nonnegative Matrix Factorization Incorporating User and Item Graphs. In *Proceedings of the Tenth SIAM Conference on Data Mining*. 199–210.

[16] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. 2016. A Novel Recommendation Model Regularized with User Trust and Item Ratings. *IEEE Transactions on Knowledge and Data Engineering* 28, 7 (2016), 1607–1620.

[17] F Maxwell Harper and Joseph A Konstan. 2016. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TIIS)* 5, 4 (2016), 19.

[18] Antonio Hernando, Jesús Bobadilla, and Fernando Ortega. 2016. A non negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model. *Knowledge-Based Systems* 97 (2016), 188–202.

[19] Mohsen Jamali and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*. 135–142.

[20] Meng Jiang, Peng Cui, Fei Wang, Wenwu Zhu, and Shiqiang Yang. 2014. Scalable recommendation with social contextual information. *IEEE Transactions on Knowledge and Data Engineering* 26, 11 (2014), 2789–2802.

[21] Bhargav Kanagal, Arif Ahmed, Shishir Pandey, Vanja Josifovski, Lluis Garcia-Pueyo, and Jiaxin Yuan. 2013. Focused matrix factorization for audience selection in display advertising. In *Proceedings of the IEEE 29th International Conference on Data Engineering (ICDE)*. 386–397.

[22] Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models - Principles and Techniques.* MIT Press. 161–168 pages.

[23] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 426–434.

[24] Steffen L Lauritzen and Thomas S Richardson. 2002. Chain graph models and their causal interpretations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 64, 3 (2002), 321–348.

[25] Xinyue Liu, Charu Aggarwal, Yu-Feng Li, Xiangnan Kong, Xinyuan Sun, and Saket Sathe. 2016. Kernelized matrix factorization for collaborative filtering. In *SIAM Conference on Data Mining*. 399–416.

[26] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. 2011. Recommender systems with social regularization. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*. 287–296.

[27] Lester Mackey. 2007. *Latent Dirichlet Markov Random Fields for Semi-supervised Image Segmentation and Object Recognition.* Technical Report. Citeseer.

[28] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*. 165–172.

[29] Calvin McCarter and Seyoung Kim. 2014. On Sparse Gaussian Chain Graph Models. In *Advances in Neural Information Processing Systems (NIPS 2014)*. 3212–3220.

[30] Andriy Mnih and Ruslan Salakhutdinov. 2007. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems (NIPS 2007)*. 1257–1264.

[31] Sanjay Purushotham, Yan Liu, and C-C Jay Kuo. 2012. Collaborative Topic Regression with Social Matrix Factorization for Recommendation Systems. *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)* (2012).

[32] Nikhil Rao, Hsiang-Fu Yu, Pradeep K Ravikumar, and Inderjit S Dhillon. 2015. Collaborative filtering with graph information: Consistency and scalable methods. In *Advances in Neural Information Processing Systems*. 2107–2115.

[33] Steffen Rendle. 2010. Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining*. 995–1000.

[34] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2000. *Application of dimensionality reduction in recommender system-a case study.* Technical Report. DTIC Document.

[35] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*. 285–295.

[36] Matthias Seeger. 2002. *Learning with labeled and unlabeled data.* Technical Report. The University of Edinburgh.

[37] Yue Shi, Martha Larson, and Alan Hanjalic. 2014. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)* 47, 1 (2014), 3.

[38] Xiaoyuan Su and Taghi M Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence* 2009 (2009), 4.

[39] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. 2008. Investigation of various matrix factorization methods for large recommender systems. In *2008 IEEE International Conference on Data Mining Workshops*. IEEE, 553–562.

[40] Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 448–456.

[41] Dingyan Wang and Irwin King. 2010. An Enhanced Semi-supervised Recommendation Model Based on Green's Function. In *International Conference on Neural Information Processing*. 397–404.

[42] Fei Wang, Sheng Ma, Liuzhong Yang, and Tao Li. 2006. Recommendation on item graphs. In *Proceedings of the Sixth International Conference on Data Mining (ICDM'06)*. 1119–1123.

[43] Jingwei Xu, Yuan Yao, Hanghang Tong, Xianping Tao, and Jian Lu. 2015. Ice-breaking: Mitigating cold-start recommendation problem by rating comparison. In *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI Press, 3981–3987.

[44] Surong Yan, Kwei-Jay Lin, Xiaolin Zheng, Wenyu Zhang, and Xiaoqing Feng. 2017. An Approach for Building Efficient and Accurate Social Recommender Systems using Individual Relationship Networks. *IEEE Transactions on Knowledge and Data Engineering* (2017).

[45] Xiaojin Zhu and Zoubin Ghahramani. 2002. *Learning from labeled and unlabeled data with label propagation*. Technical Report. Citeseer.

[46] Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*. 912–919.

Received July 2017; revised Dec 2017; accepted Dec 2017