Learning Informative Priors from Heterogeneous Domains to Improve Recommendation in Cold-Start User Domains

LIANG HU, University of Technology, Sydney and Shanghai Jiao Tong University
LONGBING CAO, University of Technology, Sydney
JIAN CAO, Shanghai Jiao Tong University
ZHIPING GU, Shanghai Technical Institute of Electronics & Information
GUANDONG XU, University of Technology, Sydney
DINGYU YANG, Shanghai Dian Ji University

In the real-world environment, users have sufficient experience in their focused domains but lack experience in other domains. Recommender systems are very helpful for recommending potentially desirable items to users in unfamiliar domains, and cross-domain collaborative filtering is therefore an important emerging research topic. However, it is inevitable that the cold-start issue will be encountered in unfamiliar domains due to the lack of feedback data. The Bayesian approach shows that priors play an important role when there are insufficient data, which implies that recommendation performance can be significantly improved in cold-start domains if informative priors can be provided. Based on this idea, we propose a Weighted Irregular Tensor Factorization (WITF) model to leverage multi-domain feedback data across all users to learn the cross-domain priors w.r.t. both users and items. The features learned from WITF serve as the informative priors on the latent factors of users and items in terms of weighted matrix factorization models. Moreover, WITF is a unified framework for dealing with both explicit feedback and implicit feedback. To prove the effectiveness of our approach, we studied three typical real-world cases in which a collection of empirical evaluations were conducted on real-world datasets to compare the performance of our model and other state-of-the-art approaches. The results show the superiority of our model over comparison models.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*; I.2.6 [**Artificial Intelligence**]: *Learning*

General Terms: Algorithms

Additional Key Words and Phrases: Recommender systems, cross-domain collaborative filtering, weighted irregular tensor factorization, probabilistic matrix factorization, multi-task learning

ACM Reference Format:
Liang Hu, Longbing Cao, Jian Cao, Zhiping Gu, Guandong Xu, and Dingyu Yang. 2016. Learning informative priors from heterogeneous domains to improve recommendation in cold-start user domains. ACM Trans. Inf. Syst. 35, 2, Article 13 (December 2016), 37 pages.
DOI: http://dx.doi.org/10.1145/2976737

This work is partially supported by China National Science Foundation (Granted Number 61272438 and 61472253), Research Funds of Science and Technology Commission of Shanghai Municipality (Granted Number 15411952502 and 14511107702), and Australia Research Council Linkage Project (LP140100937).

Authors' addresses: L. Hu (corresponding author), Advanced Analytics Institute, University of Technology, Sydney, Australia and Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China; email: rainmilk@gmail.com; L. Cao, Advanced Analytics Institute, University of Technology, Sydney, Australia; email: longbing.cao@uts.edu.au; J. Cao (corresponding author), Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China; email: cao-jian@sjtu.edu.cn; Z. Gu, Department of Electronics Engineering, Shanghai Technical Institute of Electronic Information, Shanghai, China; email: guzhiping@stiei.edu.cn; G. Xu, Advanced Analytics Institute, University of Technology, Sydney, Australia; email: guandong.xu@uts.edu.au; D. Yang, School of Electronics and Information, Shanghai Dian Ji University, Shanghai, China; email: yangdy@sdju.edu.cn.

DOI: http://dx.doi.org/10.1145/2976737

1. INTRODUCTION

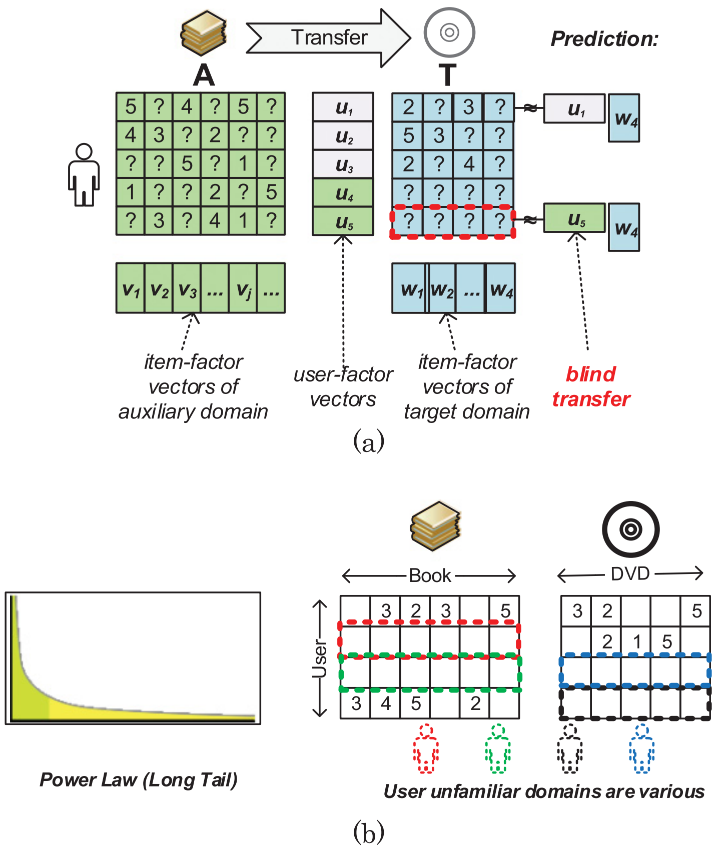
In the era of Big Data, the huge and rapidly increasing amount of information has penetrated every corner of our life. However, it is easy to become overwhelmed by so much information and it can be difficult to find what one is looking for. When we follow events on Facebook, buy books on Amazon, or install apps on a smartphone, we are encouraged by the underlying systems to provide feedback, e.g., a rating or a comment. This is because modern recommender systems can predict personalized preferences for unconsumed items based on the feedback collected from like-minded users. Collaborative filtering (CF) has been widely studied as a core component of recommender systems, but in fact, users do not always provide feedback for various personal reasons, e.g., privacy. As a result, CF methods tend to suffer from two common issues: data sparsity [Su and Khoshgoftaar 2009] and cold-start [Koren et al. 2009; Schein et al. 2002]. Some real-world applications naturally suffer from the data sparsity problem; for instance, users who have recently bought a new car may not have a new car purchase plan for another five years, therefore the lack of feedback data becomes a major barrier to the use of current CF methods.

A user usually has sufficient experience in some focused domains (rich data domains) but lacks experience in other domains (deficient data domains). A recommender system that can recommend potentially desirable items to users is therefore more useful in unfamiliar domains. However, it is inevitable that the cold-start issue will be encountered in unfamiliar domains where there is almost no data from which to learn user preferences. Since users have different interests, their rich data domains and deficient data domains also differ, and it is therefore sensible to leverage users' feedback data over multiple domains to find like-minded users to enable the inference of user preferences in unfamiliar domains. Based on this idea, Cross-Domain Collaborative Filtering (CDCF) has emerged as an important research topic in recent years [Li 2011]. Most current CDCF approaches focus on the product domain [Li et al. 2009a, 2009b; Pan et al. 2010], but although the term "domain" usually refers to product domains, it may apply to more generalized references, such as time domains and spatial domains.

1.1. Current Methods Leveraging Cross-Domain Information

Collaborative filtering (CF) methods can generally be sub-divided into two categories: neighborhood-based (a.k.a. memory-based) and model-based [Hofmann 2004; Resnick et al. 1994; Su and Khoshgoftaar 2009]. Therein, the model-based approach, such as matrix factorization (MF) [Hu et al. 2008; Koren et al. 2009], have gained dominance in recent years. In fact, we can construct an integrated item set containing the items from all domains so that traditional CF approaches can be directly applied as naïve CDCF approaches. In this article, we refer to neighborhood-based methods and MF-based methods running on such integrated item sets as kNN-CDCF and MF-CDCF, respectively. The item factors affecting user preferences in one domain may be quite different from those in another, but taking the integrated item set as input implicitly assumes the homogeneity of items as a single domain. As a result, the naïve CDCF approach may lead to poor prediction due to the failure of representing heterogeneities between domains.

A number of refined CDCF methods have recently been proposed, such as Cross-Domain Matrix Factorization (CDMF) models [Pan et al. 2010; Singh and Gordon 2008]. CDMF is based on transfer learning, whose underlying idea is illustrated in Figure 1(a): the user factor matrix $\boldsymbol{U}$ serves as a bridge to transfer knowledge from the auxiliary domain (A) to the target domain (T). CDMF models assume that auxiliary data is relatively dense for all users and all items [Pan et al. 2010]. However, we argue that this assumption is not always true. Our argument is based on the well-known



(a)

(b)

Fig. 1. (a) Demonstration of the occurrence of the blinder-transfer issue in CDMF; (b) The feedback from the majority of users in each domain is scant due to the power law distribution, and users have different unfamiliar domains due to differences in interests.

power law (long-tail) distribution, as illustrated in Figure 1(b), where the minority of users and items provide sufficient data while the majority of users and items provide only a little data. This has an impact on the hypothesis of traditional CDCF approaches, which results in the deterioration of prediction performance.

A worse, unavoidable problem of CDCF may be caused by the cold-start issue in some domains. Since users have quite different interests, a user is usually active in certain domains but silent in other domains. As shown in Figure 1(a), users always have different unfamiliar domains which may negatively reduce the recommendation performance of CDMF models due to the heterogeneities between domains. If we take a close look at how this happens in terms of Figure 1(b), we see that CDMF aims to improve recommendation on the target domain (T) by utilizing the features of user preferences (i.e., the factor matrix $\boldsymbol{U}$) learned from the auxiliary domain (A). A new user factor matrix $\boldsymbol{U}'$, which models the user preferences on the target domain (T), is updated based on the transferred matrix $\boldsymbol{U}$ using the data on (T) [Pan et al. 2010; Singh and Gordon 2008]. Therefore, the user-factor vectors for users are co-determined by the feedback in the auxiliary and target domains. If no data is available for a user in the target domain (marked with a red box), the user factor vector $\boldsymbol{u}_i$ is simply transferred from the auxiliary domain without updating. As a result, the prediction on the target domain tends to yield poor recommendation results using this $\boldsymbol{u}_i$ because of

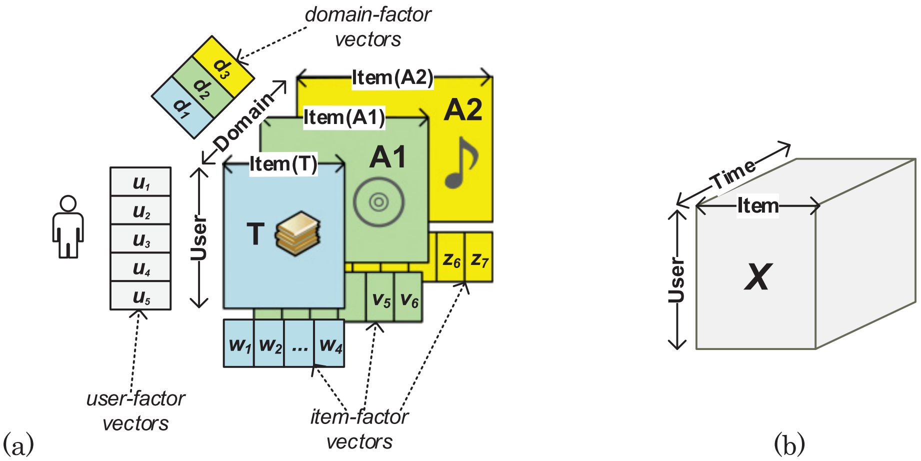


Fig. 2. (a) Irregular triadic relation for modeling heterogeneities between domains; (b) A regular triadic relation "*user-item-time*" represented by a 3D-tensor.

the heterogeneities between two domains. In this article, we call this a "*blind-transfer*" issue. Recall that the data associated with the majority of users are insufficient and even absent in a domain, so the above CDCF approaches commonly suffer from such blind-transfer issues for realistic online data.

1.2. Modeling Domain Factors

The major cause of the blind-transfer issue is the fact that CDMF deals with a set of user-item data over multiple domains in the traditional MF manner so that the rating given by user i on item j is only determined by user and item factors, i.e., $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$, but it does not contain the factors to model the difference between multiple domains. For this reason, CDMF cannot escape the blind-transfer issue, especially when the data is extremely sparse. We argue that domain factors are an essential element for representing each domain in the "*cross-domain*" problem, so CDCF should take into consideration the domain characteristics to reveal domain-specific user preferences on items in depth, rather than only user factors and item factors modeled in CDMF.

As illustrated in Figure 2(a), our approach allows an exclusive item-factor matrix for each domain to express heterogeneities. In addition, user-factor matrix $\boldsymbol{U}$ is used to model general users' concerns across all domains, and domain-factor matrix $\boldsymbol{D}$ carries the information to express the traits of each domain. Hence, each observation can be viewed as the result of the three-way interaction among user, item, and domain factors. In addition, we can interpret that the domain-specific user factors are generated by the interaction between domain factors and general user factors as shown in Figure 2(a). Since the domain-factor vector reflects the characteristics of a domain and it is always available, thus the domain-specific user preferences can be obtained to avoid the blind-transfer issue.

1.3. Irregular Triadic Relation

According to above analysis, MF-based cross-domain methods, i.e., CDMF, can only model the dyadic interaction between users and items, so they are inevitable to suffer from the blind-transfer issue. As a result, we propose to model three-way interaction among user, item, and domain to avoid this issue. A natural approach to model high-order interaction, e.g., *user-item-tag*, *user-item-time*, is in terms of tensor factorization (TF) models [Rendle et al. 2009b; Xiong et al. 2010]. For such regular triadic relations, let us denote the user set $\mathcal{U}$, the item set as $\mathcal{I}$ and $\mathcal{D}$ is the set for the third dimension, i.e., a triadic relation $\mathcal{U} \times \mathcal{I} \times \mathcal{D}$, so they can be naturally represented by a 3D-tensor as demonstrated in Figure 2(b). Unfortunately, we cannot use a regular tensor to deal with CDCF problems, since each domain d has a different domain-specific item set, $\mathcal{I}_d$ as shown in Figure 2(a). Therefore, we cannot obtain a regular triadic relation over the sets, $\mathcal{U}$, $\mathcal{D}$ and $\{\mathcal{I}_d\}_{d\in\mathcal{D}}$. As a result, regular TF models become not applicable.

In this article, we design an irregular TF model, named Weighted Irregular Tensor Factorization (WITF), to model the irregular triadic relation. As implied by its name, WITF couples a set of $\{\mathcal{U} \times \mathcal{I}_d\}_{d\in\mathcal{D}}$ relations among multiple domains and derives an optimization form consisting of a TF-equivalent component which enables WITF to capture the irregular triadic interaction and learn the domain factors.

1.4. Contributions

In this article, we aim to design a more robust CDCF method to deal with cold start in domains that are unfamiliar to users, to improve the recommendation. The main contributions of our work are summarized as follows:

—We present the emerging requirement to leverage the data from multiple domains to improve the cold-start issue in recommender systems. Furthermore, we analyze the deficiencies of current CDCF methods caused by the heterogeneities between domains. We especially address the irregular triadic relation in CDCF, where each domain has different item set. As a result, it becomes unfeasible to apply regular tensor factorization approaches. To deal with this challenge, we design a Weighted Irregular Tensor Factorization (WITF) model to couple the data among multiple heterogeneous domains.

—Subjective preference data, e.g., ratings, are not always available in many real-world scenarios in recommender systems, while objective preference data, such as purchase history, browsing behavior, and click logs, are much more easily obtained. The proposed learning algorithm for WITF can deal with both explicit preference data and implicit preference data in a unified way.

—We propose a transfer learning method in terms of a Weighted Regularized Matrix Factorization (WRMF) model, where the user factors, item factors and domain factors learned from WITF serve as informative cross-domain priors to regularize latent factor learning in WRMF. These informative priors enable WRMF to infer user preferences in cold-start domains.

—We perform a collection of experiments on three real-world cases and make comparisons with other state-of-the-art methods to test the effectiveness of our approach. These are: (1) multi-category product recommendation in an e-business site; (2) multi-category attractive item recommendation in a social networking site based on implicit feedback; (3) movie recommendation over long time periods. All the evaluation results prove that our CDCF approach significantly outperforms comparison methods.

2. RELATED WORK

Our work aims to improve recommender systems in terms of leveraging information between multiple domains. We therefore first provide a brief review of current recommender systems, and then discuss several state-of-the-art CDCF models. We also present high-order relation modeling in recommender systems, which relates strongly to our work.

2.1. Recommender Systems

The origin of collaborative filtering (CF) systems is found in the neighborhood-based approach [Su and Khoshgoftaar 2009], which has been successfully applied in a number of real-world commercial systems [Sarwar et al. 2001]. In general, it can be sub-divided into two categories: user-based nearest neighbor and item-based nearest neighbor [Su and Khoshgoftaar 2009]. However, this approach does not work well when the data is sparse. In fact, the majority of users provide very little data due to long-tail distribution [Hu et al. 2014], so it is often not possible to obtain feedback from neighbors on unpopular items to generate the prediction result.

With the rapid development of machine learning, model-based approaches have become more and more popular in recent years. Matrix factorization models, in particular, have gained dominance in the recommendation area and have demonstrated their superiority over neighborhood-based techniques by winning the Netflix Prize competition [Koren et al. 2009]. The basic idea of MF methods is to fit the user-item rating matrix using low-rank approximations and use it for prediction. To date, many matrix factorization methods have been proposed, such as Probabilistic MF (PMF) [Salakhutdinov and Mnih 2008] and Maximum-Margin MF [Srebro et al. 2005]. Apart from the MF approach, other models have also achieved success in recommendation. With the prevalence of Deep Learning techniques [Bengio et al. 2013], Restricted Boltzmann Machines have been successfully applied in collaborative filtering [Georgiev and Nakov 2013] and have achieved comparable performance to MF in the Netflix Prize competition [Salakhutdinov et al. 2007]. Choice modeling [Train 2003] is somewhat related to the recommendation problem, Hu et al. [2014] proposed a latent feature-based Bayesian Heteroscedastic Choice Model to represent the heterogeneities between users and items.

Apart from rating-oriented systems, more recent research has paid more attention to ranking-oriented recommender systems. EigenRank [Liu and Yang 2008] addresses the item ranking problem directly by modeling user preferences derived from the ratings. It ranks items based on the preferences of similar users, where the similarity is measured by the correlation between users' rankings of the items rather than the rating values. Bayesian personalized ranking (BPR) [Rendle et al. 2009a] assumes that users will express stronger likes for their chosen items than for unobserved items so the preference ordering relation can be constructed for each pair of items. As a result, BPR learns the utility of choosing an item from the ordering relationships. ListCF [Huang et al. 2015] is a memory-based CF method which directly predicts a total order of items for each user based on their similar users' probability distributions over permutations of the items.

Most current recommender systems are built on subjective feedback, e.g., ratings, to differentiate user preferences. However, subjective feedback is not always available, while objective feedback, e.g., click logs, is more easily obtained [Cao 2015]. The implicit preference data is often represented by binary values, that is, 1 for observed choices and 0 for others [Hu et al. 2014; Pan et al. 2008]. These unobserved choices have zero values, but they often do not represent true negative instances. Therefore, an often-used strategy is to assign larger confidence to observed choices to represent the high certainty of users' explicit likes, i.e., true positive instances, whereas a much smaller confidence is assigned to unobserved choices to represent the small likelihood of dislike, i.e. uncertainly negative instances [Hu et al. 2014, 2008; Pan et al. 2008].

These methods cannot fully tackle the cold-start case, however, because there is no observed data to differentiate between user preferences. They must therefore incorporate additional side information [Agarwal and Chen 2009; Porteous et al. 2010] to infer user preferences.

2.2. Cross-Domain Collaborative Filtering

Codebook Transfer [Li et al. 2009a] assumes that cluster-level rating patterns, which are represented by a codebook, can be found between the rating matrices in two related domains. The Rating-Matrix Generative Model [Li et al. 2009b] extends this idea with a probabilistic model to solve collective transfer learning problems. In reality, there are many cold-start users for most domains due to the power law. Therefore, it is always out of the question to use this model in unfamiliar domains because no data is available to match common patterns with auxiliary domains.

Coordinate System Transfer (CST) [Pan et al. 2010] is a typical CDMF model. It learns the user-factor matrix $\boldsymbol{U}_A$ from an auxiliary rating matrix in the first step, and the user-factor matrix $\boldsymbol{U}_T$ of the target domain is then updated based on $\boldsymbol{U}_A$, with the regularization in terms of penalizing the divergence between $\boldsymbol{U}_A$ and $\boldsymbol{U}_T$. Dual Transfer Learning (DTL) [Long et al. 2012] is formulated as an optimization problem of joint nonnegative matrix tri-factorizations. It exploits the duality between the marginal distribution and conditional distribution to achieve effective transfer. Given the observations of source and target domains, marginal distribution is associated with the common latent features over all domains, and conditional distribution is associated with the domain-specific latent features. Since CST and DTL are modeled on the problem of transferring knowledge from a source domain to a target domain, they cannot be directly applied to multiple domains (more than two) as studied in this article.

Since user preference is not exclusive to a single domain, a straightforward method is to transfer knowledge through the user-factor matrix. Collective matrix factorization (CMF) [Singh and Gordon 2008] couples the target domain matrix and all auxiliary domain matrices on the *user* dimension, to share the user factor matrix across all domains. Moreover, CMF assigns a weight to the loss of fitting each domain matrix so that it can control the amount of influence from each domain; however, it does not provide a mechanism for finding an optimal weight assignment. Loni et al. [2014] presented an approach that encodes rating matrices from multiple domains as real-valued feature vectors $\mathbf{x}[u, i; s_2(u), \ldots, s_D(u)]$, where u, i index a rating given by user u on item i of target-domain and $s_d(u)$ indexes all rescaled ratings on auxiliary domain d. With these vectors, an algorithm based on factorization machines [Rendle 2010] is employed to find patterns between target and auxiliary domains. However, for a cold-start user without any rating on target domain, no data is available to encode such a feature vector $\mathbf{x}$ for this user to match patterns. Therefore, this method is not suitable to work in a cold-start environment as studied in this paper. Moreover, this model is built on rating data whereas it is not defined how to deal with implicit preference data.

As illustrated in the Introduction, all of the above models do not resemble the heterogeneities between domains so they inevitably suffer from the blind-transfer issue when a user makes a cold start in unfamiliar domains.

2.3. High-Order Relation Modeling

The MF-based approach is able to model a collection of dyadic relations but it has limitations in representing high-order, i.e., multi-way, interactions. A natural approach to modeling a high-order relation is in terms of tensor which extends the matrix (second order) model to a higher order space. Karatzoglou et al. [2010] studied context-aware collaborative filtering using a High Order SVD (HOSVD) model to represent the relation of *user-item-context*, where each attribute of the context is modeled as a mode of a regular tensor. Tag recommendation is a problem that is widely studied in Web 2.0, and Rendle et al. proposed BPR-based tensor factorization models on *user-item-tag* to learn optimal ranking [Rendle et al. 2009b; Rendle and Schmidt-Thieme 2010]. Time periods can be regarded as generalized domains [Dunlavy et al. 2011] which organize temporal observations according to time-period slices to conduct temporal link prediction.

However, we cannot employ a regular TF model to deal with CDCF problems as analyzed in the section of Introduction. Inspired by PARAFAC2 [Kolda and Bader 2009; Mørup 2011], Hu et al. [2013] proposed Cross-Domain Triadic Factorization (CDTF) model which transforms the optimization problem on domain slices into an equivalent tensor factorization problem. When dealing with missing values, both CDTF and PARAFAC2 use an imputation strategy [Srebro and Jaakkola 2003] to restore missing values by predictive ones for each iteration and thus need a full matrix to store the reconstructed data on each domain. As a result, the space complexity of CDTF and PARAFAC2 tends to be even larger when more domains are involved. To avoid this huge space complexity, the Weighted Irregular Tensor Factorization (WITF) model proposed in this article uses a newly designed learning algorithm to avoid imputation on the basis of weighting strategies, thus WITF only needs to store observed data when learning. Moreover, PARAFAC2 is inapplicable to implicit feedback. To work with this challenge, an algorithm is provided by CDTF to tackle with implicit feedback; however, this algorithm is different from the explicit feedback algorithm, and has defect to apply the weight on implicit feedback data. In comparison, the newly devised WITF resolves this defect and enables to model explicit feedback and implicit feedback in a unified way by applying different weight configurations. Therefore, all the cases of CDTF can be handled by WITF in refined ways. In addition, CDTF depends on a genetic algorithm to tune the model, which is extremely time-consuming. In this article, the parameters learned from WITF serve as the priors of a probabilistic MF model, and we design an efficient post-learning procedure to fine-tune these parameters. Thus, WITF offers a significant improvement on CDTF and PARAFAC2.

3. PROBLEM FORMULATION

In recent years, matrix factorization-based methods [Hu et al. 2008; Koren et al. 2009] have gained dominance in recommender systems. However, single-domain MF methods are not able to provide a good solution to deal with the cold start issue. In the following sections, we will take a close look at how MF methods suffer from this issue and propose our CDCF solution.

3.1. Weighted Regularized Matrix Factorization from Probabilistic View

Let us consider the MF model from a probabilistic view. Given a data matrix $\boldsymbol{Y}$ with the entries indexed by (i, j), we can obtain the following joint distribution with the R-dimensional Gaussian user factor vector $\boldsymbol{u}_i$ for each user i and item factor vector $\boldsymbol{v}_i$ for each item j:

$$P(\boldsymbol{u}_i) = N(\boldsymbol{u}_i|\boldsymbol{\mu}_i, \boldsymbol{\tau}_U^{-1}\boldsymbol{I}) \qquad P(\boldsymbol{v}_j) = N\left(\boldsymbol{v}_j|\boldsymbol{\mu}_j, \boldsymbol{\tau}_V^{-1}\boldsymbol{I}\right) \tag{1}$$

$$P\left(Y_{ij}|\boldsymbol{u}_i, \boldsymbol{v}_j\right) = N\left(Y_{ij}|\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{v}_j, w_{ij}^{-1}\right) \tag{2}$$

$$P(\boldsymbol{Y}, \boldsymbol{U}, \boldsymbol{V}) = \prod_{ij} P\left(\boldsymbol{Y}_{ij}|\boldsymbol{u}_i, \boldsymbol{v}_j\right) \prod_i P(\boldsymbol{u}_i) \prod_j P\left(\boldsymbol{v}_j\right) \tag{3}$$

$$P(\boldsymbol{U}, \boldsymbol{V}|\boldsymbol{Y}) = \frac{P(\boldsymbol{Y}, \boldsymbol{U}, \boldsymbol{V})}{P(\boldsymbol{Y})} \propto P(\boldsymbol{Y}, \boldsymbol{U}, \boldsymbol{V}) \tag{4}$$

where $\boldsymbol{I}$ denotes an identity matrix, $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N]$ is the user factor matrix,$\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_M]$ is the item factor matrix, and τ_U, τ_V, w_{ij} are the precision parameters of the Gaussian distributions. We can learn the user factors and the items factors through maximum a posteriori (MAP) estimate. According to the Bayesian theorem, we have the posterior $P(\boldsymbol{U}, \boldsymbol{V}|\boldsymbol{Y}) \propto P(\boldsymbol{Y}, \boldsymbol{U}, \boldsymbol{V})$ given in Equation (4). The following objective function can then be obtained by minimizing the negative log-posterior.

$$J = \underset{\boldsymbol{U},\boldsymbol{V}}{argmin} \frac{1}{2}\left[\sum_{ij} w_{ij}\left(Y_{ij} - \boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{v}_j\right)^2 + \tau_U \sum_i ||\boldsymbol{u}_i - \boldsymbol{\mu}_i||^2 + \tau_V \sum_j ||\boldsymbol{v}_j - \boldsymbol{v}_j||^2\right] \tag{5}$$

Thus, we obtain a Weighted Regularized Matrix Factorization (WRMF) model, in which the precision parameter w_{ij} serves as the weight w.r.t. each entry. Typically, if we set $\boldsymbol{\mu}_i = \mathbf{0}$, $\boldsymbol{\mu}_j = \mathbf{0}$ (i.e., zero-mean priors), $w_{ij} = 1$ for observed rating while $w_{ij} = 0$ otherwise, and $\lambda = \tau_U = \tau_V$ (i.e., regularization parameter), then we immediately obtain the objective of the classical Probabilistic Matrix Factorization (PMF) model [Salakhutdinov and Mnih 2008].

We can estimate user factors and item factors by the gradient-based method using a coordinate descent strategy. First, the gradient w.r.t. $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ can be easily derived from the objective (4):

$$\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{u}_i} = \sum_j w_{ij} \boldsymbol{v}_j \boldsymbol{v}_j^{\mathrm{T}} \boldsymbol{u}_i - \sum_j w_{ij} Y_{ij} \boldsymbol{v}_j - \tau_U \boldsymbol{\mu}_i + \tau_U \boldsymbol{u}_i \tag{6}$$

$$\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{v}_j} = \sum_i w_{ij} \boldsymbol{u}_i \boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{v}_j - \sum_i w_{ij} Y_{ij} \boldsymbol{u}_i - \tau_V \boldsymbol{\mu}_j + \tau_V \boldsymbol{v}_i \tag{7}$$

When $\boldsymbol{V}$ is fixed, the optimization w.r.t. each $\boldsymbol{u}_i$ is convex. Therefore, it yields a close-form update equation for $\boldsymbol{u}_i$ by setting Equation (6) to zero:

$$\boldsymbol{u}_i \leftarrow \left(\tau_U \boldsymbol{I} + \sum_j w_{ij} \boldsymbol{v}_j \boldsymbol{v}_j^{\mathrm{T}} \right)^{-1} \left(\tau_U \boldsymbol{\mu}_i + \sum_j w_{ij} Y_{ij} \boldsymbol{v}_j \right) \tag{8}$$

Similarly, we can obtain the update equation for each $\boldsymbol{v}_i$ with $\boldsymbol{U}$ being fixed:

$$\boldsymbol{v}_j \leftarrow \left(\tau_V \boldsymbol{I} + \sum_i w_{ij} \boldsymbol{u}_i \boldsymbol{u}_i^{\mathrm{T}} \right)^{-1} \left(\tau_V \boldsymbol{\mu}_j + \sum_i w_{ij} Y_{ij} \boldsymbol{u}_i \right) \tag{9}$$

Now, let us consider a fully cold-start case where user i has no data, i.e., $w_{ij} = 0$ for all j, and the user factor vector $\boldsymbol{u}_i$ is always the prior mean $\boldsymbol{\mu}_i$, since no data is available to update it. In such a cold-start case, the prediction on the preferences of user i is fully dependent on the given prior. For PMF, the prior is assumed to be zero-mean so it is unavailable to conduct recommendation for fully cold-start users.

3.2. Learning Priors from Cross-Domain Feedback

A straightforward way to resolve a cold-start problem when few data are available from the viewpoint of the Bayesian probabilistic model is to find more informative priors. Since we argue that it is possible to learn user preference by leveraging information from other domains, we propose to learn the priors over the cross-domain feedback data.

As analyzed previously, we argue that CDCF should take domain factors into consideration, thus the triadic *user-item-domain* relation needs to be modeled. Since MF methods can only model two-way interactions between users and items, they cannot capture high-order interaction across domains. We therefore need to construct a higher order latent factor model to capture the three-way interaction *user-item-domain*. Intuitively, the tensor factorization (TF) model [Kolda and Bader 2009] is a good candidate for representing this type of third-order interaction. A regular tensor requires each domain slice to have identical items. However, as illustrated in Figure 2, each domain slice has a domain-specific item set in the CDCF problem, therefore it cannot form a regular tensor for factorization. To work with this problem, we propose a Weighted Irregular Tensor Factorization (WITF) model which relaxes the constraint that dictates that the same item set should be employed for all domains. As a result, WITF respectively learns an item factor matrix for each domain, as shown in Figure 2.

3.3. Learning Posterior on Target Domain for Fine-Tuning

When WITF is learned, we can obtain the general user factors $\boldsymbol{u}_i$ for each user, the domain factors $\boldsymbol{d}_k$, and item factors $\boldsymbol{v}_j^k$ for each item in domain k, as shown in Figure 2. Although these factors can be used to reconstruct the entries for each domain [Hu et al. 2013], they may not perfectly predict the user preference in a target domain for recommendation because these estimates are retrieved by fitting the data over all domains. Therefore, we need a fine-tuning procedure to adjust those learned factors by tightly refitting the evidence in the target domain.

In Bayesian statistics, the posterior probability is the conditional probability that is assigned after the relevant evidence is taken into account. As presented previously, WRMF learns parameters from the data in a given domain, and the factors learned from WITF clearly serve as good informative priors for WRMF. Therefore, we can obtain the refined user factors and item factors in a target domain k by MAP estimation (c.f. Equations (4)∼(7)). The prior mean of user factors can be constructed in terms of the general user factors $\boldsymbol{u}_i$ and the domain factors $\boldsymbol{d}_k$, and the item factors $\boldsymbol{v}_j^k$ can directly serve as the prior mean of item factors of WRMF (cf. Equation (1)):

$$\boldsymbol{\mu}_i = diag(\boldsymbol{d}_k)\,\boldsymbol{u}_i \qquad \boldsymbol{\mu}_j = \boldsymbol{v}_j^k$$

$diag(\boldsymbol{d}_k)$ generates a diagonal matrix with the diagonal elements $\boldsymbol{d}_k$. This can be regarded as a transfer learning procedure which transfers the factors estimated from WITF as the priors to regularize the user factors and item factors when learning WRMF (cf. Equation (5)).

4. WEIGHTED IRREGULAR TENSOR FACTORIZATION

4.1. Preliminary

Before introducing the WITF model, we first need to define basic notations and operations for tensor factorization (TF) to simplify the presentation.

4.1.1. Notations and Operations. The order of a tensor is the number of dimensions, also known as ways or modes. In this article, tensors are denoted by boldface script letters, e.g., $\boldsymbol{\mathcal{X}}$. Matrices are denoted by boldface capital letters, e.g., $\boldsymbol{X}$. Vectors are denoted by boldface lowercase letters, e.g., $\boldsymbol{x}$. In addition, we denote the ith row of a matrix $\boldsymbol{X}$ as $\boldsymbol{X}_{i,:}$, the jth column as $\boldsymbol{X}_{:,j}$ and $\boldsymbol{X}_{i,j}$ for the entry (i, j). $\boldsymbol{X}_{(n)}$ denotes the nth mode matricizing operation which maps a tensor into a matrix, e.g., $\boldsymbol{X}_{(2)}$ represents the mapping $\boldsymbol{\mathcal{X}}^{I\times J\times K} \rightarrow \boldsymbol{X}_{(2)}^{J\times IK}$ [Kolda and Bader 2009]. $\otimes$ denotes the Kronecker product and $\odot$ denotes the Khatri-Rao product [Kolda and Bader 2009], e.g., we have $\boldsymbol{X} \odot \boldsymbol{Y} = [\boldsymbol{X}_{:,1} \otimes \boldsymbol{Y}_{:,1} \boldsymbol{X}_{:,2} \otimes \boldsymbol{Y}_{:,2} \ldots \boldsymbol{X}_{:,R} \otimes \boldsymbol{Y}_{:,R}]$. $\circledast$ denotes the Hadamard (element-wise) product. $\langle \boldsymbol{\mathcal{X}} \circledast \boldsymbol{\mathcal{Y}} \rangle = \sum_{i,j,k} \boldsymbol{\mathcal{X}}_{i,j,k} \boldsymbol{\mathcal{Y}}_{i,j,k}$ stands for the inner product and the norm of a tensor is defined as $\|\boldsymbol{\mathcal{X}}\| = \sqrt{\langle \boldsymbol{\mathcal{X}} \circledast \boldsymbol{\mathcal{X}} \rangle}$. Table I summarizes the notations and operators used in this article.

4.1.2. Regularized CP Model. There are a number of different TF models in the literature, such as the Tucker model and the CP model (canonical decomposition/parallel factor analysis (PARAFAC)) [Kolda and Bader 2009]. Here, we mainly present the CP model, which is the most widely used TF model, because it has a concise factorization form.

As shown in Figure 3, the CP model decomposes a tensor into a sum of rank-one components. For instance, given a third-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$, the factorization

Table I. Notations and Operations

Symbol	Description
$\boldsymbol{\mathcal{X}}$	A tensor (boldface script letter)
$\boldsymbol{\mathcal{X}}^{I\times J\times K}$	The dimensionality of each mode
$\boldsymbol{X}$	A matrix (boldface capital letter)
$\boldsymbol{X}_{i,:}$	The ith row of matrix $\mathbf{X}$
$\boldsymbol{X}_{:,j}$	The jth column of matrix $\mathbf{X}$
$\boldsymbol{X}_{i,j}$	The entry (i, j) of matrix $\mathbf{X}$
$\boldsymbol{x}$	A vector (boldface lower-case letter)
$\boldsymbol{X}_{(n)}$	Mode-n matricization of a tensor
$\otimes$	Kronecker product
$\odot$	Khatri-Rao product
$\circledast$	Hadamard product
$\oslash$	Element-wise division
$\circ$	Outer product
$\|\boldsymbol{\mathcal{X}}\|$	The norm of tensor $\boldsymbol{\mathcal{X}}$
$vec\,(\boldsymbol{X})$	Vectorization of matrix $\mathbf{X}$

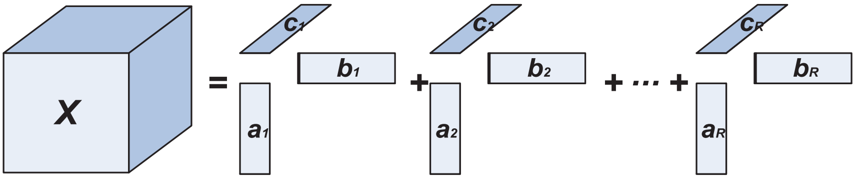


Fig. 3. The demonstration of CP factorization on a third-order tensor.

form of $\boldsymbol{\mathcal{X}}$ can be written as $\boldsymbol{\mathcal{X}} = [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] = \sum_{r=1}^{R} \boldsymbol{A}_{:,r} \circ \boldsymbol{B}_{:,r} \circ \boldsymbol{C}_{:,r}$, where $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are R-dimensional latent factor matrices and $\circ$ denotes the outer product, i.e., the entries are computed $\boldsymbol{\mathcal{X}}_{i,j,k} = \sum_{r=1}^{R} \boldsymbol{A}_{i,r}\boldsymbol{B}_{j,r}\boldsymbol{C}_{k,r}$. We can formulate the problem of fitting $\boldsymbol{\mathcal{X}}$ as a least-squares optimization problem with the following objective function [Acar et al. 2009]:

$$J = \underset{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}}{argmin}\, \frac{1}{2}\left[||\boldsymbol{\mathcal{X}} - [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]||^2 + \lambda_A||\boldsymbol{A}||_F^2 + \lambda_B||\boldsymbol{B}||_F^2 + \lambda_C||\boldsymbol{C}||_F^2\right] \tag{10}$$

where regularization terms are added to avoid overfitting, $\|\cdot\|_F$ is the Frobenius norm and λ_A, λ_B, λ_C are regularization parameters. From Equation (10), the following gradient w.r.t. $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ can be derived:

$$\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{A}} = -\boldsymbol{X}_{(1)}\,(\boldsymbol{C} \odot \boldsymbol{B}) + \boldsymbol{A}(\boldsymbol{B}^T\boldsymbol{B} \circledast \boldsymbol{C}^T\boldsymbol{C} + \lambda_A\boldsymbol{I}) \tag{11}$$

$$\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{B}} = -\boldsymbol{X}_{(2)}\,(\boldsymbol{C} \odot \boldsymbol{A}) + \boldsymbol{B}(\boldsymbol{C}^T\boldsymbol{C} \circledast \boldsymbol{A}^T\boldsymbol{A} + \lambda_B\boldsymbol{I})$$

$$\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{C}} = -\boldsymbol{X}_{(3)}\,(\boldsymbol{A} \odot \boldsymbol{B}) + \boldsymbol{C}(\boldsymbol{A}^T\boldsymbol{A} \circledast \boldsymbol{B}^T\boldsymbol{B} + \lambda_C\boldsymbol{I})$$

Similar to learning WRMF, we can alternatively learn the parameters $\{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}\}$, i.e., update one of the parameters by fixing others [Hu et al. 2013].

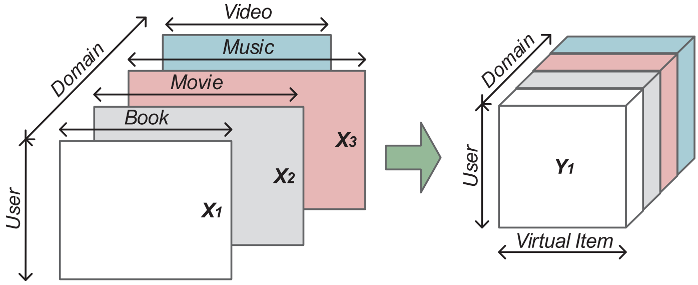


Fig. 4. WITF transforms the slices with heterogeneous domain-specific items into a regular third-order tensor containing an identical virtual item set.

4.2. Transformation

As previously discussed, we need to model the three-way interaction between *user-item-domain* in the CDCF problem to capture heterogeneities between domains. An intuitive way to capture this three-way interaction is in terms of a third-order tensor, where each frontal slice in the cube corresponds to a feedback data matrix over users and items for each domain. However, as illustrated in the left-hand image of Figure 4, each domain consists of a different number of domain-specific items, so it cannot form a regular tensor. Therefore, we need to transform the set of heterogeneous domain matrices into a regular tensor for the purpose of applying the TF. To work with this challenge, we designed a novel weighted irregular TF model that conducts a transformation over heterogeneous domain matrices into a regular tensor consisting of virtual items and virtual data, as illustrated in Figure 4.

Loss Function. Let $\{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_K\}$ denote the data matrices of all domains, where each matrix, $\boldsymbol{X}_k$, has the size $N \times M_k$, N is the number of users and M_k is the number of items in domain k. We denote $\boldsymbol{U} \in \mathbb{R}^{N\times R}$ as the user factor matrix ($\boldsymbol{U}_{i,:}$ refers to the factor vector of user i), $\boldsymbol{V}_k \in \mathbb{R}^{M_k\times R}$ is the item factor matrix of domain k ($\boldsymbol{V}_{k,j,:}$ refers to the factor vector of item j), and $\Sigma_k = diag(\boldsymbol{C}_{k,:})$ is an $R \times R$ diagonal matrix where $\boldsymbol{C}_{k,:}$ refers to the factor vector of the domain k. Given the user factor vector $\boldsymbol{U}_{i,:}$, the item factor vector $\boldsymbol{V}_{k,j,:}$ and the domain factor vector $\boldsymbol{C}_{k,:}$, the likelihood of entry $\boldsymbol{X}_{k,i,j}$ is given by:

$$P\left(\boldsymbol{X}_{k,i,j}|\boldsymbol{U}_{i,:}, \boldsymbol{V}_{k,j,:}, \boldsymbol{C}_{k,:}\right) = N\left(\boldsymbol{X}_{k,i,j}|\boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k\boldsymbol{V}^{\mathrm{T}}_{k,j,:}, w^{-1}_{k,i,j}\right)$$

Then, the likelihood over the entries over all domains can be given by:

$$P\left(\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_K|\boldsymbol{U}, \boldsymbol{V}_1, \boldsymbol{V}_2, \ldots, \boldsymbol{V}_K, \boldsymbol{C}\right) = \prod_{k=1}^{K}\prod_{i=1}^{N}\prod_{j=1}^{M_k} P\left(\boldsymbol{X}_{k,i,j}|\boldsymbol{U}_{i,:}, \boldsymbol{V}_{k,j,:}, \boldsymbol{C}_{k,:}\right) \quad (12)$$

We can easily obtain the following weighted loss function by minimizing the above negative log-likelihood:

$$J = \underset{\boldsymbol{U},\boldsymbol{V},\boldsymbol{C}}{argmin}\,\frac{1}{2}\sum_{k=1}^{K}||\boldsymbol{W}_k \circledast \left(\boldsymbol{X}_k - \boldsymbol{U}\boldsymbol{\Sigma}_k\boldsymbol{V}^{\mathrm{T}}_k\right)||^2_F \quad (13)$$

where the entry of the weight matrix is $\boldsymbol{W}_{k,i,j} = \sqrt{w_{k,i,j}}$. That is, each domain matrix $\boldsymbol{X}_k$ has the factorization form $\boldsymbol{U}\boldsymbol{\Sigma}_k\boldsymbol{V}^T_k$. However, this factorization form is not unique without additional constraints [Kiers et al. 1999]. For example, $\boldsymbol{U}\boldsymbol{\Sigma}_k\boldsymbol{V}^T_k = (\boldsymbol{U}\boldsymbol{\Sigma}_k\boldsymbol{A}^{-1}\boldsymbol{F}^{-1})\boldsymbol{F}(\boldsymbol{V}_k\boldsymbol{A})^{\mathrm{T}} = \boldsymbol{U}'\boldsymbol{F}\boldsymbol{V}'^{\mathrm{T}}$ where $\boldsymbol{F}$ is a diagonal matrix, so we get a new factorization form w.r.t. $\boldsymbol{X}_k$. To improve the uniqueness property, Harshman [1972] proposed the imposition of a constraint whereby the cross-product $\boldsymbol{V}_k^T \boldsymbol{V}_k$ is an invariant matrix over k, i.e., $\boldsymbol{\Phi} = \boldsymbol{V}_1^T \boldsymbol{V}_1 = \boldsymbol{V}_2^T \boldsymbol{V}_2 = \cdots = \boldsymbol{V}_K^T \boldsymbol{V}_K$. Following this suggestion [Harshman 1972], we impose a column-wise orthonormal matrix $\boldsymbol{P}_k \in \mathbb{R}^{M_k \times R'}$ (i.e., $\boldsymbol{P}_k^{\mathrm{T}} \boldsymbol{P}_k = I$) and a square matrix $V \in \mathbb{R}^{R \times R}$ that does not vary by slice. We then find that the cross-product constraint is enforced implicitly since

$$\boldsymbol{V}_K^T \boldsymbol{V}_K = (\boldsymbol{P}_k \boldsymbol{V})^{\mathrm{T}} (\boldsymbol{P}_k \boldsymbol{V}) = \boldsymbol{V}^{\mathrm{T}} \boldsymbol{V} = \boldsymbol{\Phi} \tag{14}$$

Accordingly, the loss function of Equation (13) can be rewritten as

$$J = \underset{\boldsymbol{U},\boldsymbol{V},\boldsymbol{C}}{argmin} \frac{1}{2} \sum_{k=1}^{K} ||\boldsymbol{W}_k \circledast \left(\boldsymbol{X}_k - \boldsymbol{U} \boldsymbol{\Sigma}_k (\boldsymbol{P}_k \boldsymbol{V})^{\mathrm{T}}\right) ||_F^2, \qquad s.t.\ \boldsymbol{P}_k^{\mathrm{T}} \boldsymbol{P}_k = I \tag{15}$$

Weighting Influence over Domains. From Equation (15), user factor matrix $\boldsymbol{U}$ is shared across all domains, i.e., learning $\boldsymbol{U}$ is affected by the loss of data in all domains. However, the amount of data in each domain is quite different. Some domains may have many items and extensive feedback while other domains may only have a few items and little feedback. As a result, $\boldsymbol{U}$ tends to be mainly determined by domains with a large amount of data. Hence, we can assign a weight $\omega_k > 0$ to the loss in each domain to control the penalty of this loss.

$$J = \underset{\boldsymbol{U},\boldsymbol{V},\boldsymbol{C}}{argmin} \frac{1}{2} \sum_{k=1}^{K} \omega_k \|\boldsymbol{W}_k \circledast (\boldsymbol{X}_k - \boldsymbol{U} \boldsymbol{\Sigma}_k (\boldsymbol{P}_k \boldsymbol{V})^{\mathrm{T}})\|_F^2, \qquad s.t.\ \boldsymbol{P}_k^{\mathrm{T}} \boldsymbol{P}_k = I \tag{16}$$

Intuitively, if we assign a large weight to the loss in a domain, then factor matrix $\boldsymbol{U}$ is largely learned from the factorization over this domain. Note that a change of $\boldsymbol{U}$ will update other factor matrices $\boldsymbol{V}$, $\boldsymbol{C}$, $\boldsymbol{P}_k$ in turn during the process of factorization. Therefore, we can control the learning result of all factor matrices by tuning the weight assigned to each slice. In fact, ω_k can be absorbed into $\boldsymbol{W}_k$, that is, we can rewrite the entry of the weight matrix as

$$\boldsymbol{W}_{k,i,j} = \sqrt{\omega_k w_{k,i,j}} \tag{17}$$

Thus, we still have the same form of loss function as given by Equation (15). In addition, we denote $\boldsymbol{W}_k = \boldsymbol{W}_k \circledast \boldsymbol{W}_k$ as the element-wise squared $\boldsymbol{W}_k$, which is involved in most of the following computations.

$$\ddot{\boldsymbol{W}}_{k,i,j} = \omega_k w_{k,i,j} \tag{18}$$

Equivalence to TF. The loss function given by Equation (15) is the summation of loss over the matrix $\boldsymbol{X}_k$ of each domain with different sizes, which still cannot capture the interaction across domains. Unfortunately, Equation (15) cannot be transformed into a TF problem in terms of PARAFAC2 [Kiers et al. 1999] to calculate the weighted loss over each domain, due to the Hadamard product. To enable it to capture cross-domain correlation, we give the following theorem to transform Equation (15) into an equivalent TF problem. The proof of this theorem is provided in the Appendix.

THEOREM 4.1. *Minimizing the weighted loss given by Equation (15) is equivalent to minimizing*

$$J = \underset{\boldsymbol{U},\boldsymbol{V},\boldsymbol{C}}{argmin} \frac{1}{2} \left[\|\boldsymbol{\mathcal{Y}} - [\![\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{C}]\!]\|^2 + \sum_k \|\hat{\boldsymbol{X}}_k \circledast \boldsymbol{H}_k\|_F^2 \right], s.t.\ \boldsymbol{P}_k^{\mathrm{T}} \boldsymbol{P}_k = \boldsymbol{I} \tag{19}$$

where $\mathcal{Y} \in \mathbb{R}^{N\times R\times K}$ *is a third-order tensor with K slices:* $\boldsymbol{Y}_k = \boldsymbol{Z}_k\boldsymbol{P}_k$, $\hat{\boldsymbol{X}}_k = \boldsymbol{U}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}}$ *and* $\boldsymbol{H}_k = \sqrt{\ddot{\boldsymbol{W}}_k - \boldsymbol{1}^{\mathrm{T}}\boldsymbol{1}}$*, i.e.* $\boldsymbol{H}_{k,i,j} = \sqrt{\omega_k w_{k,i,j} - 1}$*. Here, we denote* $\boldsymbol{Z}_k = \ddot{\boldsymbol{W}}_k\boldsymbol{X}_k$.

The first term of (19) is the unweighted loss on fitting $\mathcal{Y}$, and the second term is derived from the transformation from Equation (15) to Equation (19) by eliminating the weight matrices $\{\boldsymbol{W}_k\}$ from the weighted loss over $\{\boldsymbol{X}_k\}$ (see the derivation in the Appendix); it can be interpreted as weighted loss compensation to the unweighted loss on fitting $\mathcal{Y}$ to keep the equivalence between Equation (15) and Equation (19). Therefore, we can now use a TF approach to capture the high-order interaction between domains.

We obtain the final objective by appending the regularization terms to avoid overfitting.

$$J = \underset{\boldsymbol{U},\boldsymbol{V},\boldsymbol{C}}{argmin}\,\frac{1}{2}\left[\underbrace{\left(||\mathcal{Y} - [\boldsymbol{U},\boldsymbol{V},\boldsymbol{C}]||^2 + \lambda_U||\boldsymbol{U}||_F^2 + \lambda_V||\boldsymbol{V}||_F^2 + \lambda_C||\boldsymbol{C}||_F^2\right)}_{1:Regularized\ TF\ Model} + \underbrace{\sum\nolimits_k ||\hat{\boldsymbol{X}}_k \circledast \boldsymbol{H}_k||_F^2}_{2:Loss\ Compensation}\right]$$

$$s.t.\ \boldsymbol{P}_k^{\mathrm{T}}\boldsymbol{P}_k = \boldsymbol{I} \tag{20}$$

4.3. Parameter Learning

Given the above objective function, we designed a constrained optimization algorithm to learn the parameters $\boldsymbol{\Theta} = (\boldsymbol{U},\boldsymbol{V},\boldsymbol{C},\{\boldsymbol{P}_k\})$. This algorithm consists of two subprocedures: one is to learn $\{\boldsymbol{P}_k\}$ with the column-wise orthonormal constraint; the other is to learn a TF w.r.t. $\{\boldsymbol{U},\boldsymbol{V},\boldsymbol{C}\}$.

Finding Constrained $\{\boldsymbol{P}_k\}$. When $\{\boldsymbol{U},\boldsymbol{V},\boldsymbol{C}\}$ are fixed, learning $\boldsymbol{P}_k$ is conducted to solve the following constrained weighted least squares (WLS) problem on the data of each domain k (cf. Equation (15))

$$J = \underset{\boldsymbol{P}_k}{argmin}\,\|\boldsymbol{W}_k \circledast (\boldsymbol{X}_k - \boldsymbol{U}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}})\|_F^2, \qquad s.t.\ \boldsymbol{P}_k^{\mathrm{T}}\boldsymbol{P}_k = \boldsymbol{I} \tag{21}$$

This corresponds to a WLS-based *Orthogonal Procrustes Problem* [Gower and Dijksterhuis 2004]. Now, let $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}_k\boldsymbol{V}^{\mathrm{T}}$, and we can use an iterative approach [Groenen et al. 2005; Kiers 1997] to find $\boldsymbol{P}_k$ by minimizing

$$\left\|\left[\boldsymbol{M}\hat{\boldsymbol{P}}_k^T - \bar{\boldsymbol{W}}_k \circledast \left(\boldsymbol{M}\hat{\boldsymbol{P}}_k^T - \boldsymbol{X}_k\right)\right] - \boldsymbol{M}\boldsymbol{P}_k^{\mathrm{T}}\right\|_F^2$$

where $\bar{\boldsymbol{W}}_k = \ddot{\boldsymbol{W}}_k\boldsymbol{D}^{-1}$ and $\boldsymbol{D} = diag(\{max(\ddot{\boldsymbol{W}}_{k,i,:})\})$ is a diagonal matrix whose diagonal elements are the maximum elements of the rows of $\ddot{\boldsymbol{W}}_k$, and $\hat{\boldsymbol{P}}_k$ denotes the current estimate of $\boldsymbol{P}_k$. Now, let $\boldsymbol{G} = \boldsymbol{M}\hat{\boldsymbol{P}}_k^T - \bar{\boldsymbol{W}}_k \circledast (\boldsymbol{M}\hat{\boldsymbol{P}}_k^T - \boldsymbol{X}_k)$, and the above function leads to the orthogonal Procrustes problem of $\|\boldsymbol{G} - \boldsymbol{M}\boldsymbol{P}_k^{\mathrm{T}}\|_F^2$, which has a close-form solution [Cliff 1966; Groenen et al. 2005]:

$$\boldsymbol{G}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{M} \approx \boldsymbol{A}_R\boldsymbol{\Sigma}_R\boldsymbol{B}_R^{\mathrm{T}}$$

$$\boldsymbol{P}_k = \boldsymbol{A}_R\boldsymbol{B}_R^T \tag{22}$$

where $\boldsymbol{A}_R\boldsymbol{\Sigma}_R\boldsymbol{B}_R^{\mathrm{T}}$ is the truncated R-rank SVD on the matrix $\boldsymbol{G}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{M}$. The estimated $\boldsymbol{P}_k$ is then used as $\hat{\boldsymbol{P}}_k$ in the next iteration.

Finding $\{\boldsymbol{U},\boldsymbol{V},\boldsymbol{C}\}$. When the column-wise orthonormal matrices $\{\boldsymbol{P}_k\}$ are given, we need to resolve the TF-based optimization problem given in Equation (20). Because of the Hadamard product in the loss compensation part of Equation (20), we cannot obtain the gradient w.r.t. the matrix $\boldsymbol{U}$ as a whole, but we can obtain the gradient w.r.t. each row of $\boldsymbol{U}$, i.e., the factor vector of each user. Referring to the gradients in Equation (11) for TF, we have

$$\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{U}_{i,:}} = -\boldsymbol{Y}_{(1),i,:}(\boldsymbol{C} \circledast \boldsymbol{V}) + \boldsymbol{U}_{i,:}(\boldsymbol{C}^T\boldsymbol{C} \circledast \boldsymbol{V}^T\boldsymbol{V}) + \lambda_U \boldsymbol{U}_{i,:} + \boldsymbol{U}_{i,:}\sum_{k=1}^{K}\left[\boldsymbol{\Sigma}_k\left(\boldsymbol{P}_k\boldsymbol{V}\right)^T \boldsymbol{\Omega}_{k,i}\boldsymbol{P}_k\boldsymbol{V}\boldsymbol{\Sigma}_k\right]$$

where $\boldsymbol{\Omega}_{k,i} = diag(\ddot{\boldsymbol{W}}_{k,i,:}) - \boldsymbol{I}$ is a diagonal matrix. We set this partial derivative to $\mathbf{0}$, and the update equation w.r.t. $\boldsymbol{U}_{i,:}$ is obtained.

$$\boldsymbol{U}_{i,:} = \boldsymbol{Y}_{(1),i,:}\left(\boldsymbol{C} \odot \boldsymbol{V}\right)\left(\boldsymbol{C}^T C \circledast \boldsymbol{V}^T\boldsymbol{V} + \lambda_U \boldsymbol{I} + \sum_{k=1}^{K}\left[\boldsymbol{\Sigma}_k\left(\boldsymbol{P}_k\boldsymbol{V}\right)^T \boldsymbol{\Omega}_{k,i}\boldsymbol{P}_k\boldsymbol{V}\boldsymbol{\Sigma}_k\right]\right)^{-1} \tag{23}$$

Similarly, we can obtain the update equation w.r.t. each domain factor vector $\mathrm{C}_{\mathrm{k},:}$:

$$\boldsymbol{C}_{k,:} = \boldsymbol{Y}_{(3),k,:}\left(\boldsymbol{V} \odot \boldsymbol{U}\right)\left(\boldsymbol{U}^T\boldsymbol{U} \circledast \boldsymbol{V}^T\boldsymbol{V} + \lambda_C \boldsymbol{I} + \sum_{i=1}^{N}\left[\boldsymbol{\Sigma}_i\left(\boldsymbol{P}_k\boldsymbol{V}\right)^T \boldsymbol{\Omega}_{k,i}\boldsymbol{P}_k\boldsymbol{V}\boldsymbol{\Sigma}_i\right]\right)^{-1} \tag{24}$$

where $\boldsymbol{\Sigma}_i = diag(\boldsymbol{U}_{i,:})$. The partial derivative w.r.t. $\boldsymbol{V}$ is

$$\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{V}} = -\boldsymbol{Y}_{(2)}\left(\boldsymbol{C} \odot \boldsymbol{U}\right) + \boldsymbol{V}(\boldsymbol{C}^T\boldsymbol{C} \circledast \boldsymbol{U}^T\boldsymbol{U}) + \lambda_V \boldsymbol{V} + \sum_{k=1}^{K}\sum_{i=1}^{N}\boldsymbol{P}_k^T\boldsymbol{\Omega}_{k,i}\boldsymbol{P}_k\boldsymbol{V}\boldsymbol{\Sigma}_k\boldsymbol{U}_{i,:}^T\boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k$$

Due to $(\boldsymbol{P}_k^T\boldsymbol{\Omega}_{k,i}\boldsymbol{P}_k)\boldsymbol{V}(\boldsymbol{\Sigma}_k\boldsymbol{U}_{i,:}^T\boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k)$ having the form $\boldsymbol{AVB}$, we cannot simply obtain the update equation as above. According to the property of the Kronecker product, we place the vectorization operator on both sides of the above equation, and then we obtain:

$$\begin{aligned} vec\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{V}} &= -vec\left[\boldsymbol{Y}_{(2)}\left(\boldsymbol{C} \odot \boldsymbol{U}\right)\right] + vec\left[\boldsymbol{V}\left(\boldsymbol{C}^T\boldsymbol{C} \circledast \boldsymbol{U}^T\boldsymbol{U} + \lambda_V \boldsymbol{I}\right)\boldsymbol{I}\right] \\ &\quad + vec\left(\sum_{k=1}^{K}\sum_{i=1}^{N}\boldsymbol{P}_k^T\boldsymbol{\Omega}_{k,i}\boldsymbol{P}_k\boldsymbol{V}\boldsymbol{\Sigma}_k\boldsymbol{U}_{i,:}^T\boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k\right) = -vec\left[\boldsymbol{Y}_{(2)}\left(\boldsymbol{C} \odot \boldsymbol{U}\right)\right] \\ &\quad + \left[\left(\boldsymbol{C}^T\boldsymbol{C} \circledast \boldsymbol{U}^T\boldsymbol{U} + \lambda_V \boldsymbol{I}\right) \otimes \boldsymbol{I} + \sum_{k=1}^{K}\sum_{i=1}^{N}\left(\boldsymbol{\Sigma}_k\boldsymbol{U}_{i,:}^T\boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k\right) \otimes \left(\boldsymbol{P}_k^T\Omega_{k,i}\boldsymbol{P}_k\right)\right] vec\boldsymbol{V} \end{aligned}$$

where the vectorization of a matrix is a transformation which converts the matrix into a column vector. Accordingly, we can obtain the update equation w.r.t. $vec\boldsymbol{V}$:

$$\begin{aligned} vec\boldsymbol{V} &= \left[\left(\boldsymbol{C}^T\boldsymbol{C} \circledast \boldsymbol{U}^T\boldsymbol{U} + \lambda_V \boldsymbol{I}\right) \otimes \boldsymbol{I} + \sum_{k=1}^{K}\sum_{i=1}^{N}\left(\boldsymbol{\Sigma}_k\boldsymbol{U}_{i,:}^T\boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k\right) \otimes \left(\boldsymbol{P}_k^T\boldsymbol{\Omega}_{k,i}\boldsymbol{P}_k\right)\right]^{-1} \\ &\quad vec\left[\boldsymbol{Y}_{(2)}\left(\boldsymbol{C} \odot \boldsymbol{U}\right)\right] \end{aligned} \tag{25}$$

Algorithm Summarization. Algorithm 1 summarizes the parameter learning procedure. In particular, we find that $\boldsymbol{U}_{i,:}$, $\boldsymbol{C}_{k,:}$, and $\boldsymbol{P}_k$ are updated in a parallel scheme by taking advantage of the conditional independence. $\boldsymbol{V}$ is updated as a whole, but its size is very small, only $R \times R$, and is not dependent on the number of users and items. In Equations (23), (24), and (25), we need to compute their matrix inversions respectively. In fact, we can avoid computing these inversions by solving linear systems. Algorithm 1 consists of two parts of sub-iterations, and in practice, we find that the number of sub-iterations can be set with a small value, i.e., m and n are set as less than 5 in general.

ALGORITHM 1: Weighted Irregular Tensor Factorization

```
[$\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{C}, \{\boldsymbol{P}_k\}$] = WITF ($\{\boldsymbol{X}_k\}, \{\omega_k\}, \{w_{k,i,j}\}, \lambda_U, \lambda_V, \lambda_C$)
Input:  $\boldsymbol{X}_k$ is the data matrix for each domain
        $\omega_k$ is the influence weight for each domain
        $w_{k,i,j}$ is the weight on each entry
        $\lambda_U, \lambda_V, \lambda_C$ are the regularization parameters
Output: $\boldsymbol{U}$ is the factor matrix for users
        $\boldsymbol{C}$ is the factor matrix for domains
        $\boldsymbol{V}, \boldsymbol{P}_k$ are the factor matrices for items
Begin
Initialization:
 1: $\ddot{\boldsymbol{W}}_{k,i,j} \leftarrow \omega_k w_{k,i,j}, \boldsymbol{V} \leftarrow \boldsymbol{I}$
 2: Randomly initialize $\boldsymbol{U}, \boldsymbol{C}$
 3: $\boldsymbol{P}_k \leftarrow \boldsymbol{A}_R \boldsymbol{B}_R^{\mathrm{T}}$, with the SVD: $\boldsymbol{X}_k^{\mathrm{T}} \boldsymbol{U} \boldsymbol{\Sigma}_k \boldsymbol{V}^{\mathrm{T}} \approx \boldsymbol{A}_R \boldsymbol{\Sigma}_R \boldsymbol{B}_R^{\mathrm{T}}$
Iteration:
Add neighbor noisy examples (optional):
 4: Randomly select $S$ blank entries for each user $i$
 5: Fill neighbor noisy examples in the selected entries
 6: Generate tensor $\boldsymbol{\mathcal{Y}}$ with the slice for each domain $k$:
                    $\boldsymbol{Y}_k \leftarrow (\ddot{\boldsymbol{W}}_k \circledast \boldsymbol{X}_k)\boldsymbol{P}_k$
Sub-iteration for $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{C}$:
 7: Update $\boldsymbol{U}_{i,:}$ in parallel for each user $i$ using Equation (23)
 8: Update $\boldsymbol{C}_{k,:}$ in parallel for each domain $k$ using Equation (24)
 9: Update $\boldsymbol{V}$ using Equation (25)
10: Repeat 7-9 with m iterations
Sub-iteration for $\{\boldsymbol{P}_k\}$:
11: Update $\boldsymbol{P}_k$ in parallel for each domain $k$ using Equation (22)
12: Repeat 11 with n iterations
Repeat 4–12 until convergence
13: Return $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{C}, \{\boldsymbol{P}_k\}$
End
```

In our experiments, we set $m = n = 1$ which produces a sufficiently good result. This is because too many iterations easily become stuck in poor local minima. According to the analysis, this algorithm can be executed very efficiently in a powerful parallel computing environment. Note that Steps (4) and (5) in Algorithm 1 constitute an optional sub-procedure for adding noisy data from neighbors to relieve overfitting and improve generalization when data is very sparse. The detail of this sub-procedure will be discussed in Section 4.5.

4.4. Weight Matrix Configuration

The weight matrices play an important role in the WITF model. From Equation (18), we can find the weight matrix in each domain which consists of the weight on each entry of data matrix, $w_{k,i,j}$, and the weight of domain influence, ω_k.

4.4.1. Weights Over Data. The data in recommender systems can usually be divided into two categories: n-ary preference, such as ratings, and unary preference, such as clicks or purchases.

N-ary Preference Data. The multilevel ratings, e.g., five-star rating data, which explicitly differentiate user preferences, are typical of n-ary feedback ($n \geq 2$). This kind of feedback is usually treated as a missing data problem. That is, we only model observed ratings and the remaining entries are treated as missing. An indicative matrix of binary weights, which has been successfully applied in MF and TF methods [Acar et al. 2010; Srebro and Jaakkola 2003], is often used in this type of case. Based on this setting, we set the weights for n-ary preference data as follows:

$$w_{k,i,j} = \begin{cases} 1 & (k,i,j)\, \textit{is an observation} \\ a & (k,i,j)\, \textit{is a noisy example} \\ 0 & \textit{else} \end{cases} \tag{26}$$

Here, $a \leq 1$ is a smaller constant weight assigned to the additional noisy examples, which serve as virtual data. In practice, we can simply let $a = 1$ and tune the number of imposed noisy examples. This is discussed in more detail in the following subsection. As a result, the entries of weight matrices for n-ary preference data are given by

$$\ddot{\mathbf{W}}_{k,i,j} = \begin{cases} \omega_k & (k,i,j)\, \textit{is an observation} \\ a\omega_k & (k,i,j)\, \textit{is a noisy example} \\ 0 & \textit{else} \end{cases} \tag{27}$$

Under this setting, we find that the zero weights eliminate the loss on fitting missing entries. Therefore, we can use a set of sparse matrices to store only observed entries for each domain.

Unary Preference Data. In real-world applications, explicit preference data is not always available but implicit feedback, e.g., clicks and purchases, is more easily obtained. For example, users rarely rate their bought items, but their purchase records are available in the system. Observed entries are usually represented by 1 and blanks by 0 for this kind of data. This implicit feedback is called unary preference rather than binary preference because the blanks are usually generated as a result of users' lack of awareness and do not necessarily indicate user dislike [Herlocker et al. 2004]. Unary preference data are also known as one-class data in some literature [Pan et al. 2008].

As a result, we assign a higher confidence level to entries with observed choices and a lower confidence level to entries without observed choices [Hu et al. 2014]. Recall that the confidence level is controlled by the precision parameter in a Gaussian distribution. In our model, the precision parameter of the distribution of each entry corresponds to the weight. Therefore, we can differentiate between the weights on observed choice entries and those on unobserved choice entries. A similar weighting strategy has been successfully applied in some MF models for unary feedback [Hu et al. 2008; Pan et al. 2008]. In the WITF model, we set the weights for unary feedback as follows:

$$w_{k,i,j} = \begin{cases} c_{k,i,j} + 1 & (k,i,j)\, \textit{is observed} \\ 1 & \textit{else} \end{cases} \tag{28}$$

where $c_{k,i,j} \geq 0$ denotes the confidence parameter which can be set a single value [Hu et al. 2008] for each domain or can be set different values for each user and item [Hu et al. 2014; Pan et al. 2008]. We find that the weights on unobserved choices are always 1 (a minimal confidence) whereas the weights on observed choices are $c_{k,i,j} + 1 \geq 1$. Hence, this guarantees that the confidence level on observed choices, i.e., true positive instances, is higher than it is on unobserved ones, i.e., uncertainly negative instances. Accordingly, the entries of the final weight matrices for unary preference data are given by

$$\ddot{\mathbf{W}}_{k,i,j} = \begin{cases} \omega_k c_{k,i,j} + \omega_k & (k,i,j)\, \textit{is observed} \\ \omega_k & \textit{else} \end{cases} \tag{29}$$

4.4.2. Weights Over Domains. As previously noted, ω_k is used to trade off the influence between domains. The optimal weight configuration can be found by some advanced search methods, e.g., genetic algorithm [Hu et al. 2013]. However, the number of possible configuration increases exponentially with the number of domains so this approach is too time-consuming in practice. Here, we present an empirical strategy to seek a suboptimal weight configuration for WITF. The number of training examples on each domain may be quite different, so learning WITF can be regarded as a problem to fit imbalance data over multiple domains. Cost-sensitive learning [Zhou and Liu 2010] is an often used approach to dealing with such imbalance data, where a typical process is to assign different weights to training examples of different classes (here corresponds to domains) in proportion to the misclassification costs (here corresponds to the loss of fitting all training examples of a domain).

Without loss of generality, we fix ω_T for the target domain T as 1 because it does not change the optimization problem by scaling all $\{\omega_k\}$. Now, let $\#O_T$ denote the number of training examples on the target domain and $\#O_k$ denote the number of training examples on an auxiliary domain k; we then empirically set ω_k as follows:

$$\omega_k = \alpha_k \frac{\#O_T}{\#O_k} \tag{30}$$

where $\alpha_k \geq 0$ is a proportional parameter which controls the amount of influence from auxiliary domains. Since loss $\mathcal{L}_k$ of fitting training examples is proportional to $\#O_k$, i.e., $\mathcal{L}_k \propto \omega_k \#O_k$, we have the contribution ratio between the loss of an auxiliary domain $\mathcal{L}_k$ and the that of the target domain $\mathcal{L}_T$ by using Equation (30):

$$\frac{\mathcal{L}_k}{\mathcal{L}_T} \propto \frac{\omega_k \#O_k}{1\#O_T} = \alpha_k \tag{31}$$

That is, the amount of contribution from the auxiliary domain is controlled by α_k in terms of scaling the cost of loss on fitting the data on this auxiliary domain. A small α_k tends to bypass the influence from the auxiliary domain so the user preference are mainly learned from the target domain, whereas a large α_k borrows large amount of information from the auxiliary domain. That is, a larger α_k implies more compatible user preferences between the target domain and the auxiliary domain, therefore we can assess the heterogeneities between different auxiliary domains and target domain according to $\{\alpha_k\}$. In general, an optimal α_k is often selected from a set of given values in terms of cross-validation.

4.5. Remarks

4.5.1. Tricks on Sparse Weight Matrices for Complexity Reduction. The weight matrices are only involved in the derivatives of the loss compensation term when computing the parameter updating equations: Equations (23∼25). In particular, we find that the term related to these weights has the form $\boldsymbol{A}_k^{\mathrm{T}}\boldsymbol{\Omega}_{k,i}\boldsymbol{A}_k$, and specifically, $\boldsymbol{A}_k = \boldsymbol{P}_k\boldsymbol{V}\boldsymbol{\Sigma}_k$ in Equation (23), $\boldsymbol{A}_k = \boldsymbol{P}_k\boldsymbol{V}$ in Equation (24) and $\boldsymbol{A}_k = \boldsymbol{P}_k$ in Equation (25). Clearly, $\boldsymbol{A}_k$ does not contain the subscript i, so it can be pre-computed before looping the user index i. As a result, we can design a more efficient computing strategy in consideration of the weights.

Let us now take Equation (23) as the example. We can expand $\boldsymbol{A}_k^{\mathrm{T}}\boldsymbol{\Omega}_{k,i}\boldsymbol{A}_k$ as $\boldsymbol{A}_k^{\mathrm{T}} diag(\ddot{\boldsymbol{W}}_{k,i,:})\boldsymbol{A}_k - \boldsymbol{A}_k^{\mathrm{T}}\boldsymbol{A}_k$ for explicit preference data. Since $diag(\ddot{\boldsymbol{W}}_{k,i,:})$ only contains $m_{k,i}$ non-zero entries (cf. Equation (27)) for user i on domain k, $\boldsymbol{A}_k^{\mathrm{T}} diag(\ddot{\boldsymbol{W}}_{k,i,:})\boldsymbol{A}_k$ can be computed in time $\mathrm{O}(\sum_k R^2 m_{k,i})$. Note that both R and $m_{k,i}$ are small, so the time complexity is very low. The total time complexity for looping all users is $\mathrm{O}(N\sum_k R^2\bar{m}_k)$ where $\bar{m}_k$ denotes the mean of the number of observations over users on domain k. The computation on the term $\sum_{k=1}^{K}[\boldsymbol{A}_k^{\mathrm{T}}\boldsymbol{A}_k]$ is in time $\mathrm{O}(\sum_k R^2 M_k)$, which can be precomputed before looping. As a whole, the total time complexity on looping all users is

Table II. Statistics of Epinions Dataset over Five Heterogeneous Domains

Domain	# Items	# Ratings / # Users	# Ratings / # Items	Sparsity
*Kids & Family**	3,769	4.9309	9.9077	0.0013
*Hotels & Travel**	2,545	3.9210	11.6676	0.0015
Restaurants & Gourmet	2,543	3.3394	9.9446	0.0013
Wellness & Beauty	3,852	3.5481	6.9756	0.0009
Home and Garden	2,785	2.6003	7.0707	0.0009

*Each of these domains is chosen as the target domain for evaluation respectively.

$\mathrm{O}(N\sum_k R^2\bar{m}_k + \sum_k R^2 M_k)$. In fact, it can be finished in time $\mathrm{O}(\sum_k R^2\bar{m}_k + \sum_k R^2 M_k)$, when we loop users in a parallel scheme, as given in Algorithm 1.

The diagonal weight matrix $\Omega_{k,i} = diag(\ddot{\mathbf{W}}_{k,i,:}) - \boldsymbol{I}$ in the case of unary preferences can be equivalently rewritten as $[diag(\ddot{\mathbf{W}}_{k,i,:}) - \omega_k \boldsymbol{I}] - (1-\omega_k)\boldsymbol{I}$. From Equation (29), we find that the term $[diag(\ddot{\mathbf{W}}_{k,i,:}) - \omega_k \boldsymbol{I}]$ only contains $m_{k,i}$ non-zero diagonal entries with the value $\omega_k c$. Therefore, $\sum_{k=1}^{K}[\boldsymbol{A}_k^{\mathrm{T}}[diag(\ddot{\mathbf{W}}_{k,i,:}) - \omega_k \boldsymbol{I}]\boldsymbol{A}_k]$ can be computed in time $\mathrm{O}(N\sum_k R^2\bar{m}_k)$. That is, it can be finished in time $\mathrm{O}(\sum_k R^2\bar{m}_k)$ in a parallel scheme. The term $(1-\omega_k)\sum_{k=1}^{K}[\boldsymbol{A}_k^{\mathrm{T}}\boldsymbol{A}_k]$ can be pre-computed in time $\mathrm{O}(\sum_k R^2 M_k)$. Therefore, the total time complexity is the same as the explicit feedback case, i.e., $\mathrm{O}(N\sum_k R^2\bar{m}_k + \sum_k R^2 M_k)$ and $\mathrm{O}(\sum_k R^2\bar{m}_k + \sum_k R^2 M_k)$ in a parallel fashion. Similar tricks can be applied to efficiently compute Equations (24) and (25) to loop users.

4.5.2. Training with Additional Noisy Examples for Improving Generalization. As mentioned previously, users' feedback follows power-law distribution in most domains; that is to say, all users in the long tail are cold-start and have very little data. We only model the observed ratings for explicit feedback, which tends to lead to very poor generalization performance as a result of overfitting with too few data. Table II depicts statistics from a real-world explicit feedback dataset in the experiment section, from which we find that the mean number of ratings over users is less than 5 in each domain, i.e., the total number of observations is less than $5N$. The data are even sparser in the real-world scenario since we removed some users and items with too few data from the experiments. However, the number of factors, R, is normally larger than 5. Therefore, the total number of parameters for a domain (all latent user factors, item factors and domain factors) is $NR + M_k R + R \gg 5N$, even if we set $R = 5$. The number of parameters is much greater than the number of observations, which leads to overfitting issues and results in very poor prediction performance.

In the case of fewer observations and more parameters, we have tried to tune the regularization parameters $\lambda_U, \lambda_V, \lambda_C$ (cf. Equation (20)) but have failed to improve performance significantly, which may be attributable to data being too sparse and having a rank problem of a matrix, namely many rows are all zeros. As a result, we found another effective way to tackle this issue, and to be free from tuning $\lambda_U, \lambda_V, \lambda_C$. It has been illustrated in the literature [Bishop 1995] that training by adding noise is equivalent to regularization. However, simply adding noise to observations does not change the sparsity of data. In our framework, we train WITF to fill noisy examples into randomly selected blank entries instead of merely smoothing objective functions (regularization) by adding noise onto observed data. This strategy has been effectively applied to improve generalization [Niyogi et al. 1998; Vincent et al. 2008]. In detail, we fill the blanks with Gaussian noisy virtual examples as specified in Algorithm 1. First, we randomly select S blank entries for each user. Then, a noisy virtual example is generated as $\tilde{X}_{kij} = \mu_k + e_{kij}$ where $e_{kij} \sim N(0, \sigma^2)$ for each selected blank entry, where $\boldsymbol{\mu}_k$ denotes the mean of observed data on domain k. Accordingly, we assign small weights to these randomly generated examples (cf. Equation (26)) while larger weights to real observations to differentiate between their confidence levels. The setting of S is dependent on the amount of data. Generally, sparser data requires larger S, and we will compare performance using different settings of S over the data with different sparsities in the experiment section.

5. POST-LEARNING

When the parameters $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{C}, \{\boldsymbol{P}_k\}$ are learned from WITF, the user factor matrix is represented as $\boldsymbol{U}_k = \boldsymbol{\Sigma}_k \boldsymbol{U}^{\mathrm{T}}$ and the item factor matrix is $\boldsymbol{V}_k = (\boldsymbol{P}_k \boldsymbol{V})^{\mathrm{T}}$ in a given target domain k. We can then immediately use $\boldsymbol{U}_k, \boldsymbol{V}_k$ for prediction, as follows:

$$\hat{\boldsymbol{X}}_k = \boldsymbol{U}_k^{\mathrm{T}} \boldsymbol{V}_k = \boldsymbol{U} \boldsymbol{\Sigma}_k (\boldsymbol{P}_k V)^{\mathrm{T}} \tag{32}$$

where $\hat{\boldsymbol{X}}_k$ is the predictive data matrix of domain k. The recommendation list can then be ranked according to the predictive values.

As presented in Section 3.3, $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{C}, \{\boldsymbol{P}_k\}$ are estimated from the data over all domains, so $\boldsymbol{U}_k$ may not perfectly represent the user preference feature in the target domain and $\boldsymbol{V}_k$ also may not perfectly represent the feature of items. Therefore, we use the data in the target domain to finely tune $\boldsymbol{U}_k$ and $\boldsymbol{V}_k$. The factors $\boldsymbol{U}_k$ and $\boldsymbol{V}_k$ learned from WITF serve as a good informative prior means for the WRMF (cf. Equation (1)), so we have

$$\boldsymbol{\mu}_i = \boldsymbol{U}_{k,:,i} \qquad \boldsymbol{\mu}_j = \boldsymbol{V}_{k,:,j}$$

Then, we use the MAP estimator to alternately update user factor vector $\boldsymbol{u}_i$ by Equation (8) and item factor vector $\boldsymbol{v}_i$ by Equation (9) until convergence. Thus, we obtain the refined user factor matrix $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N]$ and item factor matrix $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{M_k}]$.

From Equation (8) and Equation (9), we find that the regularization parameters τ_U and τ_V control the strength of shrinkage of $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ towards the prior means $\boldsymbol{\mu}_i$, i.e., $\boldsymbol{U}_{k,:,i}$ and $\boldsymbol{\mu}_j$, i.e., $\boldsymbol{V}_{k,:,j}$. In particular, it obtains $\boldsymbol{u}_i = \boldsymbol{\mu}_i$ and $\boldsymbol{v}_j = \boldsymbol{\mu}_j$ when we set large $\boldsymbol{\tau}_U$ and $\boldsymbol{\tau}_V$ to place huge regularization on $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$. Therefore, the prediction performance on finely tuned parameters will never be worse than the performance achieved directly using the parameters learned from WITF.

6. EXPERIMENTS

In this section, we evaluate our models and other state-of-the-art approaches with a set of metrics to compare prediction performance. The experiments are conducted on three real world datasets covering ratings, user clicks history, and time-period separated data.

6.1. Comparison Methods

In the following experiments, a group of state-of-the-art methods are employed for comparison, where some of these methods are used for explicit feedback and some others are used for implicit feedback.

—*Most-Pop.* This applies the simplest strategy to rank items by their popularity (measured by the number of observations associated with items).

—*kNN-CDCF:* The naïve user-based neighborhood CDCF model uses the integrated item set. In this experiment, we use the top 10 nearest neighbors (i.e., $k = 10$) with the highest similarity.

—*PMF.* This is a classic probabilistic MF model [Salakhutdinov and Mnih 2008]. It is used as a representative single-domain CF method. In the experiments, we train it only using the target domain data.

—*PMF-CDCF.* We use PMF as a naïve CDMF model which takes the concatenated rating matrix over all domains as the training data.

—*MF-IF.* This can be regarded as a zero-mean based WRMF model to deal with implicit feedback [Hu et al. 2008; Pan et al. 2008]. We use it as a single-domain MF model on implicit feedback data of target domain.

—*MF-CDCF-IF.* We use MF-IF as an implicit feedback-based CDMF model which takes the concatenated implicit data matrix over all domains as the training data.

—*CMF.* Collective matrix factorization [Singh and Gordon 2008] is a CDMF model. In the experiments, we couple the rating matrix of each domain on *user* dimension.

—*PARAFAC2*. It is a basic TF model [Harshman 1972] to deal with inconsistently sized matrices. It does not have a mechanism to control the influence from each domain when applied to CDCF problem. Furthermore, it cannot deal with implicit feedback under unary preference assumption. Therefore, we run it under binary assumption when conducting experiments on implicit feedback.

—*CDTF.* This a model developed in our previous work [Hu et al. 2013], which tunes the weight to control the influence between auxiliary domains and target domain by a genetic algorithm.

—*CDTF-IF.* This is a CDTF-based model that works with implicit feedback [Hu et al. 2013].

—*WITF.* This is an irregular TF model proposed in this article, which is able to deal with explicit and implicit feedback in a unified framework. In the experiments, we directly use the parameters estimated from WITF to conduct prediction.

—*WITF+WRMF.* The parameters learned from WITF serve as the priors for WRMF and are finely tuned by MAP on the target domain (cf. Section 5). Then, we use the refined parameters for prediction.

As discussed previously, we set the number of latent factors as 5, i.e., $R = 5$, which produces the best results in the following experiments, since a larger R easily leads to the overfitting on sparse datasets for all above latent factor models.

6.2. Evaluation Metric

We conducted experiments with all the comparison methods on rating data (the case of n-ary preference) and click data (the case of unary preference). We used rating metrics to assess performance on the rating data and ranking metrics to assess performance on the click data.

6.2.1. Rating Metrics. To measure the accuracy of rating prediction, we utilized the most widely used evaluation metrics, namely Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) [Herlocker et al. 2004].

$$MAE = \frac{\sum_{i=1}^{N} abs(Y_i - \hat{Y}_i)}{N} \qquad RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}{N}}$$

where Y_i denotes a true rating in testing set, $\hat{Y}_i$ is the predicted rating, and N denotes the number of ratings in the testing set.

6.2.2. Ranking Metrics. The most common way to assess the prediction performance on unary preference data is to measure whether relevant items are placed in the top positions of a recommendation list. Information retrieval metrics are therefore often employed to evaluate the ranking performance of a recommender system. Here, we denote $rel\,(k) = 1$ if the item at position k is relevant and $rel\,(k) = 0$ otherwise.

—*recall@K*. Considers the fraction of relevant items over all N relevant items:

$$recall@K = \frac{\sum_{k=1}^{K} rel\,(k)}{N}$$

—*precision@K*. Considers the fraction of relevant items over top K recommended items:

$$precision@K = \frac{\sum_{k=1}^{K} rel\,(k)}{K}$$

—*AP@K*. Average Precision is the average result over *precision*@1 $\sim$ K, which is defined as:

$$AP@K = \frac{\sum_{k=1}^{K} rel\,(k) \times precision@k}{min\,(K, N)}$$

—*nDCG@K*. Normalized Discounted Cumulative Gain [Burges et al. 2005] is a measure of ranking quality which places strong emphasis on relevant items:

$$nDCG@K = \frac{DCG@K}{IDCG@K}$$

where IDCG means ideal DCG and

$$DCG@K = \sum_{k=1}^{K} \frac{2^{rel(k)} - 1}{\log_2(k+1)}, \qquad IDCG@K = \sum_{k=1}^{K} \frac{1}{\log_2(k+1)}$$

6.3. Rating Prediction on Epinions.com

Epinions.com, set up in 1999 and ultimately acquired by eBay in 2005, was a general product review platform where ratings, buying tips, and advice were generated by consumers to help users decide on purchases. The products on Epinions.com cover a number of domains, such as electronics, movies, health, etc. The crawled dataset [Tang et al. 2012] contains 5-level ratings over products in 28 domains. In this experiment, we chose five domains to construct the cross-domain dataset: *Kids & Family*, *Hotels & Travel*, *Restaurants & Gourmet*, *Wellness & Beauty,* and *Home and Garden*. User preferences are clearly relevant but quite heterogeneous between these domains, so this dataset is very suitable for testing which CDCF methods are able to appropriately leverage information between the target domain and heterogeneous auxiliary domains.

6.3.1. Data Preparation. In this experiment, we extracted those users who had rated at least two items in one of the five domains and the items which had been rated by two users from the raw dataset. As a result, we obtained 7,573 users and 15,494 items. Some evaluation statistics from this dataset are reported in Table II, from which we can see that the mean number of ratings over users is less than 5 in each domain. That is, most users have unfamiliar domains and largely suffer from the cold-start issue.

In this experiment, we respectively chose *Kids & Family* and *Hotels & Travel* as the target domains for evaluating the rating prediction performance for all comparison methods. Figure 5 demonstrates the rating distribution over items on each of the target domains, where we can observe obvious long tails. Therefore, it is a very suitable scenario to borrow information from other domains to improve the inference on user preferences on the target domain. Given a target domain, we constructed the training/testing sets as follows. First, we randomly withheld 20% of the rating data from the target domain as the ground truth for testing, denoted as **TS-20%**, and the remaining 80% of data, denoted as **TR-80%** were used as the training set. In the same way, we constructed a sparser training set by withholding 50% of data, denoted as **TS-50%**, for testing, and the remaining 50% of data, denoted as **TR-50%**, were used as the training set.

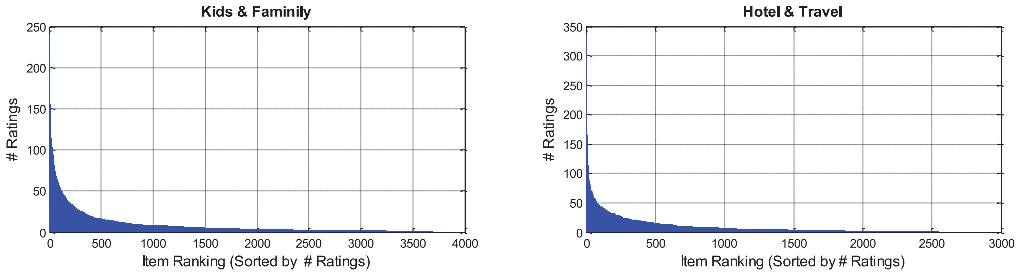


Fig. 5. Rating distributions of items on *Kids & Family* and *Hotel & Travel* show obvious long tails.

Table III. RMSEs of Comparison CDCF Methods on Epinions Dataset

Target Domain Method	*Kids & Family*		*Hotels & Travel*	
	TR-80%	TR-50%	TR-80%	TR-50%
kNN-CDCF	1.2562	1.3016	1.1605	1.3338
PMF-CDCF	1.1719$^{\wedge}$	1.3547$^{\wedge}$	1.1260$^{\wedge}$	1.2925$^{\wedge}$
CMF	1.1312*	1.2908*	1.0805*	1.2457*
PARAFAC2	1.1102*	1.1458*	1.0647*	1.0891*
CDTF	1.0968*	1.1219*	1.0351*	1.0585*
WITF	1.1043*	1.1293*	1.0375*	1.0619*
WITF+WRMF	1.0563**	1.0835**	0.9983**	1.0284**

$^{\wedge}$baseline, $^{*}p < 0.01$, **smallest p.

6.3.2. Rating Prediction Performance Comparison. Table III reports the RMSEs on the target domain *Kids & Family* and *Hotels & Travel,* respectively, over all testing users. Clearly, kNN-CDCF underperforms all comparison methods for all cases. This is because a user-based method such as this often fails to find existing ratings on tail items from neighbors to generate effective prediction. As for model-based methods, we can see that the TF-based models, i.e., PRAFAC2, CDTF, WITF, WITF+WRMF overall achieve better performance than MF-based models. This proves that the TF methods benefit from the modeling of three-way interaction between users, items and domains, which captures higher order information that is unable to learn from MF methods. In particular, we find the performance of all comparison methods is relatively close when the data are relatively dense (**TR-80%** cases), whereas the margins between TF-based methods and other methods become much wider when the data become sparser (**TR-50%** cases). Comparing CMF with PMF-CDCF, we find that CMF overall outperforms PMF-CDCF. The reason is that PMF-CDCF learns user preference on an integrated item set which ignores the heterogeneity between domains, especially when the amount of data in auxiliary domains is much larger than it is in the target domain. In comparison, CMF retains the rating matrix of each domain and provides a more effective way of controlling the knowledge transfer between domains.

Comparing all TF-based methods, we find that PARAFAC2 underperforms others since it is unable to trade off the amount of influence between the auxiliary domains and the target domain. WITF lags slightly behind CDTF because WITF uses an empirical suboptimal weights configuration in domains (cf. Section 4.4.2) while CDTF uses a search procedure to find an optimal weights configuration by genetic algorithm (GA) [Hu et al. 2013]. Note that WITF can also employ GA to search the optimal configuration of weights. However, running a GA-based search procedure is very time- and space-consuming because it is necessary to rerun the whole learning algorithm under each possible configuration generated by GA. In comparison, WITF uses a much more

Table IV. Statistics of Testing Users Grouped by the Number of Ratings

Target Domain Group	# Ratings	*Kids & Family*	*Hotels & Travel*
		# testing users in TS-50%	# testing users in TS-50%
Experienced	>20	120	55
Little Experienced	6∼20	816	517
Cold-Start	1∼5	2,260	2,807
Fully Cold-Start	0	695	1,072

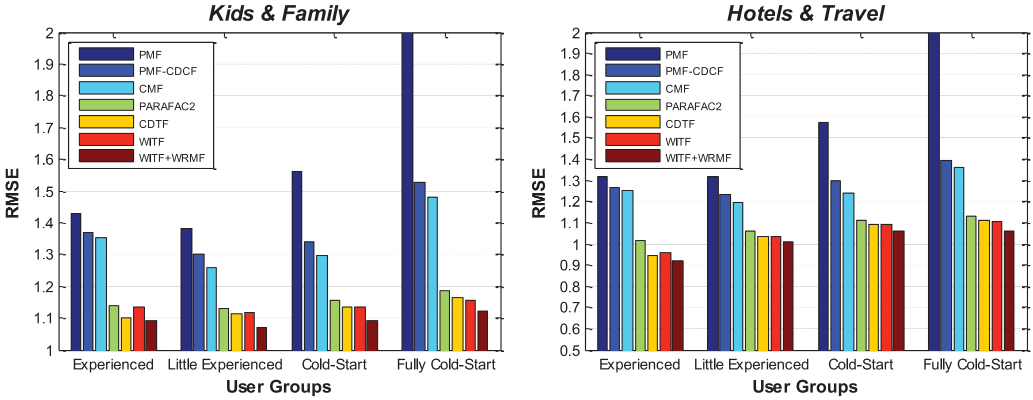


Fig. 6. RMSEs comparison over user groups with different number of ratings.

economical strategy to find a suboptimal weights configuration, and the parameters learned from WITF are also learned by MAP using WRMF. As a result, we obtained smaller RMSEs when the parameters learned from CDTF were finely tuned by WRMF. This proves the effectiveness of using the user, item, and domain factors learned from WITF as the informative priors for WRMF. Overall, WITF+WRMF is a more efficient and practical approach than all comparison methods.

Moreover, we performed a significant test by choosing PMF-CDCF as the baseline to compare with another model through their paired prediction errors. In particular, we used the sign test [Gunawardana and Shani 2015] to measure if a comparison method can significantly outperform the baseline with the p-value < 0.01 (marked with $*$ in Table III). In paritcular, the smallest p-value is returned from the test on comparing the outputs between WITF-WRMF and baseline, i.e., WITF+WRMF achieves the most significance improvement among all comparison methods.

6.3.3. The Prediction Performance Over Different Numbers of Training Ratings. To evaluate the performance with different amounts of training data, we split testing users into four groups according to the number of ratings they have in the training set **TR-50%**. Table IV shows statistics of these four groups in the target domain *Kids & Family* and *Hotels & Travel,* respectively. From the statistics, we find that most users are cold-start, which follows the typical long-tail distribution in the real world.

Figure 6 depicts the comparable RMSE results over the four user groups in the target domain *Kids & Family* and *Hotels & Travel,* respectively. We find that the performance on non-cold-start user groups (Experienced and Little Experienced) between PMF and PMF-CDCF is close. This is because it is not necessary to borrow much information from other domains when user feedback in the target domain is relatively adequate, so the single-domain PMF achieves comparable performance with PMF-CDCF in this case. In comparison, the performance on cold-start user groups of single-domain PMF is very poor because insufficient data is available to learn user preferences. As

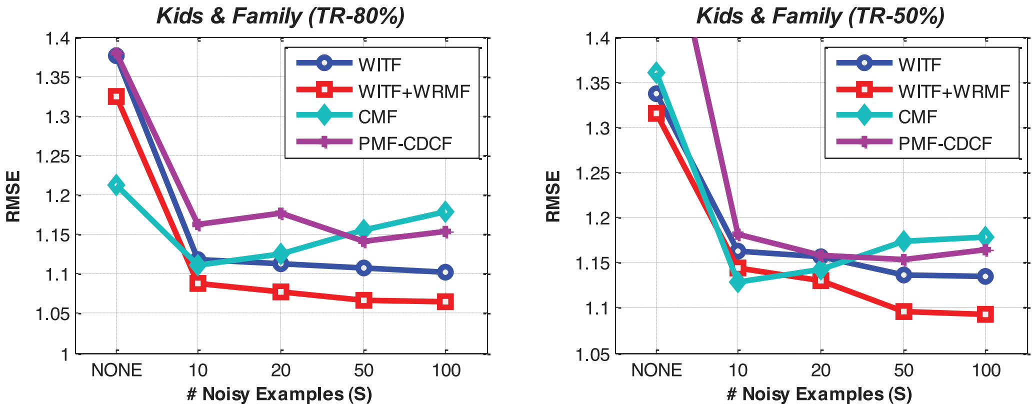


Fig. 7. RMSEs comparison over different number of noisy examples for each user.

analyzed in Section 3.1, PMF cannot deal with fully cold-start users who do not have any data, so we only show the truncated RMSE for PMF for the case of Fully Cold-Start. Obviously, CMF and PMF-CDCF lag well behind TF-based methods, especially in the case of Fully Cold-Start. This is because CMF and PMF-CDCF only model dyadic interaction between users and items so they cannot model domain factors to represent heterogeneities among domains. As a result, CMF and PMF-CDCF inevitably suffer from the blind transfer issue when no data are available in the target domain, i.e., fully cold-start. In comparison, TF-based methods are capable of representing the heterogeneities by domain factors and therefore achieve more stable performance over all four user groups. In all TF-based methods, PRAFAC2 underperforms all others due to the lack of a mechanism to control influence between domains. WITF+WRMF achieves the best performance in all cases. In particular, the fine-tuning procedure on WITF+WRMF leads to apparent margins from WITF.

6.3.4. Adding Gaussian Noisy Examples for Regularization. As discussed in Section 4.5.2, adding Gaussian noisy examples acts as a regularizer to avoid overfitting and improve generalization ability. The number of noisy examples added for each user is controlled by the parameter S. Intuitively, too small S generates insufficient data to survive from overfitting while too large S not only increases the time and space complexity but may overwhelm observed data. We therefore evaluated the prediction performance over different settings of S in this experiment. Apart from our WITF model, we also impose noisy examples to the baseline methods, PMF-CDCF and CMF, as the regularization method to test if it can imporve their generalization ability.

Figure 7 illustrates the rating prediction performance in the target domain *Kids & Family* using the training sets **TR-80%** and **TR-50%,** respectively. From Table II, we easily find that the mean number of ratings over users is less than 4 with **TR-80%** and less than 2.5 with **TR-50%** in the target domain, and even fewer in auxiliary domains. As analyzed in Section 4.5.2, the models become overfitting even if we use a small number of latent factors. Here, we demonstrate the change in RMSEs when $\{0, 10, 20, 50, 100\}$ noisy examples are added. When no noisy examples ($S = 0$) are imposed, the performance of all comparison methods is evidently poor due to lack of regularization. When a small number of noisy examples ($S = 10$) are added, the improvement is significant because overfitting is relieved (in this case, the total number of real and virtual observations becomes larger than the number of parameters).

As expected, too much noisy examples may over-regularize the parameter learning. CMF achieves the best performance when 10 noisy examples are imposed, whereas

Table V. Statistics of Tmall.com Click-Log Dataset over Four Anonymous Domains

Domain	# Items	# Clicks / # Users	# Clicks / # Items	Sparsity
*D1**	8,179	23.2003	19.7170	0.0028
*D2**	6,940	18.5455	18.5749	0.0027
D3	5,561	22.5005	28.1246	0.0040
D4	6,145	16.0606	18.1671	0.0026

*Each of these domains is chosen as the respective target domain for evaluation.

the performance drops when more noisy examples are added. Similar situation is observed w.r.t. PMF-CDCF. In comparison, WITF needs more noisy examples to overcome overfitting since it has more model parameters than those of CMF and PDF-CMF. For WITF and WITF-WRMF, the improvement in the case of **TR-80%** is relatively small when $S \geq 20$. In comparison, the improvement in **TR-50%** is more apparent than that it is in **TR-80%** when $S \geq 20$, since **TR-50%** is sparser than **TR-80%** and needs more noisy data for regularization. In conclusion, we only need to add a small number of noisy examples when training, which is effective to improve the generalization ability.

6.4. Click Prediction on Tmall.com

Merchants usually need to personalize the layout of items to attract potential buyers. It is well known that in the field of online advertising, customer targeting is extremely challenging, especially for new buyers. By targeting these potential customers, merchants can greatly reduce the cost of promotion and enhance the return on investment. Unfortunately, the explicit feedback from buyers is usually sparse and may not even be available for buyers in new domains. To alleviate this problem, it is valuable for merchants to utilize implicit feedback, e.g., the user click log recorded by the backend system. In this experiment, we use our models to solve this problem with the click log accumulated by Tmall.com[1] which is the largest B2C platforms in China.

6.4.1. Data Preparation. The dataset provided by Tmall.com contains anonymized user click logs for six months[2] covering 1,627 fine-grained anonymous domains, including clothes, furniture, books, food, and so on. In this raw dataset, all domains are anonymized, so we selected the four domains with the largest number of click records for evaluation, and denoted them as *D1*, *D2*, *D3,* and *D4*. First, we extracted 7,000 users who had at least 15 clicks in any two domains so that at least one focused domain for each user was available to serve as the auxiliary domain. Then, we trimmed the items accounting for fewer than five clicks. The statistics of this preprocessed dataset for evaluation are illustrated in Table V. In this experiment, *D1* and *D2* are, respectively, chosen as the target domains for evaluation. For each target domain, we applied the same strategy as the previous experiment to construct two training sets, **TR-80%** and **TR-50%**, and used the withheld data as the testing sets **TS-20%** and **TS-50%**.

6.4.2. Ranking Prediction Performance Comparison. In this experiment, we evaluated the prediction performance on clicks using the metrics AP@5, AP@20, nDCG@5, and nDCG@20. Table VI reports the mean results with the sign test over all testing users on the target domain *D1* and *D2,* respectively. Predicting a new item from thousands of items that a user has never clicked is a very difficult task, so the AP and nDCG of all comparison methods are relatively low. We find that the overall results for **TR-50%** are higher than the results for **TR-80%**. This is because 50% of the withheld data is used as the testing set for **TS-50%** so the hit probability is naturally higher than it is for the 20% of withheld testing data **TS-20%**.

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

[2] http://tianchi.aliyun.com/datalab/introduction.htm?id=1.

Table VI. The Mean AP@5,10 and nDCG@5,10 on Tmall.com Dataset

Target Domain Method	*D1*							
	TR-80%				TR-50%			
	AP@5	AP@20	nDCG@5	nDCG@20	AP@5	AP@20	nDCG@5	nDCG@20
Most-Pop	0.0161^	0.0175^	0.0269^	0.0382^	0.0322^	0.0223^	0.0567^	0.0577^
N-CDCF	0.0252*	0.0240*	0.0441*	0.0465*	0.0352*	0.0210	0.0604*	0.0534
MF-IF	0.0263*	0.0293*	0.0432*	0.0631*	0.0455*	0.0324	0.0813*	0.0854*
MF-IF-CDCF	0.0242*	0.0258*	0.0399*	0.0552*	0.0431*	0.0296	0.0763*	0.0775*
PARAFAC2	0.0213*	0.0226*	0.0350*	0.0476*	0.0395*	0.0267	0.0691*	0.0687*
CDTF-IF	0.0258*	0.0276*	0.0425*	0.0587*	0.0423*	0.0294	0.0758*	0.0767*
WITF	0.0267*	0.0285*	0.0451*	0.0623*	0.0484*	0.0340	0.0849*	0.0872*
WITF+WRMF	**0.0271****	**0.0290****	**0.0462****	**0.0643****	**0.0486****	**0.0343****	**0.0851****	**0.0879****
Target Domain Method	*D2*							
	TR-80%				TR-50%			
	AP@5	AP@20	nDCG@5	nDCG@20	AP@5	AP@20	nDCG@5	nDCG@20
Most-Pop	0.0175^	0.0194^	0.0288^	0.0424^	0.0297^	0.0231^	0.0530^	0.0591^
N-CDCF	0.0281*	0.0261*	0.0435*	0.0520*	0.0228	0.0243*	0.0380	0.0357
MF-IF	0.0320*	0.0354*	0.0528*	0.0747*	0.0501*	0.0370*	0.0872**	0.0924**
MF-IF-CDCF	0.0240*	0.0262*	0.0397*	0.0563*	0.0380*	0.0285*	0.0675	0.0724*
PARAFAC2	0.0215*	0.0234*	0.0356*	0.0506*	0.0327*	0.0251*	0.0589*	0.0638*
CDTF-IF	0.0326*	0.0337*	0.0526*	0.0662*	0.0454*	0.0316*	0.0761*	0.0750*
WITF	0.0338*	0.0363*	0.0552*	0.0753*	0.0538*	0.0383*	0.0905*	0.0909*
WITF+WRMF	**0.0343****	**0.0369****	**0.0556****	**0.0758****	**0.0542****	**0.0386****	**0.0907****	**0.0915***

^baseline, $^*p < 0.01$, **smallest p.

The baseline method, Most-Pop, underperforms all comparison methods. This is because popular items only account for a small proportion (a short head) and most users have their personally preferred items in the long tail. Anderson's well-known research in economics proved this phenomenon, and Anderson [2006] suggested that future business would obtain more profit from long-tail selling. Most-Pop is unable to find the personally preferred items in the long tail, which leads to the poorest performance. N-CDCF also does not perform well, and the reason is similar to Most-Pop—that is, the preferred long-tail items for each user are quite different, so using neighbors' clicked long-tail items as the prediction result deviates significantly from users' true preferences.

The single domain method, MF-IF, achieves relatively good performance because it exploits both the clicked items and the unclicked ones for each user to models their implicit preferences, i.e., the observed data as positive instances and unobserved data as uncertainly negative instances. Thus, it not only resolves the unclassifiable issue on unary (one-class) preference data [Pan et al. 2008] but also relieves the data sparsity issue due to the small amount of observed data. Although MF-IF-CDCF also exploit the unobserved data, its results are worse than those of MF-IF, which can be mainly attributed to the heterogeneities between domains. That is, much more data (both observed and unobserved ones) from heterogeneous auxiliary domains overwhelm the learning of user factors in the target domain. PARAFAC2 is based on binary assumption which treats the unclicked items as true negative instances and assigns too strong dislike weights to them. In addition, PARAFAC2 cannot control the influence from auxiliary domains. These two deficiencies result in it even underperforming MF-IF-CDCF. In comparison, CDTF-IF surpasses MF-IF-CDTF since it controls the mutual influence between domains. However, the design of CDTF-IF is defective when assigning different confidence levels [Hu et al. 2013] on clicked and unclicked items. In

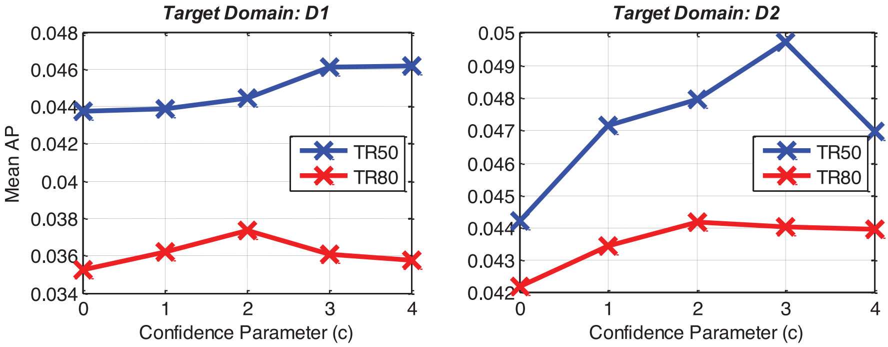


Fig. 8. Mean APs over different values of confidence parameter c.

fact, the different confidence levels do not take effect when training the model. In this article, WITF fixes this defect and we devise a more efficient algorithm to learn the parameters, with the result that it outperforms CDTF-IF. After fine-tuning, the model WITF+WRMF achieves a level of improvement over WITF.

6.4.3. The Impact of Confidence Parameter. Click records are typical unary preference data so we cannot directly differentiate user preference levels over them. As presented in related work, both observed data and unobserved data should be exploited to resolve this problem. In Section 4.4.1, the confidence parameter $c_{k,i,j}$ (cf. Equation (28)) is introduced to emphasize the observed clicks. Although $c_{k,i,j}$ can be tuned for each user and item, respectively [Hu et al. 2014; Pan et al. 2008], it is beyond the main topic of this article. In this experiment, we use a single confidence parameter c for all domains [Hu et al. 2008] since we mainly focus on evaluating the impact of c using different values.

We increase the value of c from 0 to 4 to evaluate the change of mean AP for WITF. Figure 8 plots the change of performance in the target domain *D1* and *D2* respectively. It degenerate to the case of binary preference data when $c = 0$ is set, i.e., the weights on observed clicks (positive instances) and unobserved clicks (negative instances) are the same. As presented previously, those unclicked items in the training sets **TR-80%** and **TR-50%** do not necessarily indicate user dislike; moreover, a part of truly observed clicks has been held out for testing. Therefore, a lot of unclicks in these training sets are not true negative instances. Hence, it is improper to learn model parameters under binary preference assumption. As a result, we find that the left-hand plots and the right-hand ones in Figure 8, $c = 0$ underperforms the other settings. When c is increased, we find that the performance is improved. Specially, the best performance is achieved over **TR-80%** when c is around 2 while the best performance is achieved over **TR-50%** when c is around 3. It can be interpreted that the amount of observed clicks in **TR-50%** is less than that in **TR-80%**, hence a larger c is required to emphasize these positive instances in **TR-50%**. Intuitively, too large c on observed clicks will overwhelm the information learned from unclicks. As expected, the performance drops when we continue increasing c to a larger value.

6.4.4. Recall Over Different User Groups. The metric recall@k effectively assesses if a recommender system has successfully generated a personalized item list for which a user has shown a positive preference. In this experiment, we separated the testing users in the target domain into two groups, namely the *Cold-Start User Group* where users had fewer than 10 clicks and the *Experienced User Group* where users had clicked

Table VII. Statistics of Testing Users Grouped by the Number of Ratings

Target Domain / Group	# Clicks	*D1*	*D2*
		# testing users in TS-50%	# testing users in TS-50%
Experienced	>10	1,000	617
Cold-Start	1∼10	3,779	3,552

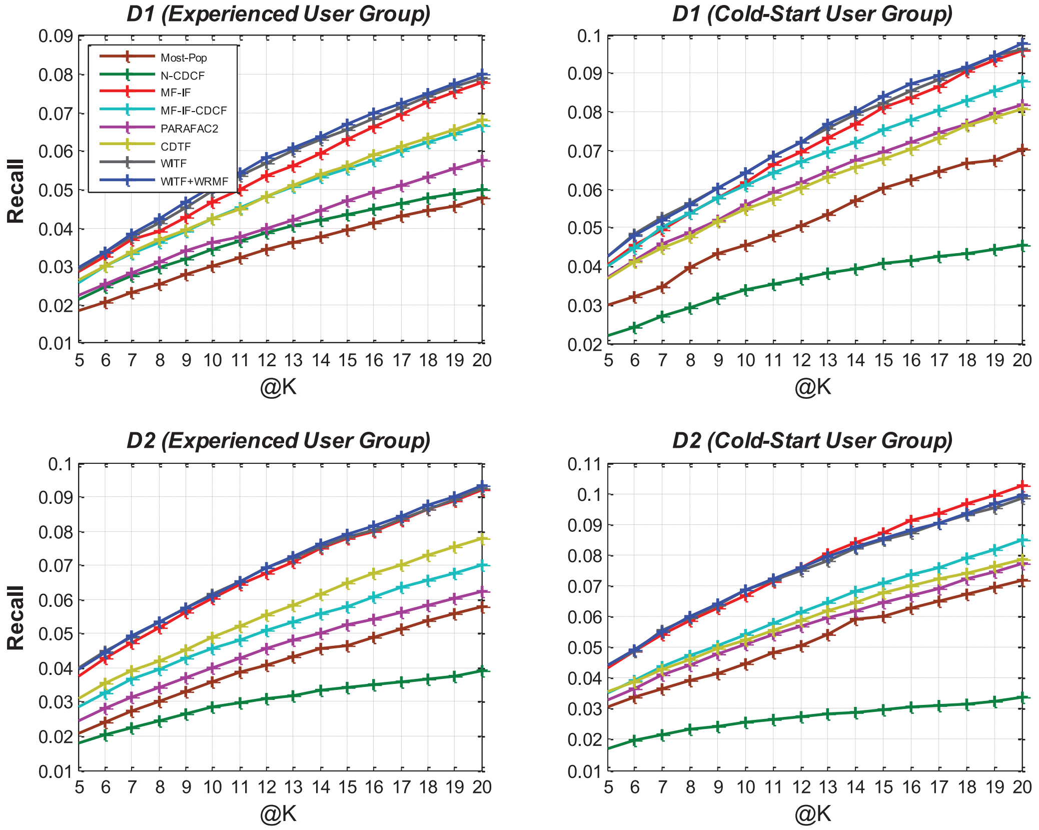


Fig. 9. Recall@5∼20 of comparison methods on target domain D1 and D2.

more than 10 items. Table VII shows statistics of these two groups in the target domain *D1* and *D2* respectively.

We evaluated recall@5∼20 over all comparison methods and Figure 9 illustrates the results. Most-Pop does not achieve a high recall and the margin between it and other methods widens as *K* increases, which illustrates that users have different long-tail items of interest apart from the most popular items. The performance of N-CDCF is even worse than that of Most-Pop, especially for the Cold-Start User Group. This is because the similarity of neighbors is computed from the data of all domains, hence the neighbors are largely determined by the data in auxiliary domains when a user is cold start in the target domain. However, user preferences in each domain are quite different, and each neighbor of a user tends to have personally preferred long-tail items in the target domain. Accordingly, the clicked items from neighbors are quite varied, which makes the prediction from N-CDCF aimless. MF-IF-CDCF, PARAFAC2, and CDTF-IF underperform MF-IF, for the reason explained in the previous experiment, i.e., blindly borrowing the knowledge of user preferences from other domains may even

Table VIII. The Average Time (in Seconds) to Run An Iteration

#CPUs Method	1 CPU	4 CPUs	8 CPUs	12 CPUs
WITF (RAW)	50.83	28.64	24.89	22.53
WITF	21.78	17.05	15.27	13.31
WRMF	2.91	2..26	1.87	1.58

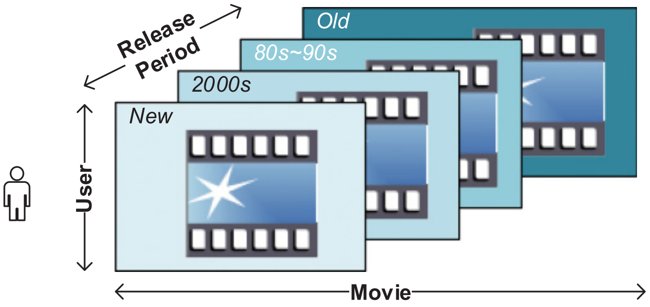


Fig. 10. Time period as domain to model the heterogeneities of user preferences on movies.

hurt preference learning from the target domain due to the heterogeneities between domains. In comparison, as shown in all four sub-figures of Figure 9, the plots of WITF and WITF+WRMF are above those of other methods with apparent margins. Therefore, we conclude that WITF provides an effective way to model the unary preference data, such as clicks, over multiple domains, which enables personal preferences for target domain items to be better captured by this method than by other methods.

6.4.5. Training Time Comparison. Time cost is a critical problem to take into account when a model is deployed in a real-world production environment. In this article, we designed two strategies to reduce the time cost: (1) a parallelizable parameter learning algorithm (cf. Algorithm 1); (2) the trick to reduce time complexity (cf. Section 4.4.1).

We trained our models on a cluster with 12 Intel® Xeon® CPUs and 94G memory. Our parallel parameter learning algorithm is implemented with MATLAB® Parallel Computing Toolbox. Table VIII illustrates the average time to run an iteration under using different number of CPUs, where WITF (RAW) denotes the model without taking the time complexity trick (cf. Section 4.4.1). Obviously, WITF using the time complexity trick is almost twice faster than WITF (RAW), and such improvement will be more significant when more items are involved in a real-world production environment. The time cost reduces when more CPUs are used, which proves the effectiveness of our parallel learning algorithm. The speed can be further improved when the model is implemented with some industrial parallel technique in a production environment. Moreover, the time cost of WRMF for post-learning takes few seconds, and it only needs to be run a few iterations on the basis of the well-estimated parameters from WITF.

6.5. Time Period as Domain on Movie Rating Prediction

Watching movies is one of the most frequent recreational activities. People usually see newly released movies because of a movie's popularity or shared tastes with friends, whereas they may watch old movies for nostalgia, or their relation to new movies. That is, the attractiveness of new movies vs. old movies is quite different. This suggests that movies should be grouped by different time periods to better represent such heterogeneities. As shown in Figure 10, the movies in our experiment are grouped into four slices from *New* to *Old* according to their release period. This construction

Table IX. Statistics of the MovieLens Dataset over Different Release Time Periods

Domain	Time Period	#Movies	# Ratings / # Users	# Ratings / # Movies	Sparsity
Old*	∼1979	6,750	98.8418	83.8471	0.0146
80s90s	1980∼1999	3,385	106.5423	180.2248	0.0315
2000s	2000∼2012	7,850	210.2548	153.3655	0.0268
New*	2013∼	1,138	18.6030	93.6037	0.0163

*Each of these domains is chosen as the target domain for evaluation, respectively.

Table X. MAEs and RMSEs of Comparison Methods on MovieLens Dataset

Target Domain	*New (TR-50%)*		*Old (TR-50%)*	
Method	MAE	RMSE	MAE	RMSE
kNN-CDCF	0.8552	1.1107	0.7916	1.0395
PMF	0.6758	0.9153	0.6277	0.8159
PMF-CDCF	0.5995$^\wedge$	0.8047$^\wedge$	0.5844$^\wedge$	0.7745$^\wedge$
CMF	0.5996	0.8111	0.5903	0.7891
PARAFAC2	0.6012	0.8069	0.5723*	0.7638*
CDTF	0.5986*	0.8087*	0.5655*	0.7572*
WITF	0.5978*	0.8043*	0.5622*	0.7510*
WITF+WRMF	**0.5976****	**0.8041****	**0.5620****	**0.7508****

$^\wedge$baseline, $^*p < 0.01$, **smallest p.

implies a cross-domain recommendation problem in which time periods serve as domains. Therefore, we can apply our WITF methods to better capture user preferences from the high-order interaction between users, movies, and time periods.

6.5.1. Data Preparation. In this experiment, we used the MovieLens20M dataset[3] which contains the latest movies released prior to 04/2015. We grouped the movies into four domains according to their release date, as reported in Table IX. We removed those users who had watched fewer than three new movies from the raw dataset so that we extracted 5,726 users for evaluation. Table IX summarizes the statistics for each domain. In this experiment, we use *New* and *Old* as the respective target domains for evaluation. For each target domain, we create a training set **TR-50%** and a testing set **TS-50%**, just as in previous experiments.

6.5.2. Rating Prediction Performance Comparison. Table X reports the MAEs and RMSEs on the target domain *New* and *Old* respectively over all testing users. It is interesting to observe that the prediction performance on the target domain *Old* is overall better than the performance on the target domain *New*. This phenomenon can be interpreted as follows. A user tends to watch new movies due to the movie's popularity, promotion and social relationships, that is, the feedback on new movies is more or less influenced the opinions of others. Old movies, by comparison, are no longer hot topics, so users tend to watch them and give feedback only if they are personally really interested in these movies. The randomness of feedback on new movies is therefore much higher than it is on old movies, and this is why most approaches more easily capture the user preferences on movies in the *Old* domain and consequently make better predictions.

From Table X, we find that the margins between the different methods are relatively small, and the improvement between the TF-based methods and MF-based methods is not so obvious compared to the previous experiments. This is because all the items in each domain are movies, so the heterogeneities are much smaller than in the previous experiments. As a result, the MF-based methods, e.g. PMF-CDCF, do not suffer from

[3] http://grouplens.org/datasets/movielens/.

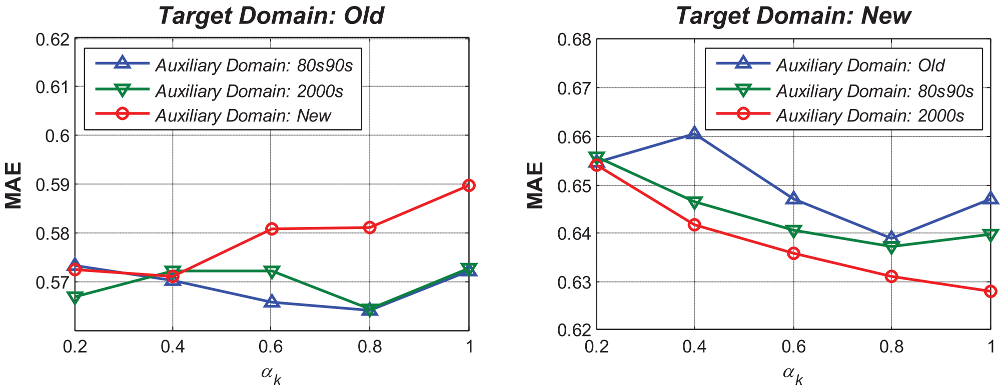


Fig. 11. MAE comparison with α_k increasing.

significant heterogeneity issues when leveraging information from auxiliary domains. Moreover, we find the apparent margins by comparing the performance of PMF and PMF-CDCF in the *New* domain, which suggests that it is helpful to incorporate more data to train models, since the available user feedback within a small time period is relatively little. WITF-based models outperform the other methods thanks to the sophisticated heterogeneity representation and influence control mechanism. Table X also reports the sign tests performed between the baseline, i.e., PMF-CDCF, and other models. The results of comparison models siginificantly outperform the results of baseline are marked with * (p-value < 0.01).

6.5.3. The Amount of Influence from Auxiliary Domains. To determine how much amount of information to be leveraged from auxiliary domains has a direct influence on prediction performance in the target domain. In Section 4.4.2, we presented a way to trade off the influence between domains in terms of setting the proportional parameter α_k (cf. Equation (30)). In this experiment, we use WITF to evaluate the rating prediction performance in the target domain *New* and *Old* respectively by changing α_k. We use only a single auxiliary domain to investigate its influence on the target domain in each case.

Figure 11 depicts MAEs changing with the proportional parameter α_k which controls the influence of the auxiliary domains. The results show that performance changes with the value of α_k. In both cases, we find that the prediction performance reaches its best point when a certain amount of influence is borrowed from auxiliary domain. Either too large or too small α_k may degenerate the performance.

From the left figure, i.e., the case of *Old* movies as the target domain, the best α_k is about 0.4 when *New* movies serves as the auxiliary domain, and the performance gets worse when α_k increases. In comparison, the best α_k is around 0.8 when *80s90s* movies serves as the auxiliary domain. This proves that the user behavior on watching new movies is quite different from watching old ones, so incorporating too much influence from *New* movies domain may hurt the performance. Comparatively, the time gap between *Old* movies and *80s90s* movies is naturally smaller than that between *Old* movies and *New* movies, so their heterogeneities are correspondingly smaller. As a result, the target domain *Old* can incorporate more information from *80s90s* auxiliary domain, i.e., a relative larger α_k.

From the right figure, i.e. the case of *New* movies as the target domain, similar phenomenon can be observed. The best α_k is about 0.8 when *Old* movies serves as the auxiliary domain whereas $\alpha_k = 1$ still does not reach the best point and we can continue to set it a larger value when the auxiliary domain *2000s* movies. Furthermore, we can find that the MAE using *2000s* as the auxiliary domain is consistently smaller than the MAE using either *Old* or *80s90s* as the auxiliary. This proves that the heterogeneities of user preferences between *New* movies and *Old* movies is relative larger than those between *New* and others. As a result, it is helpful to set a smaller α_k so as to control the knowledge learned from the domain of *Old* movies. In comparison, the user preferences on *2000s* movies are relatively closer to those on the *New* movies, so a larger α_k is preferred to transfer relatively more information learned from this auxiliary domain. Therefore, the optimal configuration of α_k can be used to quantify the degree of heterogeneity between target domain and auxiliary domain.

7. CONCLUSION AND DISCUSSION

In this article, we discussed the emerging requirements of CDCF in the current era and analyzed the limitations of the current CDCF approaches in users' cold-start domains. Hence, we proposed WITF to model the triadic relation *user-item-domain*. Specially, WITF addresses an irregular tensor factorization problem where each domain having their specific item set. Furthermore, we designed an effective post-learning procedure to fine-tune the user and item factors that are learned from WITF. In addition, our framework is able to deal with explicit preference data, e.g., ratings, and implicit preference data, e.g., clicks, in a unified algorithm.

We evaluated our approach on three typical real-world application scenarios. The evidence from all the results has shown that our approach outperforms all other state-of-the-art methods, especially for cold-start cases. This is because WITF can better capture the domain-specific user preference through the triadic relation between users, items and domains, whereas traditional models only model the dyadic relation so they fail to represent the heterogeneities when transferring knowledge between domains.

The recent literature on cross-domain recommender systems [Cantador et al. 2015] defines domains from four different levels: (1) attribute level: same type of items with different values of certain attribute (e.g. movies with different genres); (2) type level: similar types of items (e.g. movies and videos); (3) item level: different types of items (movies and books); (4) system level: same type of items on different systems (e.g. movies in IMDb.com and Netflix.com). Obviously, the levels (1), (2), and (3) are trivially supported by our approach, and the empirical studies with several real-world datasets has covered these cases. In this article, we do not demonstrate the case of system-level CDCF, but it is naturally supported by our approach. To date, many systems encourage to sign in using the third part account, e.g. Facebook, Google+. For example, a user may have rated many movies in IMDb.com with a Facebook account. Now, she uses this account to sign in Netflix.com as a new user (a typical cold-start user to a single-domain recommender system). By applying WITF, we can predict her preferences on TV shows in Netflix.com by transferring the knowledge learned from IMDb.com.

Although we focus on studying the CDCF problem in this paper, tensor factorization was originally applied in many other research problems [Kolda and Bader 2009] that are quite different from recommendation. Therefore, it is expected that our models can be applied in other areas, such as spectral detection, natural language processing.

APPENDIX

This appendix gives the proof of Theorem 4.1 on equivalence of the optimization problem between Equation (15) and Equation (19)

Proof. From Equation (15), we have

$$\begin{aligned}
\mathcal{J} &= \underset{\boldsymbol{U},\boldsymbol{V},\boldsymbol{C}}{argmin} \sum_k \sum_i \|\boldsymbol{W}_{k,i,:} \circledast \boldsymbol{X}_{k,i,:} - \boldsymbol{W}_{k,i,:} \circledast \boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}}\|_F^2 \\
&= \underset{\boldsymbol{U},\boldsymbol{V},\boldsymbol{C}}{argmin} \sum_k \sum_i \left\|\left(\boldsymbol{W}_{k,i,:} \circledast \boldsymbol{X}_{k,i,:} - \boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}}\right) + \boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}}\left(\boldsymbol{I} - diag\boldsymbol{W}_{k,i,:}\right)\right\|_F^2 \\
&= \underset{\boldsymbol{U},\boldsymbol{V},\boldsymbol{C}}{argmin} \sum_k \sum_i \|\boldsymbol{W}_{k,i,:} \circledast \boldsymbol{X}_{k,i,:} - \boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}}\|_F^2 + \|\boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}}(\boldsymbol{I} - diag\boldsymbol{W}_{k,i,:})\|_F^2 \\
&\quad + 2\left(\boldsymbol{W}_{k,i,:} \circledast \boldsymbol{X}_{k,i,:} - \boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}}\right)\left[\boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}}(\boldsymbol{I} - diag\boldsymbol{W}_{k,i,:})\right]^{\mathrm{T}}
\end{aligned}$$

Let $\boldsymbol{Z}_k = \ddot{\boldsymbol{W}}_k \circledast \boldsymbol{X}_k$, we obtain

$$\begin{aligned}
\mathcal{J} &= \underset{\boldsymbol{U},\boldsymbol{V},\boldsymbol{C}}{argmin} \sum_k \sum_i \|W_{k,i,:} \circledast \boldsymbol{X}_{k,i,:}\|_F^2 + \|\boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}}\|_F^2 - 2\boldsymbol{Z}_{k,i,:}\left[\boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}}\right]^{\mathrm{T}} \\
&\quad + \boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}}(diag\ddot{\boldsymbol{W}}_{k,i,:} - \boldsymbol{I})\left[\boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}}\right]^{\mathrm{T}}
\end{aligned}$$

Let $\boldsymbol{H}_{k,i,:} = \sqrt{\ddot{\boldsymbol{W}}_{k,i,:} - 1}$ where $\sqrt{\cdot}$ denotes element-wise square root, and add the dummy term $(\|\boldsymbol{Z}_k\|_F^2 - \|\boldsymbol{Z}_k\|_F^2)$, we obtain

$$\begin{aligned}
\mathcal{J} &= \underset{\boldsymbol{U},\boldsymbol{V},\boldsymbol{C}}{argmin} \left\{\sum_k \sum_i \|\boldsymbol{Z}_{k,i,:}\|_F^2 + \|\boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}}\|_F^2 - 2\boldsymbol{Z}_{k,i,:}\left[\boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}}\right]^{\mathrm{T}}\right. \\
&\quad \left. + \|\boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}} \circledast \boldsymbol{H}_{k,i,:}\|_F^2\right\} + \sum_k \left(\|\boldsymbol{W}_k\boldsymbol{X}_k\|_F^2 - \|\boldsymbol{Z}_k\|_F^2\right)
\end{aligned}$$

The term $\sum_k(\|\boldsymbol{W}_k \circledast \boldsymbol{X}_k\|_F^2 - \|\boldsymbol{Z}_k\|_F^2)$ does not contains any parameter for optimization, so it can be removed, and we obtain

$$\begin{aligned}
\mathcal{J} &= \underset{\boldsymbol{U},\boldsymbol{V},\boldsymbol{C}}{argmin} \sum_k \sum_i \left\{\|\boldsymbol{Z}_{k,i,:} - \boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}}\|_F^2 + \|\boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}} \circledast \boldsymbol{H}_{k,i,:}\|_F^2\right\} \\
&= \underset{\boldsymbol{U},\boldsymbol{V},\boldsymbol{C}}{argmin} \sum_k \|\boldsymbol{Z}_k - \boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k\boldsymbol{V}^{\mathrm{T}}\boldsymbol{P}_k^{\mathrm{T}}\|_F^2 + \sum_k \|\boldsymbol{U}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}} \circledast \boldsymbol{H}_k\|_F^2
\end{aligned}$$

Using the constraint $\boldsymbol{P}_k^{\mathrm{T}}\boldsymbol{P}_k = \boldsymbol{I}$ and adding the dummy term $\|\boldsymbol{Z}_k\boldsymbol{P}_k\|_F^2 - \|\boldsymbol{Z}_k\boldsymbol{P}_k\|_F^2$, we obtain

$$\mathcal{J} = \underset{\boldsymbol{U},\boldsymbol{V},\boldsymbol{C}}{argmin} \sum\nolimits_k \|\boldsymbol{Z}_k\boldsymbol{P}_k - \boldsymbol{U}_{i,:}\boldsymbol{\Sigma}_k\boldsymbol{V}^{\mathrm{T}}\|_F^2 + \sum_k \|\boldsymbol{U}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}} \circledast \boldsymbol{H}_k\|_F^2 + \sum_k \left(\|\boldsymbol{Z}_k\|_F^2 - \|\boldsymbol{Z}_k\boldsymbol{P}_k\|_F^2\right)$$

The term $\sum_k(\|\boldsymbol{Z}_k\|_F^2 - \|\boldsymbol{Z}_k\boldsymbol{P}_k\|_F^2)$ is not related to optimization, it can be removed. Let $\hat{\boldsymbol{X}}_k^{\mathrm{T}} = \boldsymbol{U}\boldsymbol{\Sigma}_k(\boldsymbol{P}_k\boldsymbol{V})^{\mathrm{T}}$, we obtain

$$\mathcal{J} = \underset{\boldsymbol{U},\boldsymbol{V},\boldsymbol{C}}{argmin} \|\boldsymbol{\mathcal{Y}} - [\![\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{C}]\!]\|^2 + \sum\nolimits_k \|\hat{\boldsymbol{X}}_k^{\mathrm{T}} \circledast \boldsymbol{H}_k\|_F^2$$

where $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{N\times R\times K}$ is a third-order tensor with K slices: $\boldsymbol{Y}_k = \boldsymbol{Z}_k\boldsymbol{P}_k$. □

REFERENCES

E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Morup. 2010. Scalable tensor factorizations with missing data. In *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM, 701–712.

E. Acar, T. G. Kolda, and D. M. Dunlavy. 2009. An optimization approach for fitting canonical tensor decompositions. In Sandia National Laboratories, Tech. Rep. SAND2009-0857.

D. Agarwal and B. C. Chen. 2009. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 19–28.

C. Anderson. 2006. *The Long Tail: Why the Future of Business is Selling Less of More*. Hachette Digital, Inc.

Y. Bengio, A. Courville, and P. Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Patt. Anal. Mach. Intel*, 35, 1798–1828.

C. M. Bishop. 1995. Training with noise is equivalent to tikhonov regularization. *Neural Comput.* 7, 108–116.

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, (Bonn, Germany). ACM, 1102363, 89–96.

I. Cantador, I. Fernández-Tobías, S. Berkovsky, and P. Cremonesi. 2015. Cross-domain recommender systems. In *Recommender Systems Handbook*, F. Ricci, L. Rokach, and B. Shapira (Eds.). Springer US, 919–959.

L. Cao. 2015. Coupling learning of complex interactions. *Info. Process. Manage.* 51, 167–186.

N. Cliff. 1966. Orthogonal rotation to congruence. *Psychometrika* 31, 33–42.

D. M. Dunlavy, T. G. Kolda, and E. Acar. 2011. Temporal link prediction using matrix and tensor factorizations. *ACM Trans. Knowl. Discov. Data* 5, 1–27.

K. Georgiev and P. Nakov. 2013. A non-iid framework for collaborative filtering with restricted Boltzmann machines. In *Proceedings of the 30th International Conference on Machine Learning*, USA2013 JMLR: W&CP.

J. C. Gower and G. B. Dijksterhuis. 2004. *Procrustes Problems*. Oxford University Press Oxford.

P. F. Groenen, P. Giaquinto, and H. L. Kiers. 2005. An improved majorization algorithm for robust procrustes analysis. In *New developments in Classification and Data Analysis.* Springer, Berlin, 151–158.

A. Gunawardana and G. Shani. 2015. Evaluating recommender systems. In *Recommender Systems Handbook*, F. Ricci, L. Rokach, and B. Shapira (Eds.). Springer US, 265–308.

R. A. Harshman. 1972. Parafac2: Mathematical and technical notes. *UCLA Working Papers in Phonetics* 22, 30–44.

J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.* 22, 5–53.

T. Hofmann. 2004. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.* 22, 89–115.

L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and C. Zhu. 2013. Personalized recommendation via cross-domain triadic factorization. In *Proceedings of the 22nd International Conference on World Wide Web*, Rio de Janeiro, Brazil. International World Wide Web Conferences Steering Committee, 2488441, 595–606.

L. Hu, W. Cao, J. Cao, G. Xu, L. Cao, and Z. Gu. 2014. Bayesian heteroskedastic choice modeling on non-identically distributed linkages. In the *Proceedings of the 2014 IEEE International Conference on Data Mining (ICDM)*, 851–856.

Y. Hu, Y. Koren, and C. Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Eighth IEEE International Conference on Data Mining*, 15–19 Dec. 2008 2008, 263–272.

S. Huang, S. Wang, T. Y. Liu, J. Ma, Z. Chen, and J. Veijalainen. 2015. Listwise collaborative filtering. In *Proceedings of the Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Santiago, Chile.ACM, 2767693, 343–352.

A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. 2010. Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the 4th ACM Conference on Recommender Systems*, ACM, 1864727, 79–86.

H. A. Kiers. 1997. Weighted least squares fitting using ordinary least squares algorithms. *Psychometrika* 62, 251–266.

H. A. L. Kiers, J. M. F. Ten Berge, and R. Bro. 1999. Parafac2—part i. A direct fitting algorithm for the parafac2 model. *J. Chemometrics* 13, 275–294.

T. G. Kolda and B. W. Bader. 2009. Tensor decompositions and applications. *SIAM Rev.* 51, 455–500.

Y. Koren, R. Bell, and C. Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 30–37.

B. Li. 2011. Cross-domain collaborative filtering: A brief survey. In *Proceedings of the 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence 2011 IEEE Computer Society*, 2084563, 1085–1086.

B. Li, Q. Yang, and X. Xue. 2009a. Can movies and books collaborate? Cross-domain collaborative filtering for sparsity reduction. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*. 2052–2057.

B. Li, Q. Yang, and X. Xue. 2009b. Transfer learning for collaborative filtering via a rating-matrix generative model. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ACM, 1553454, 617–624.

N. N. Liu and Q. Yang. 2008. Eigenrank: A ranking-oriented approach to collaborative filtering. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Singapore, Singapore2008 ACM, 1390351, 83–90.

M. Long, J. Wang, G. Ding, W. Cheng, X. Zhang, and W. Wang. 2012. Dual transfer learning. In *Proceedings of the 12th SIAM International Conference on Data Mining* 2012, 540–551.

B. Loni, Y. Shi, M. Larson, and A. Hanjalic. 2014. Cross-domain collaborative filtering with factorization machines. In *Advances in Information Retrieval*, M. De Rijke, T. Kenter, A. de Vries, C. Zhai, F. de Jong, K. Radinsky, and K. Hofmann (Eds.). Springer International Publishing, 656–661.

M. Mørup. 2011. Applications of tensor (multiway array) factorizations and decompositions in data mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1, 24–40.

P. Niyogi, F. Girosi, and T. Poggio. 1998. Incorporating prior information in machine learning by creating virtual examples. *Proc. IEEE* 86, 2196–2209.

R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. 2008. One-class collaborative filtering. In *Proceedings of the IEEE International Conference on Data Mining.* IEEE, 502–511.

W. Pan, E. W. Xiang, N. N. Liu, and Q. Yang. 2010. Transfer learning in collaborative filtering for sparsity reduction. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence 2010*.

I. Porteous, A. U. Asuncion, and M. Welling. 2010. Bayesian matrix factorization with side information and dirichlet process mixtures. In *AAAI 2010*.

S. Rendle. 2010. Factorization machines. In *Proceedings of the IEEE 10th International Conference on Data Mining (ICDM)*, 2010, 995–1000.

S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. 2009a. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence* (Montreal, Quebec, Canada. AUAI Press, 1795167, 452–461.

S. Rendle, L. B. Marinho, A. Nanopoulos, and L. Schmidt-Thieme. 2009b. Learning optimal ranking with tensor factorization for tag recommendation. In *Proceedings of the 15th ACM SIGKDD International Conference On Knowledge Discovery and Data Mining*, ACM, 1557100, 727–736.

S. Rendle and L. Schmidt-Thieme. 2010. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*. ACM, 1718498, 81–90.

P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. 1994. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, ACM, 192905, 175–186.

R. Salakhutdinov and A. Mnih. 2008. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems 2008*, 1257–1264.

R. Salakhutdinov, A. Mnih, and G. Hinton. 2007. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*. ACM, 1273596, 791–798.

B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, ACM, 372071, 285–295.

A. I. Schein, A. Popescul, L. H. Ungar and D. M. Pennock. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, 564421, 253–260.

A. P. Singh and G. J. Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Las Vegas, Nevada, USA2008 ACM, 1401969, 650–658.

N. Srebro and T. Jaakkola. 2003. Weighted low-rank approximations. In *Proceedings of the 20th International Conference on Machine Learning*, 720.

N. Srebro, J. D. M. Rennie, and T. Jaakkola. 2005. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems* 1329–1336.

X. Su and T. M. Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 19.

J. Tang, H. Gao, H. Liu, and A. D. Sarma. 2012. Etrust: Understanding trust evolution in an online world. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2339574, 253–261.

K. Train. 2003. *Discrete Choice Methods with Simulation*. Cambridge University Press.

P. Vincent, H. Larochelle, Y. Bengio, and P. A. Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ACM, 1390294, 1096–1103.

L. Xiong, X. Chen, T. K. Huang, J. Schneider, and J. G. Carbonell. 2010. Temporal collaborative filtering with Bayesian probabilistic tensor factorization. In *Proceedings of SIAM Data Mining* 2010.

Z. H. Zhou and X. Y. Liu. 2010. On multi-class cost-sensitive learning. *Computat Intel* 26, 232–257.

Received August 2015; revised May 2016; accepted July 2016