Heterogeneous Recommendation via Deep Low-Rank Sparse Collective Factorization

Shuhui Jiang, Zhengming Ding, *Member, IEEE*, and Yun Fu, *Fellow, IEEE*

Abstract—A real-world recommender usually adopts heterogeneous types of user feedbacks, for example, numerical ratings such as 5-star grades and binary ratings such as likes and dislikes. In this work, we focus on transferring knowledge from binary ratings to numerical ratings, facing a more serious data sparsity problem. Conventional Collective Factorization methods usually assume that there are shared user and item latent factors across multiple related domains, but may ignore the shared common knowledge of rating patterns. Furthermore, existing works may also fail to consider the hierarchical structures in the heterogeneous recommendation scenario (i.e., genre, sub-genre, detailed-category). To address these challenges, in this paper, we propose a novel Deep Low-rank Sparse Collective Factorization (DLSCF) framework for heterogeneous recommendation. Specifically, we adopt low-rank sparse decomposition to capture the common rating patterns in related domains while splitting the domain-specific patterns. We also factorize the model in multiple layers to capture the affiliation relation between latent categories and sub-categories. We propose both batch and Stochastic Gradient Descent (SGD) based optimization algorithms for solving DLSCF. Experimental results on MoviePilot, Netfilx, Flixter, MovieLens10M and MovieLens20M datasets demonstrate the effectiveness of the proposed algorithms, by comparing them with several state-of-the-art batch and SGD based approaches.

Index Terms—Recommendation, cross-domain, collaborative factorization, low-rank decomposition

1 Introduction

In real-world recommender systems, usually heterogeneous types of user feedbacks are applied, for example, binary ratings such as likes & dislikes and numerical ratings such as 5-star grades. For binary ratings, users are willing to provide feedbacks due to the easy operation in which they only need to click the like or dislike button. Compared to binary ratings, 5-star ratings provide not only the like or dislike information, but also how much he or she likes the item. Though 5-star ratings contain more comprehensive information, there are usually more serious data sparsity problems due to the less convenient operation. The sparsity problem means that users may only rate a very limited number of items in contrast with the huge item space. In this paper, we aim to borrow the knowledge from binary ratings to address the data sparsity problem in 5-star ratings. We work on the scenario that users may provide some binary ratings to product set B and some 5-star ratings to product set C. There is a very small or no overlap between set B and C. Our basic assumption is that users' preferences towards items or categories are consistent across numerical ratings and binary ratings, although the rating scales are different. Thus, there are shared knowledge between these two domains.

- S. Jiang is with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115. E-mail: shjiang@ece.neu.edu.
- Z. Ding is with the Department of Computer, Information and Technology, Indiana University-Purdue University Indianapolis, Indianapolis, IN 46202. E-mail: zd2@iu.edu.
- Y. Fu is with the Department of Electrical and Computer Engineering, College of Engineering, Khoury College of Computer and Information Sciences, Northeastern University, Boston, MA 02115. E-mail: yunfu@ece.neu.edu.

Manuscript received 30 Jan. 2018; revised 21 Oct. 2018; accepted 28 Nov. 2018. Date of publication 21 Jan. 2019; date of current version 1 Apr. 2020. (Corresponding author: Shuhui Jiang.)
Recommended for acceptance by G. Chechik.
Digital Object Identifier no. 10.1109/TPAMI.2019.2894137

Cross-domain collaborative filtering (CDCF) exploits useful knowledge from auxiliary domains to facilitate the recommendation in the target domain, when there is a data sparsity problem in the target domain [1], [2], [3], [4], [5]. In this paper, we address the sparsity problem of heterogeneous recommendation. We transfer binary ratings (i.e., likes/dislikes) from the auxiliary domain to numerical ratings (e.g., 5-star grades) in the target domain. Only a few previous works have worked on the same scenario [1], [2], [6], [7], [8]. Liu et al. [1] applied a cross-domain collective matrix factorization (CMF) to transfer the "whether rate" binary rating data [6] for numerical rating prediction, while Zhang et al. applied "whether purchased" binary rating data [7]. Pan at el. proposed to transfer the shared user latent factor and item latent factor through a Transfer by Collective Factorization (TCF) framework [2], [8]. TCF is most related to our work. However, TCF learns domain specific rating patterns of each domain and may neglect the shared knowledge in rating patterns. Recent works demonstrate that there are a certain part of shared rating patterns of related domains [9], [10], [11], [12], [13], [14]. It remains a challenging problem to collectively learn latent user and item latent factors and shared rating patterns.

Furthermore, in the cross-domain recommendation scenarios, previous works have not explored the hierarchical category structures (e.g., genre → sub-genre → detailed-category) which is usually applied to categorize items in a real-word recommender, for example, the categories on the

0162-8828 © 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on October 09,2021 at 03:35:47 UTC from IEEE Xplore. Restrictions apply.

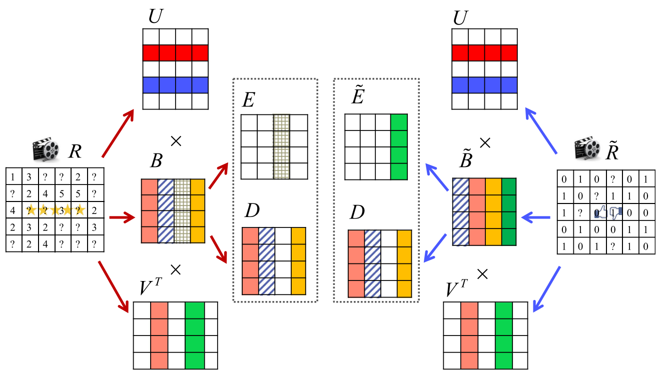


Fig. 1. Illustration of one-layer Low-rank Sparse Collective Factorization (LSCF) framework. R denotes a numerical rating (i.e., 5-star rating) matrix in the target domain. $\tilde{R}$ denotes a binary rating (i.e., like or dislike) matrix in the auxiliary domain. U and V denote shared user and item latent factors, respectively. B and $\tilde{B}$ denote rating patterns in the target and the auxiliary domain, respectively. B and $\tilde{B}$ are further decomposed into a shared common pattern D and domain-specific patterns E and $\tilde{E}$ respectively with low-rank sparse decomposition.

DVD rental page in Netflix.[1] In single domain recommendation works, the hierarchical structure has been explored recently, such as exploiting the hierarchical users and items latent factors [15], [16], and a deep version for matrix factorization [17], [18]. While these works demonstrate the effectiveness of applying hierarchical structure in single domain recommendation, very few works explore the hierarchical structure in the cross-domain recommender. Intuitively, there is also a hierarchical structure of users' preferences in the cross-domain scenario. For example, if a user likes "Horror" genre → "Foreign Horror" sub-genre → "Asian Horror"detailed-category in the movie hierarchical structure in Netflix, he or she may also like "Horror" → "Foreign Horror" → "Asian Horror" in the book hierarchical structure. In cross-domain recommendation, how to transfer the shared hierarchical knowledge remains a challenging problem.

To address the above challenges, we propose a novel Deep Low-rank Sparse Collective Factorization (DLSCF) framework for heterogeneous recommendation in this paper. We work on the scenario of transferring binary rating (e.g., like/dislike) in the auxiliary domain to numerical rating (e.g., 5-star gradings) in the target domain, which is the same as TCF [2], [8]. Although the scales of binary ratings and numerical ratings are not the same, a user's preference towards items and categories shall keep consistent. First, in single layer LSCF, target and auxiliary domain rating matrix R and $\tilde{R}$ are tri-factorized into user latent factor U, item latent factor V and rating patterns B and $\tilde{B}$ respectively, as shown in Fig. 1. Compared to (TCF) [2], [8], low-rank sparse decomposition is adopted onto B and $\tilde{B}$, to further capture the shared knowledge between them. LSCF assumes that B and $\tilde{B}$ can be decomposed to a common pattern D and domain-specific rating patterns E and $\tilde{E}$, respectively. Intuitively, users have similar preferences are likely to have similar rating patterns [11], which indicates the group characteristic in rating patterns. This group characteristic is modeled through a low-rank constraint on D and a group-sparsity constraint on E and $\tilde{E}$.

1. http://dvd.netflix.com/AllGenresList

Second, we further explore the deep/hierarchical knowledge transfer model of LSCF in cross-domain scenario, considering the real-world hierarchical recommender. We uncover the shared knowledge in hierarchical structures from learning a latent category affiliation matrix which captures the affiliation relation between latent categories and latent sub-categories. In each layer of the hierarchical structures, we obtain the user and item latent category affiliation matrices through further tri-factorizing the rating pattern matrices, as shown in Fig. 2.

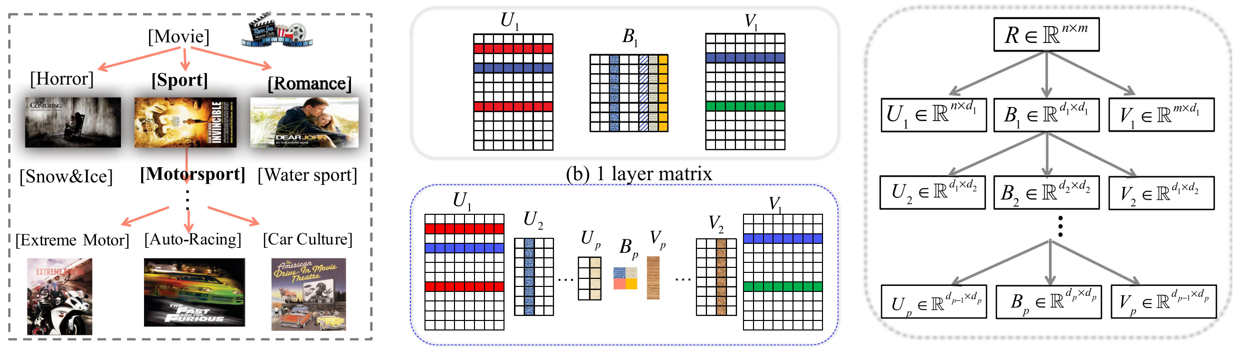
We provide both batch based and Stochastic Gradient Descent (SGD) based optimization algorithms for DLSCF, and named as DLSCF-B and DLSCF-S respectively. Batch based optimization algorithms such as PMF [19] and TCF [8] are more widely explored in this problem. They update model parameters only once after scanning the whole data. However, in real-world recommendation tasks, there are two main challenges. First, the number of users, items, and ratings are very large. Batch based methods update all the user and item factors together, which may take a lot of memory. Second, batch based methods are hard to deal with upcoming new users and items. SGD based methods are widely used in the large-scale data problem such as deep learning methods and recommendation methods, such as Regularized Singular Value Decomposition (RSVD) [20] and Collective Matrix Factorization [21]. They usually randomly update sample by sample. In this way, we address these two challenges through an SGD based optimization algorithm. Extensive experiments on five real-world benchmarks demonstrate that DLSCF with both batch and SGD based optimization algorithms outperforms the state-of-the-arts in the heterogeneous recommendation scenario of transferring binary ratings in the auxiliary domain to numerical rating in the target domain.

This paper is an extension of our previous conference work [22]. Compared to the conference version, the journal version has three major differences. First, we propose an SGD based solution of DLSCF (DLSCF-S) to reduce the memory of the computational cost compared to the batch based solution (DLSCF-B) for real-world large-scale data scenario. Compared to DLSCF-B, DLSCF-S is able to do recommendation for the continuously-arrived new users. Second, we add experiments of comparing DLSCF-S with existing SGD based collective factorization methods and DLSCF-S outperforms these methods. Third, we add three more large-scale benchmarks Flixter, ML10M and ML20M to evaluate the performances of compared methods.

2 Related Work

In this section, we describe related works of cross-domain recommender. It is usually applied to facilitate the personalized recommendation in the target domain when it is with limited data and information, from exploiting knowledge such as user preferences from auxiliary/source domains. There are mainly two types of approaches based on how to exploit knowledge from the auxiliary domain. The first one is to *aggregate knowledge* of both target and auxiliary domains. The second one is to *transfer knowledge* from the auxiliary domain to the target domain.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on October 09,2021 at 03:35:47 UTC from IEEE Xplore. Restrictions apply.


[Movie]
[Horror]
[Sport]
[Romance]
[Snow&Ice]
[Motorsport]
[Water sport]
[Extreme Motor]
[Auto-Racing]
[Car Culture]
U_1
B_1
V_1
(b) 1 layer matrix
U_1 U_2 U_p B_p V_p V_2 V_1
$R \in \mathbb{R}^{n\times m}$
$U_1 \in \mathbb{R}^{n\times d_1}$
$B_1 \in \mathbb{R}^{d_1\times d_1}$
$V_1 \in \mathbb{R}^{m\times d_1}$
$U_2 \in \mathbb{R}^{d_1\times d_2}$
$B_2 \in \mathbb{R}^{d_2\times d_2}$
$V_2 \in \mathbb{R}^{d_1\times d_2}$
$U_p \in \mathbb{R}^{d_{p-1}\times d_p}$
$B_p \in \mathbb{R}^{d_p\times d_p}$
$V_p \in \mathbb{R}^{d_{p-1}\times d_p}$

(a) hierarchical category (c) p layers matrix (d) p layers tri-factorizing

Fig. 2. Illustration of hierarchical structures with the real-world movie recommender in figure (a), 1-layer and p-layer matrices of DLSCF in figures (b) and (c), and the deep tri-factorizing process in figure (d). In figures (b), (c) and (d), we apply target domain matrix tri-factorizing as an example and the auxiliary domain tri-factorizing is similar. Figure (a) shows three movie genres "Horror", "Sport" and "Romance". Three sub-genres are under "Sport" and three detailed categories are under "Motorsport". In figure (b), B_1, U_1, and V_1 are the rating matrix, user and item latent factors using one layer tri-factorizing towards rating matrix R. p layer matrices $U_1 \dots U_p$, $V_1 \dots V_p$ and B_p are shown in figure (c). Figure (d) shows the tri-factorizing process. From 1 to $p-1$ layer, we further tri-factorize B_i into U_{i+1}, B_{i+1} and V_{i+1}. $d_1 \geq d_2 \dots \geq \dots \geq \dots d_p$.

Knowledge aggregating methods can generally be categorized into three strategies: (1) merging user preferences [23] from user's review, rating, click-through data etc.; (2) mediating user modeling data such as neighborhoods and similarities [24]; and (3) combining recommendations such as rating probability distributions and estimations in both domains [25].

The knowledge transfer methods can also be categorized into three strategies based on which information or knowledge to share or transfer: (1) sharing common information of two domains, which includes inter-domain correlations, semantic networks, item attributes, etc. [26], [27]; (2) sharing the user or item latent features [2], [8], [28], [29], [30], [31]; (3) transferring rating patterns [11], [12], [13], [14].

Among knowledge transfer methods, Li et al. captured the common rating patterns of user ratings in two domains from the latent feature space [9], [10]. Geo et al. further enhanced the common rating pattern learning by the latent feature model based on a user group cluster [11], because they thought that specific and diverse features among each domains may degrade the effectiveness of the common rating pattern. However, the above works pay little attention on the shared latent features of users and items [32], [33]. In cross-domain recommender, especially transferring knowledge in heterogeneous domains (e.g., binary ratings to numerical ratings), it remains a problem to effectively transfer both rating patterns and the latent features of users and items.

To exploit the knowledge in heterogeneous domains, Pan et al. explored regularized matrix factorization models to transfer the latent features from auxiliary domain to target domain [2], [8], [31]. Especially, for the scenario of transferring knowledge from binary ratings to numerical ratings, Pan et al. proposed a Transfer by Collective Factorization framework.

TCF is most related to our work. However, there are two major differences. First, TCF learns domain-independent rating patterns of each domain, while we uncover more shared common rating patterns from auxiliary domain via low-rank sparse decomposition. The second difference is the hierarchical structure of our model, which is enlighten from the hierarchical category structure in real-word recommender.

3 The Proposed Algorithm

In this section, we first describe the problem definition of heterogeneous recommendation in this paper. Second, we describe the preliminary knowledge. Third, we present our single-layer model Low-rank Sparse Collective Factorization (LSCF). Fourth, we extend the single-layer model to multi-layer Deep Low-rank Sparse Collective Factorization, which is enlighten from the hierarchical structure of the real-world recommendation system.

3.1 Problem Definition

Assume that we have a 5-star numerical rating matrix $R = [r^{(u,v)}]_{n\times m} \in \{1,2,3,4,5,?\}^{n\times m}$ in the target domain and a like or dislike binary rating matrix $\tilde{R} = [\tilde{r}^{(u,v)}]_{n\times m} \in \{0, 1, ?\}^{n\times m}$ in the auxiliary domain. n is number of users and m is number of items. In the target domain, each element at location (u, v) of R denotes a 5-star rating of item v rated by user u. In the auxiliary domain, each element denotes a binary rating. The rating 1 presents that the user likes the item and 0 presents dislike. The question mark "?" denotes the unobserved or missing value, which is in $\{1,2,3,4,5\}$ in R and in $\{0,1\}$ in $\tilde{R}$. $Y = [y^{(u,v)}]_{n\times m} \in \{0,1\}^{n\times m}$ and $\tilde{Y} = [\tilde{y}^{(u,v)}]_{n\times m} \in \{0,1\}^{n\times m}$ are mask matrices in the target domain and the auxiliary domain respectively, where 1 presents that the rating is observed and 0 presents not. Usually, the proportion of missing value in the target domain is larger or equivalent to the auxiliary domain. The goal of work is to predict the numerical missing values in R in the target with the help of transferring the knowledge of binary rating in $\tilde{R}$ in the auxiliary domain. We would like to note that the proposed method in this paper can also be applied to predict the missing values in $\tilde{R}$. However it is out of the scope of this paper.

3.2 Preliminary

PMF. Probabilistic Matrix Factorization (PMF) [19] focuses on single domain recommendation. It aims to uncover the missing values in rating matrix R, which could be either numerical or binary rating matrix. PMF factorizes R into a user latent factor matrix and an item latent factor matrix,

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on October 09,2021 at 03:35:47 UTC from IEEE Xplore. Restrictions apply.

$$R \sim UV^\top, \tag{1}$$

where $U \in \mathbb{R}^{n\times d}$ denotes user latent factor matrix and $V \in \mathbb{R}^{m\times d}$ denotes item latent factor matrix. n and m are the number of users and items respectively. Assuming that $U_{(u\cdot)} \in \mathbb{R}^{1\times d}$ denotes user u's latent factor and $V_{(v\cdot)} \in \mathbb{R}^{1\times d}$ denotes item v's latent factor, the rating located at (u, v) could be recovered via $\hat{r}_{uv} = U_{(u\cdot)}V_{(v\cdot)}^\top$. However, PMF does not deal with the cross-domain recommendation problem, and ignores to use of auxiliary data.

CMF. Collective Matrix Factorization [21] focuses on cross-domain recommendation. CMF factorizes both target and auxiliary domain rating matrix R and $\tilde{R}$ into user and item latent factor matrix. CMF transfers the knowledge in $\tilde{R}$ to facilitate the rating prediction in R from assuming that the two domains share the same item latent factor.

$$R \sim UV^\top, \tilde{R} \sim WV^\top, \tag{2}$$

where $U, W \in \mathbb{R}^{n\times d}$ denote the user latent factor matrix in the target and auxiliary domain respectively. $V \in \mathbb{R}^{m\times d}$ denotes the shared item latent factor matrix in both domains. However, CMF may neglect the share information in user-side auxiliary data.

TCF. Different from PMF and CMF, TCF [2], [8] adopts matrix tri-factorization model and decomposes the rating matrix into user and item latent factors and ratting patterns. TCF assumes that target and auxiliary domain rating matrix R and $\tilde{R}$ share the same user and latent factors U and V and domain specific rating patterns B and $\tilde{B}$. The objective function of TCF is as following:

$$\begin{aligned} \min_{U,V,B,\tilde{B}} \frac{1}{2}\|Y \odot (R - UBV^\top)\|_F^2 + \frac{\lambda}{2}\|\tilde{Y} \odot (\tilde{R} - U\tilde{B}V^\top)\|_F^2 \\ \text{s.t.} \quad U^\top U = \mathrm{I},\ V^\top V = \mathrm{I}, \end{aligned} \tag{3}$$

where Y and $\tilde{Y}$ indicate whether a rating is observed or not, which is described in "Problem Definition". $\|\cdot\|_F$ denotes the Frobenius norm. I denotes the identity matrix. $\odot$ denotes the dot production. However, as pointed by [9], [10], [11], B and $\tilde{B}$ should share some common parts since two domains are related. Thus, model (3) may fail to fully use the shared knowledge in two rating patterns.

3.3 Low-Rank Sparse Collective Factorization (LSCF)

Here we describe single-layer model Low-rank Sparse Collective Factorization. Our motivation is to effectively capture the shared knowledge in the auxiliary domain rating matrix. Different from TCF which learns domain specific rating patterns in each domain, LSCF assumes that related domains share a common rating pattern and preserve domain-specific pattern for each domain. Intuitively, the rating patterns of one user group, where users are with similar preference towards items or categories, should be similar [11]. Geo et al. modeled the group similar behavior from cluster-level based user and item latent factor. Meanwhile, they a concatenated a common rating patten matrix and a domain specific rating pattern matrix as a whole rating pattern matrix for each domain. In LSCF, we model the group similar behavior via low-rank sparse decomposition [34], [35], [36]. As shown in Fig. 1, after tri-factorizing the rating matrix into shared user latent factor U, item latent factor V and rating patterns B and $\tilde{B}$ for each domain, we further decompose B and $\tilde{B}$ into a common rating pattern D with a low-rank constraint which drives the rating patterns of same user group similar, and column sparse domain-specific patterns E and $\tilde{E}$ [37].

To this end, the objective function f is proposed as

$$\begin{aligned} f = &\frac{1}{2}\|Y \odot (R - UBV^\top)\|_F^2 + \beta_1(\|E\|_{2,1} + \|\tilde{E}\|_{2,1}) \\ &+ \frac{\lambda}{2}\|\tilde{Y} \odot (\tilde{R} - U\tilde{B}V^\top)\|_F^2 + \mathrm{rank}(D) + \\ &\frac{\beta_2}{2}(\|B - D - E\|_F^2 + \|\tilde{B} - D - \tilde{E}\|_F^2) \\ &\text{s.t. } U^\top U = \mathrm{I}, V^\top V = \mathrm{I}, \end{aligned} \tag{4}$$

where D is with a low-rank constraint. $\mathrm{rank}(\cdot)$ denotes the rank operator of a matrix. The column-sparse matrices E and $\tilde{E}$ are constrained with $l_{2,1}$.

Compared to TCF-based methods [2], [8] learn domain-specific rating pattern in each domain, LSCF applies low-rank sparse decomposition to decompose the rating pattern into a common and a domain-specific rating pattern. In this way, LSCF transfers more common knowledge in the auxiliary domain.

3.4 Deep Low-Rank Sparse Collective Factorization (DLSCF)

Real-world recommender is usually with a hierarchical structure. Fig. 2a shows an example of hierarchical category for movie recommendation system in Netflix DVD rental page. Different layers of the tree structure represent genre, sub-genre to detailed-category. There are three genres in the first layer: Horror, Romance and Sports & Fitness. In the second layer, there are three sub-genres under Sport: Snow & Ice, Motorsport, and Water Sports. Under Motorsport, there are three detailed categories: Auto Racing, Car Culture and Extreme Motorsport. A user's preference can also be hierarchical. For example, a user may prefer more to Sport genre than Horror genre. Under Sport genre, he or she may prefer more on Snow & Ice than Motorsport. The hierarchical structure has also been explored in other real-word applications such as image recognition and classification [38], [39].

Recently, a few works have learned hierarchical user or item latent factors to capture the multi-layer knowledge [15], [16], [40]. For example, Wang et al. factorized U into hierarchical user latent factors as $U \approx U_1U_2\cdots U_i\cdots U_p$, and factorized V into hierarchical item latent factors as $V \approx V_1V_2\cdots V_i\cdots V_p$, where $i \in \{1, 2, \ldots, p\}$. To this end, they turned $R \approx UV^\top$ into

$$R \approx U_1U_2\cdots U_i\cdots U_pV_p^\top\cdots V_i^\top\cdots V_1^\top. \tag{5}$$

where U_i and V_i ($i \in \{1, 2, \ldots, p\}$) are non-negative.

However, these existing works only focus on single domain recommendation. In cross-domain recommendation scenario, only factorizing U and V as hierarchical latent factors as model (5) [15] may not work. There are two main reasons. First, we tri-factorize $R \approx UBV^\top$ and constrain U and V using $UU^\top = \mathrm{I}, VV^\top = \mathrm{I}$ instead of constraining U and V to be nonnegative, which follows model TCF [2], [8], [31] in Eq. (3). Second, as our motivation is to uncover

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on October 09,2021 at 03:35:47 UTC from IEEE Xplore. Restrictions apply.

common rating patterns as well as common user and item latent factors, we also need to uncover the hierarchical common rating patterns. To address these challenges, we extend the one layer LSCF to multi-layer model and named as Deep Low-rank Sparse Collective Factorization.

We first introduce how to capture the hierarchical structures through progressively tri-factorizing the rating matrix. We take target domain as an example first. In the first layer, as shown in Figs. 2b and 2d, assuming that there are n users, m items and d_1 categories, we tri-factorize $R \in \mathbb{R}^{n\times m}$ to three matrices: $R \approx U_1 B_1 V_1^\top$. $U_1 \in \mathbb{R}^{n\times d_1}$ and $V_1 \in \mathbb{R}^{m\times d_1}$ are the user and item latent factors. $B_1 \in \mathbb{R}^{d_1\times d_1}$ is the rating matrix. We would like to note that in the first layer we want to learn user's preference towards all the detailed categories, and thus, the number of layer size of the first layer is the largest among all the layers.

In the second layer, assuming that there are d_2 categories, we tri-factorize B_1 to $U_2 B_2 V_2^\top$ and keep U_1 and V_1 unchanged, where $B_2 \in \mathbb{R}^{d_2\times d_2}$ and $U_2, V_2 \in \mathbb{R}^{d_1\times d_2}$. U_2 and V_2 are the user and item latent category affiliation matrix. They capture the affiliation relation between categories in the first and second layer on user and item side respectively. Usually the categories of the second layer are more general than the categories of the first layer and we have $d_1 > d_2$. In the second layer, we factorize R as

$$R \approx U_1 U_2 B_2 V_2^\top V_1^\top. \tag{6}$$

In the p-layer, as shown in Figs. 2c and 2d, assuming that there are d_p categories in the pth layer and d_{p-1} categories in the $(p-1)$th layer, we further tri-factorize B_{p-1} to $U_p B_p V_p^\top$, where $B_p \in \mathbb{R}^{d_p\times d_p}$ and $U_p, V_p \in \mathbb{R}^{d_{p-1}\times d_p}$. In the p-layer, we factorize R as following:

$$R = U_1 U_2 \cdots U_p B_p V_p^\top \cdots V_2^\top V_1^\top. \tag{7}$$

In the auxiliary domain, similarly, we tri-factorize $\tilde{B}_{i-1}$ as $U_i \tilde{B}_i V_i^\top$ layer by layer. In the pth layer, $\tilde{R}$ is progressively factorized as

$$\tilde{R} = U_1 U_2 \cdots U_p \tilde{B}_p V_p^\top \cdots V_2^\top V_1^\top. \tag{8}$$

Assuming that there is a p-layer hierarchical structure of the recommender, the objective function of DLSCF is formulated as

$$\begin{aligned} f_p = &\frac{1}{2}\|Y \odot (R - U_1 \cdots U_p B_p V_p^\top \cdots V_1^\top)\|_F^2 \\ &+ \frac{\lambda}{2}\|\tilde{Y} \odot (\tilde{R} - U_1 \cdots U_p \tilde{B}_p V_p^\top \cdots V_1^\top)\|_F^2 \\ &+ \frac{\beta_2}{2}(\|B_p - D - E\|_F^2 + \|\tilde{B}_p - D - \tilde{E}\|_F^2) \\ &+ \mathrm{rank}(D) + \beta_1(\|E\|_{2,1} + \|\tilde{E}\|_{2,1}) \\ \text{s.t. } & U_i^\top U_i = \mathrm{I},\ V_i^\top V_i = \mathrm{I}, i \in \{1, \ldots, p\}, \end{aligned} \tag{9}$$

where U_1 and V_1 are latent user and item factors. U_i and V_i $i \in \{2, 3, \ldots, p\}$ are the latent category affiliation matrices. We assume that two domains share the same user and item latent factors and the latent category affiliation matrices. B_p and $\tilde{B}_p$ are pth ratting patterns of the target and auxiliary domain. B_p and $\tilde{B}_p$ are divided into a common rating pattern D with the low-rank constraint and domain-specific rating patterns E and $\tilde{E}$ with the group-sparsity constraint.

4 Batch Based Solution (DLSCF-B)

In this section, we introduce batch based solution for DLSCF and named as DLSCF-B. Batch based solutions update model parameters only once after scanning the whole data. DLSCF-B iteratively updates two parts. The first part is to learn the shared latent user and item factors U_1 and V_1 and the latent category affiliation matrices U_i and V_i $i \in \{2, 3, \ldots, p\}$. The second part is to learn target and auxiliary domain rating patterns B_p and $\tilde{B}_p$ with the low-rank and sparse constraint.

We optimize the variables one by one, which means we optimize one variable while fixing all the others, since it is not easy to update all the variables jointly [35], [36].

Updating U_i. We learn U_i and V_i following the similar strategy as [8]. We first remove the constraints irrelevant to U_i in Eq. (9), and rewrite Eq. (9) as

$$f_p = \frac{1}{2}\|Y \odot (R - A_i U_i H_i)\|_F^2 + \frac{\lambda}{2}\|\tilde{Y} \odot (\tilde{R} - A_i U_i \tilde{H}_i)\|_F^2, \tag{10}$$

where we define A_i, H_i and $\tilde{H}_i$, $1 \le i \le p$ as following:

$$A_i = \begin{cases} U_1 U_2 \cdots U_{i-1}, & \text{if } i \neq 1, \\ \mathrm{I}, & \text{if } i = 1. \end{cases} \tag{11}$$

And

$$H_i = \begin{cases} U_{i+1} \cdots U_p B_p V_p^\top \cdots V_1^\top, & \text{if } i \neq p, \\ B_p V_p^\top \cdots V_1^\top, & \text{if } i = p. \end{cases} \tag{12}$$

And

$$\tilde{H}_i = \begin{cases} U_{i+1} \cdots U_p \tilde{B}_p V_p^\top \cdots V_1^\top, & \text{if } i \neq p, \\ \tilde{B}_p V_p^\top \cdots V_1^\top, & \text{if } i = p. \end{cases} \tag{13}$$

Then we have

$$\begin{aligned} \frac{\partial f_p}{\partial U_i} = &A_i^\top (Y \odot (A_i U_i H_i - R)) H_i^\top \\ &+ \lambda A_i^\top (\tilde{Y} \odot (A_i U_i \tilde{H}_i - \tilde{R})) \tilde{H}_i^\top. \end{aligned} \tag{14}$$

We learn U_i using a gradient descent algorithm on the Grassmann manifold,

$$U_i \leftarrow U_i - \gamma(\mathrm{I} - U_i U_i^\top) \frac{\partial f_p}{\partial U_i} = U_i - \gamma_{U_i} \nabla_{U_i}, \tag{15}$$

where the learning rate γ_{U_i} is defined as $\gamma_{U_i} = \frac{-\mathrm{tr}(t_1^\top t_2) - \lambda \mathrm{tr}(\tilde{t}_1^\top \tilde{t}_2)}{\mathrm{tr}(t_2^\top t_2) + \lambda \mathrm{tr}(\tilde{t}_2^\top \tilde{t}_2)}$, in which $t_1 = Y \odot (R - A_i U_i H_i)$, $\tilde{t}_1 = \tilde{Y} \odot (\tilde{R} - A_i U_i \tilde{H}_i)$, $t_2 = Y \odot (A_i \nabla_{U_i} H_i)$, $\tilde{t}_2 = \tilde{Y} \odot (A_i \nabla_{U_i} \tilde{H}_i)$, and $\mathrm{tr}(\cdot)$ denotes the trace operator of a matrix.

Updating V_i. We update V_i using the similar rule as U_i. After removing terms which do not contain V_i, we rewrite Eq. (9) as

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on October 09,2021 at 03:35:47 UTC from IEEE Xplore. Restrictions apply.

$$f_p = \frac{1}{2}\|Y \odot (R - C_i V_i F_i)\|_F^2 + \frac{\lambda}{2}\|\tilde{Y} \odot (\tilde{R} - \tilde{C}_i V_i F_i)|_F^2, \quad (16)$$

where F_i, C_i and $\tilde{C}_i$ $(1 \leq i \leq p)$ are defined as following:

$$F_i = \begin{cases} V_{i-1}V_{i-2}\cdots V_1, & \text{if } i \neq 1, \\ \mathrm{I}, & \text{if } i = 1. \end{cases} \quad (17)$$

And

$$C_i = \begin{cases} U_1 \cdots U_p B_p V_p^\top \cdots V_{i+1}^\top, & \text{if } i \neq p, \\ U_i \cdots U_p B_p, & \text{if } i = p. \end{cases} \quad (18)$$

And

$$\tilde{C}_i = \begin{cases} U_1 \cdots U_p \tilde{B}_p V_p^\top \cdots V_{i+1}^\top, & \text{if } i \neq p, \\ U_i \cdots U_p \tilde{B}_p, & \text{if } i = p. \end{cases} \quad (19)$$

Then we have

$$\begin{aligned}\frac{\partial f_p}{\partial V_i} &= C_i^\top (Y \odot (C_i V_i F_i - R)) F_i^\top \\ &+ \lambda C_i^\top (\tilde{Y} \odot (C_i V_i \tilde{F}_i - \tilde{R})) \tilde{F}_i^\top.\end{aligned} \quad (20)$$

Similar as U_i, we then learn V_i via the gradient descent algorithm on the Grassmann manifold,

$$V_i \leftarrow V_i - \gamma(\mathrm{I} - V_i V_i^\top)\frac{\partial f_p}{\partial V_i} = V_i - \gamma_{V_i}\nabla_{V_i}, \quad (21)$$

where we learn γ_{V_i} in the same way as γ_{U_i}.

Updating B_p and $\tilde{B}_p$. We only introduce rule for updating B_p here, since $\tilde{B}_p$ can be updated in a similar way. We first remove the parts irrelevant to B_p and rewrite the objective function as

$$\min_{B_p} \frac{1}{2}\|Y \odot (R - U B_p V^\top)\|_F^2 + \frac{\beta_2}{2}\|B_p - D - E\|_F^2, \quad (22)$$

where $U = U_1 \cdots U_p$ and $V = V_1 \cdots V_p$. We consider it as a corresponding least square SVM problem to estimate B_p

$$\begin{aligned}\varrho &= \arg\min_{\varrho} \frac{1}{2}\|r - \Theta\varrho\|_2^2 + \frac{\beta_2}{2}\|\varrho - \rho\|_2^2 \\ &= (\Theta^\top\Theta + \beta\mathrm{I})^{-1}(\Theta^\top r + \beta\rho),\end{aligned} \quad (23)$$

where $\rho = \mathrm{vec}(D + E) = \mathrm{vec}(\bar{D}) = [\bar{D}_1^\top, \ldots, \bar{D}_d^\top]^\top \in \mathbb{R}^{d^2 \times 1}$. I denotes the identity matrix. We concatenate all the columns of matrix B_p as $\varrho = \mathrm{vec}(B_p) = [B_{1,1}^\top, \ldots, B_{1,d}^\top]^\top \in \mathbb{R}^{d^2 \times 1}$. For observed rating (i.e., $y_{u,v} = 1$), we construct the instances as $\{(\theta_{u,v}, r_{u,v})\}$, where $\theta_{u,v}$ is the value at location (u, v) of Θ and $\theta_{u,v} = \mathrm{vec}(U^{(u\cdot)^\top} V^{(v\cdot)}) \in \mathbb{R}^{d^2 \times 1}$. Then, we can consider this problem as a linear compact operator and solve B_p or ρ using existing off-the-shelf tools.

Updating D. The objective for updating D can be rewritten as following after removing the parts irrelevant to D:

$$\begin{aligned}D &= \arg\min_D \|D\|_* \\ &+ \frac{\beta_2}{2}(\|B_p - D - E\|_F^2 + \|\tilde{B}_p - D - \tilde{E}\|_F^2) \\ &= \arg\min_D \frac{1}{\beta_2}\|D\|_* + \frac{1}{2}(\|D - \frac{(B_p - E + \tilde{B}_p - \tilde{E})}{2}\|_F^2),\end{aligned} \quad (24)$$

where minimizing the rank problem is replaced with nuclear norm minimization [34], [35], [36]. In this way, we can apply the singular value thresholding (SVT) operator [41] to solve the problem.

Updating E and $\tilde{E}$. Similarly, the objective for updating D can be rewritten after removing the parts irrelevant to E

$$\begin{aligned}E &= \arg\min_E \beta_1\|E\|_{2,1} + \frac{\beta_2}{2}(\|B_p - D - E\|_F^2) \\ &= \arg\min_E \frac{\beta_1}{\beta_2}\|E\|_{2,1} + \frac{1}{2}(\|E - (B_p - D)\|_F^2).\end{aligned} \quad (25)$$

We solve the problem by the shrinkage operator [42]. $\tilde{E}$ can be solved in the same way as E.

To this end, we optimize each variable iteratively until convergence for DLSCF-B.

4.1 Complexity Analysis of Batch Based Solution

In the algorithm of solving DLSCF, there are three main time-consuming steps: (1) updating U_i and V_i $(i = 1, \ldots, p)$ in each layer, (2) updating rating pattern B_p and $\tilde{B}_p$, (3) updating shared rating pattern D.

The total time complexity of the first two steps is $O\big(K(\max(q, \tilde{q})d_1^3 + \sum_{i=2}^p d_{i-1}d_i + d_p^6)\big)$. It is added up with time cost of updating U_i and V_i, $i \in \{1, \ldots, p\}$ in each layer, since our model is with the deep structure. d_i denotes the number of latent factors. n and m denote the number of users and items respectively. q and $\tilde{q}(q, \tilde{q} > n, m)$ denote the number of observed ratings in target and auxiliary domain respectively. K denotes the number of iterations for converge. The time complexity of one layer model LSCF is similar as [8]. The third step costs $O(Kd_p^3)$ for K iterations, mainly for full SVD operator for nuclear norm [34], [36], when $D \in \mathbb{R}^{d_p \times d_p}$. The total time complexity of DLSCF is about $O\big(K(\max(q, \tilde{q})d_1^3 + \sum_{i=2}^p d_{i-1}d_i + d_p^6 + d_p^3)\big)$ when adding the cost of three steps together. The time complexity of low-rank part is much lower than the first two parts, and meanwhile, the low-rank constraint shows effectiveness in the prediction accuracy in the experiments.

We also analyze the time complexity of comparison methods PMF, cPMF and TCF. PMF costs $O(Kqd^2 + K\max(n, m)d^3)$. cPMF costs $O(Kq\tilde{c}d^2 + K\max(n, m)d^3)$ [19]. TCF costs $O(K\max(q, \tilde{q})d^3 + Kd^6)$ [8]. DLSCF's time complexity is comparable to TCF. We would like to note that the very deep hierarchical structure could cause the high time complexity, however, according to our observation and analysis from experiments, the two-layer model would be enough to achieve the state-of-the-art performance. It demonstrates that our model would not cause high time complexity in practice.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on October 09,2021 at 03:35:47 UTC from IEEE Xplore. Restrictions apply.

5 Stochastic Gradient Descent (SGD) Based Solution (DLSCF-S)

Batch based algorithms such as PMF [19] and TCF [8] update model parameters only once after scanning the whole data. In real world large data scenario, it might not be applicable for two reasons. First, updating all the user and item factors may take a lot of memory if the number of users and items are very large. Second, batch based methods are hard to deal with upcoming new users and items.

Although some stochastic based methods achieve efficient performance than their alternative batch based algorithms, such as Regularized Singular Value Decomposition [20] and Collective Matrix Factorization [21]. However, the prediction accuracy of these methods are less accurate than batch based ones. For homogeneous recommendation, there are some stochastic gradient descent based methods achieved high accuracy, such as distributed stochastic gradient descent [43] and online multi-task collaborative filtering [44]. However, few works focused on heterogeneous as our scenario.

To address these challenges, in this section, we introduce the Stochastic Gradient Descent based optimization algorithm and name it as (DLSCF-S).

First, we obtain the following equivalent minimization problem for DLSCF as Eq. (9). Here we make the problem easier, since generally only U_1 and V_1 could be huge matrices when R and $\tilde{R}$ are large-scale. Thus, we only update U_1 and V_1 in a SGD way. And further, we release the orthogonal constraints on U_1 and V_1 to an l_2 regularizer on them.

$$\begin{aligned}\min_{\Phi} \sum_{u=1}^{n}\sum_{v=1}^{m} y^{(uv)}\bigg[&\frac{1}{2}\|r^{(uv)} - U_1^{(u\cdot)}\cdots U_p^{(u\cdot)}B_pV_p^{(v\cdot)\top}\cdots V_1^{(v\cdot)\top}\|_F^2 \\ &+\frac{\alpha_u}{2}\|U_1^{(u\cdot)}\|^2+\frac{\alpha_v}{2}\|V_1^{(v\cdot)}\|^2\bigg] \\ +\lambda\sum_{u=1}^{n}\sum_{v=1}^{m}\tilde{y}^{(uv)}\bigg[&\frac{1}{2}\|\tilde{r}^{(uv)} - U_1^{(u\cdot)}\cdots U_p^{(u\cdot)}\tilde{B}_pV_p^{(v\cdot)\top}\cdots V_1^{(v\cdot)\top}\|_F^2 \\ &+\frac{\alpha_u}{2}\|U_1^{(u\cdot)}\|^2+\frac{\alpha_v}{2}\|V_1^{(v\cdot)}\|^2\bigg] \\ &+\frac{\beta_2}{2}(\|B_p-D-E\|_F^2+\|\tilde{B}_p-D-\tilde{E}\|_F^2) \\ &+\operatorname{rank}(D)+\beta_1(\|E\|_{2,1}+\|\tilde{E}\|_{2,1}) \\ &\text{s.t. } U_i^\top U_i=\mathrm{I},\ V_i^\top V_i=\mathrm{I}, i\in\{2,\ldots,p\} \\ &\Phi=\{U_i,V_i,i\in\{1,\ldots,p\},B_p,\tilde{B}_p,D,E,\tilde{E}\}.\end{aligned} \tag{26}$$

Note that we do not directly apply a stochastic update rules on Eq. (9) because of the orthonormal constraints on user and latent feature matrices in the adopted matrix tri-factorization model.

In Eq. (26) we see that only the size of U_1 and V_1 is effected by the number of users and items and could be huge. The sizes of U_i, V_i, $i\in\{2,\ldots,p\}$, B_p, $\tilde{B}_p$, D, E, $\tilde{E}$ are predefined and could not be huge. Thus, in our algorithm, we only update U_1 and V_1 in the SGD way, and update other items in the same way as batch based solution.

Updating U_1. Define $f_u=\sum_{v=1}^m y^{(uv)}[\frac{1}{2}\|r^{(uv)}-U_1^{(u\cdot)}\cdots U_p^{(u\cdot)}B_pV_p^{(v\cdot)\top}\cdots V_1^{(v\cdot)\top}\|_F^2+\frac{\alpha_u}{2}\|U_1^{(u\cdot)}\|^2+\frac{\alpha_v}{2}\|V_1^{(v\cdot)}\|^2]+\sum_{v=1}^m\tilde{y}^{(uv)}[\frac{\lambda}{2}\|\tilde{r}^{(uv)}-U_1^{(u\cdot)}\cdots U_p^{(u\cdot)}\tilde{B}_pV_p^{(v\cdot)\top}\cdots V_1^{(v\cdot)\top}\|_F^2+\frac{\alpha_u}{2}\|U_1^{(u\cdot)}\|^2+\frac{\alpha_v}{2}\|V_1^{(v\cdot)}\|^2]+\frac{\beta_2}{2}(\|B_p-D-E\|_F^2+\|\tilde{B}_p-D-\tilde{E}\|_F^2)+\operatorname{rank}(D)+\beta_1(\|E\|_{2,1}+\|\tilde{E}\|_{2,1})$.

By defining $U_2^{(u\cdot)}\cdots U_p^{(u\cdot)}B_pV_p^{(v\cdot)\top}\cdots V_1^{(v\cdot)\top}=A$ and $U_2^{(u\cdot)}\cdots U_p^{(u\cdot)}\tilde{B}_pV_p^{(v\cdot)\top}\cdots V_1^{(v\cdot)\top}=\tilde{A}$, we have

$$\begin{aligned}\frac{\partial f_u}{\partial U_1^{(u\cdot)}} =& \sum_{v=1}^m y^{(uv)}\Big[(-r^{(uv)}+U_1^{(u\cdot)}\cdot A)A^\top+\alpha_uU_1^{(u\cdot)}\Big] \\ &+\sum_{v=1}^m\tilde{y}^{(uv)}\Big[(-\tilde{r}^{(uv)}+U_1^{(u\cdot)}\cdot A)A^\top+\alpha_uU_1^{(u\cdot)}\Big] \\ =&-\sum_{v=1}^m\Big(y^{(uv)}r^{(uv)}A^\top+\lambda\tilde{y}^{(uv)}\tilde{r}^{(uv)}\tilde{A}^\top\Big) \\ &+\alpha_uU_1^{(u\cdot)}\sum_{v=1}^m\Big(y^{(uv)}+\lambda\tilde{y}^{(uv)}\Big) \\ &+U_1^{(u\cdot)}\sum_{v=1}^m\Big(y^{(uv)}AA^\top+\lambda\tilde{y}^{(uv)}\tilde{A}\tilde{A}^\top\Big).\end{aligned} \tag{27}$$

Setting $\frac{\partial f_u}{\partial U_1^{(u\cdot)}}=0$, we have the update rule for each $U_1^{(u\cdot)}$,

$$U_1^{(u\cdot)}=b^{(u)}(G^{(u)})^{-1}, \tag{28}$$

where $G^{(u)}=\sum_{v=1}^m(y^{(uv)}AA^\top+\lambda\tilde{y}^{(uv)}\tilde{A}\tilde{A}^\top)+\alpha^{(u)}\sum_{v=1}^m(y^{(uv)}+\lambda\tilde{y}^{(uv)})\mathrm{I}$, and $b^{(u)}=\sum_{v=1}^m(y^{(uv)}r^{(uv)}A^\top+\lambda\tilde{y}^{(uv)}\tilde{r}^{(uv)}\tilde{A}^\top)$. Given B and V, $U_1^{(u\cdot)}$ in Eq. (28) is independent of all other users' latent features, thus we can obtain U_1 analytically.

Updating V_1. Similarly, given B, $U_1,\ldots,U_p$, $V_2,\ldots,V_p$ the latent factor $V_1^{(v\cdot)}$ of the vth item can be estimated in a closed form. Let $U_1^{(u\cdot)}\cdots U_p^{(u\cdot)}B_pV_p^{(v\cdot)\top}\cdots V_2^{(v\cdot)\top}=A$ and let $U_1^{(u\cdot)}\cdots U_p^{(u\cdot)}\tilde{B}_pV_p^{(v\cdot)\top}\cdots V_2^{(v\cdot)\top}=\tilde{A}$, and thus we can obtain the item-specific latent factor V_1 analytically.

$$V_1^{(v\cdot)}=b^{(v)}(G^{(v)})^{-1}, \tag{29}$$

where $G^{(v)}=\sum_{u=1}^n(y^{(uv)}A^\top A+\lambda\tilde{y}^{(uv)}\tilde{A}^\top\tilde{A})+\alpha_v\sum_{u=1}^n(y^{(uv)}+\lambda\tilde{y}^{(uv)})\mathrm{I}$, and $b^{(v)}=\sum_{u=1}^n(y^{(uv)}r^{(uv)}A+\lambda\tilde{y}^{(uv)}\tilde{r}^{(uv)}\tilde{A})$.

To this end, we optimize each variable iteratively until convergence for DLSCF-S. First, different from DLSCF-B which initializes U_1 and V_1 by factorizing rating matrix by SVD, DLSCF-S initializes U_1 and V_1 randomly. In this way, we can avoid SVD operation on the large matrix. Meanwhile, for upcoming new users or items, we just need to initialize its latent factors and update the parameters. There is no need to form a new rating matrix and re-train the model from the beginning. The initialization of other parameters is the same as DLSCF-B. After initialization, since there are more than one variables to be optimized, we optimize one variable while fixing others. For each iteration, instead of updating U_1 or V_1 as a whole matrix, we only update one row of $U_1^{(u\cdot)}$ or $V_1^{(v\cdot)}$ each time. Other parts of the DLSCF-S are the same as DLSCF-B.

6 Experiment

In this section, we first describe datasets, evaluation metrics and comparison methods. Then we present the evaluation

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on October 09,2021 at 03:35:47 UTC from IEEE Xplore. Restrictions apply.

TABLE 1
Previous Usage (i.e., for Evaluating Batch or/and SGD Based Methods) and Statistic of Datasets Netflix, MoviePilot, Flixter, MovieLens10M and MovieLens20M

	previous usage	#user	#item	# target rating	target sparsity	#aux rating	aux sparsity	#test rating	test sparsity
Netflix	both	5,000	5,000	45,000	$\le 1\%$	500,000	2%	2,824,204	11.27%
MoviePilot	Batch	2,000	2,000	30,000	$\le 1\%$	101,572	2%	454,710	11.37%
Flixter	SGD	147,612	48,794	3,278,431	0.046%	3,278,431	0.046%	1,639,215	0.023%
MovieLens10M	SGD	71,567	10,681	4,000,022	0.52%	4,000,022	0.52%	2,000,010	0.26%
MovieLens20M	/	138,493	26,744	8,000,106	0.22%	8,000,105	0.22%	4,000,219	0.11%

We show the number of users (#user), items (#item) and ratings on target (# target rating), auxiliary(#aux rating), and test data(#test rating). We also show the sparsity of ratings on target (target sparsity), auxiliary(aux sparsity), and test data(test sparsity). The sparsity means the average rated items by user dividing the number of all the items.

results on both accuracy and memory aspect. We also evaluate the influence of parameters.

6.1 Datasets and Settings

We conduct experiments on five benchmarks for recommendation collected from real-world, (1) Moviepilot, (2) Netflix, (3) Flixter, (4) MovieLens10M and (5) MovieLens20M. Since there is no existing dataset designed for the same scenario as ours (i.e., one numerical domain and one binary domain), we follow existing works [8], [45] on same scenario for the experimental setting. We describe the way to generate target and auxiliary domain data in Moviepilot in detail, and describe other four datasets briefly as they are generated in the similar way.

The Moviepilot rating data contains more than 4.5×10^6 observed ratings, given by more than 1.0×10^5 users on around 2.5×10^4 movies. We first randomly extract a dense rating matrix R with size $2,000 \times 2,000$ from observed ratings, given by 2,000 users towards 2,000 items. We then split R into training set T_R and test set T_E randomly by 50 percent. T_R, $T_E \subset (u, v, r_{u,v}) \in N \times N \times [1,5] | 1 \le u \le n, 1 \le v \le m$. To evaluate the effectiveness of solving sparsity problem, we further randomly sample ratings from T_R for training with sparsity $(u, v, y_{u,v}, y_{u,v}/n/m)$ levels of 0.2, 0.4, 0.6 and 0.8 percent correspondingly. The sparsity means the average rated items by user dividing the number of all the items. For example, the sparsity 0.2 percent means if there are 1000 items, one user has only rated 2 items by average. It demonstrates that we are working on a very sparse situation. The auxiliary data matrix $\tilde{R}$ is constructed by 40 observed ratings for each user on average, which are randomly sampled from T_R. The sparsity of auxiliary data is 2 percent. The overlap between $\tilde{R}$ and $R(u, v, y_{u,v}, \tilde{y}_{u,v}/n/m)$ is 0.026, 0.062, 0.096 and 0.13 percent at different sparsity levels correspondingly.

To simulate heterogeneous target and auxiliary data (i.e., numerical and binary) for Moviepilot dataset, we relabel ratings $\tilde{R}$ with value $r_{u,v} \le 3$ in $\tilde{R}$ as 0 (dislike) and with value $r_{u,v} > 3$ as 1 (like). This pre-processing follows the common sense that if a user gives a high rating to an item, he or she likes the item and vice-versa. We simulate heterogeneous data for other datasets in a similar way.

We extract the second dataset Netflix in the same way as Moviepilot. The size of R in Netflix is $5,000 \times 5,000$. The overlap between $\tilde{R}$ and R is 0.035, 0.070, 0.10 and 0.14 percent at target domain sparsity level 0.2, 0.4, 0.6 and 0.8 percent respectively. By default, we set the target domain sparsity levels at 0.8 percent following [45] for these two datasets.

We construct the third dataset from Flixter[2] following the same setting in [45]. It contains 8.2×10^6 observed ratings given by 1.5×10^5 users towards 4.9×10^4 items. The range of ratings is {0.5, 1, 1.5, ... ,5}. For simulating binary feedback in the auxiliary domain, the rating which is larger than 4 is converted to “like”, and the rating which is smaller than 4 is converted to “dislike”.

We construct the fourth dataset from MovieLens10M[3] in the same way as [45]. This data contains 10×10^6 ratings given by 7.1×10^4 users on 1.1×10^4 products. The range of ratings is {0.5, 1, 1.5, ... ,5}.

We build the fifth dataset from MovieLens20M[4] in the same way as MovieLens10M. This data contains 20×10^6 ratings given by 1.4×10^5 users on 2.7×10^4 products. The range of ratings is {0.5, 1, 1.5,...,5}.

Table 1 summarizes the usage of the five datasets in previous works and the statistics of number of users, items and ratings on target, auxiliary, and test data. Previously, batch based methods [8], [22] evaluate the performance on Netflix and MoviePilot. SGD based methods [45] evaluate the performance on Netflix, Filxter and MovieLens10M dataset. MovieLens20M has not been applied on this problem yet.

6.2 Evaluation Metrics

We evaluate the prediction performance of each method with two metrics: Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), which is following [8], [22], [45]. The equations for MAE and RMSE are as following:

$$\mathrm{MAE} = \sum_{(u,v,r_{u,v}) \in T_E} |r_{u,v} - \hat{r}_{u,v}| / |T_E|, \tag{30}$$

$$\mathrm{RMSE} = \sqrt{\sum_{(u,v,r_{u,v}) \in T_E} (r_{u,v} - \hat{r}_{u,v})^2 / |T_E|}, \tag{31}$$

where $r_{u,v}$ and $\hat{r}_{u,v}$ denote the ground truth rating and predicted rating of the uth user and the vth item. $|T_E|$ denotes the number of test ratings.

2. www.cs.ubc.ca/ jamalim/datasets
3. www.grouplens.org/node/73/
4. https://grouplens.org/datasets/movielens/20m/

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on October 09,2021 at 03:35:47 UTC from IEEE Xplore. Restrictions apply.

TABLE 2
Compared Performance of Both Batch and SGD Algorithms on MoviePilot Dataset

	Algorithm	MAE	RMSE	Memory
batch	cPMF [19]	0.7462	0.9599	1.92
	CMF-link [21]	0.6905	0.9072	2.40
	TCF(CMTF) [8]	0.6776	0.8875	1.92
	TCF(CSVD) [8]	0.6612	0.8744	2.40
	DCSVD-B [22] (conference)	0.6590	0.8733	2.88
	LSCF-B [22] (conference)	0.6530	0.8644	2.40
	DLSCF-B [22] (conference)	0.6504	0.8626	2.88
SGD	RSVD [20]	0.7081	0.9043	1.76
	CMF [21]	0.6843	0.8954	1.76
	iTCF [45]	0.6700	0.8832	1.76
	DCSVD-S (baseline1)	0.6601	0.8711	1.76
	LSCF-S (baseline2)	<u>0.6474</u>	<u>0.8599</u>	1.68
	DLSCF-S (ours)	**0.6460**	**0.8549**	1.76

Bold and underline denote the best and second best performance.

6.3 Compared Methods

We compare our batch and SGD based DLSCF with the state-of-the-art batch and SGD based collective filtering methods.

6.3.1 Batch Based Methods

We compare batch based DLSCF (DLSCF-B) with four batch based transfer learning methods:

cPMF [19]: Constrained Probabilistic Matrix Factorization applies two latent factor matrices, user latent factor and item latent factor to model the preference data as shown in Eq. (1). [8] integrated the auxiliary data of PMF in their experiments.

CMF-link [8], [21]: CMF-link is short for CMF with logistic link function. CMF [21] jointly factorizes target and auxiliary domain rating matrices with the constraint of shared item latent factors, as shown in Eq. (2). However, both users and items should be aligned in CMF. CMF-link [8] embeds a logistic link function in the auxiliary data matrix factorization, to address the data heterogeneity problem in CMF,

TCF(CMTF), *TCF(CSVD)* [2], [8]: Transfer by Collective Factorization constructs a shared latent space and collectively learns the domain-dependent and domain-independent knowledge. Two variants of TCF with different constraints on user and item latent features are named as CMTF and CSVD.

More detailed descriptions for the compared methods could be found in [8]. We follow the settings of [8] strictly for a fair comparison.

We generate two variants of DLSCF: DCSVD and LSCF, in order to evaluate the effectiveness of low-rank sparse decomposition and the multi-layer structure.

We name the batch and SGD based DCSVD as DCSVD-B and DCSVD-S respectively, and the batch and SGD based LSCF as LSCF-B and LSCF-S.

DCSVD-B [22] (conference): We keep the multi-layer structure of DLSCF, but remove the low-rank sparse constraint via setting $\beta_1 = \beta_2 = 0$. DCSVD-B can also be regarded as a multi-layer variant of CSVD.

LSCF-B [22] (conference): This method is the one layer variant version of DLSCF. The detailed description of LSCF-B is in Section 3.3.

DLSCF-B [22] (conference): It is the batch based solution of DLSCF. It is our previous conference version of this journal extension.

6.3.2 SGD Based Methods

We evaluate the effectiveness of SGD based DLSCF (DLSCF-S) with three SGD based methods as following:

RSVD[20]: Regularized Singular Value decomposition learns latent factors of the corresponding user and item to approximate the observed rating scores. RSVD integrates neighborhood model and latent factor model, to better capture the localized and global information simultaneously.

CMF [21]: Collective Matrix Factorization assumes that target and auxiliary domain share same items' latent factors. In this way, it transfers user-side shared knowledge in auxiliary domain data. However, CMF may ignore the user-side auxiliary domain data, since it does not assume there is shared user specific knowledge.

iTCF [45]: Interaction-rich transfer by collective factorization (iTCF) captures both the shared user-side and item-side knowledge in auxiliary domain data, compared to CMF. It introduces rich interactions by assuming that two domain data share the same user and item latent factors and predictability.

We also implement SGD based DCSVD-S and LSCF-S for two reasons. The first is to evaluate the impact of low-rank sparse decomposition and the hierarchical structure in SGD based optimization algorithm. Second, we compare the corresponding SGD and batch based variants of DLSCF.

DCSVD-S: This method is the SGD based solution of DCSVD. We remove the low-rank sparse constraint by setting $\beta_1 = \beta_2 = 0$ in DLSCF-S.

LSCF-S: This method is the SGD based solution of LSCF. It would be easily implemented when we set $p = 1$ in Eq. (26) in DLSCF-S.

DLSCF-S: It is the SGD based solution of DLSCF as shown in Algorithm 2.

6.4 Experimental Results

The results in five datasets on test set for both batch and SGD based methods are reported in Tables 2, 3, 4, 5 and 6. We report the MAE, RMSE and memory for each method.

We conduct experiments with under different trade-off parameters $\lambda \in [0.1, 5]$, $\beta_1 \in [0, 1]$, $\beta_2 \in [0, 1]$. For multi-layer structure based methods DCSVD and DLSCF, we have tried different number of layers $p \in \{1, 2, 3, 4, 5\}$. In DLSCF-S, we have tried α_u and α_v from 0.1 to 5, and set $\alpha_u = 1$ and $\alpha_v = 1$. We set the parameters and convergence according to cross-validation. We observe that when the number of layers $p = 2$ or 3, DLSCF usually achieves lower error rate. The observation is consistent with the hierarchical categories of the real-world recommender. From Tables 2 to 6, we could observe that:

6.4.1 Results of Accuracy

(1) Among all the batch based methods, DLSCF-B achieves the lowest error under MAE and RMSE metric, comparing to the state-of-the-art batch based method TCF(CSVD) and its two variants DCSVD-B and LSCF-B.

(2) Among all the SGD based methods, DLSCF-S achieves the lowest error, comparing to the state-of-the-art SGD based method iTCF and its two variants DCSVD-S and LSCF-S.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on October 09,2021 at 03:35:47 UTC from IEEE Xplore. Restrictions apply.

TABLE 3
Compared Performance of Both Batch and SGD Algorithms on Netflix Dataset

	Algorithm	MAE	RMSE	Memory
batch	cPMF [19]	0.7642	0.9691	2.50
	CMF-link [21]	0.7295	0.9277	2.78
	TCF(CMTF) [8]	0.6962	0.8884	2.50
	TCF(CSVD) [8]	0.6877	0.8809	2.78
	DCSVD-B [22] (conference)	0.6862	0.8793	2.78
	LSCF-B [22] (conference)	0.6870	0.8780	2.78
	DLSCF-B [22] (conference)	0.6854	0.8775	2.83
SGD	RSVD [20]	0.7236	0.9201	1.54
	CMF [21]	0.7054	0.9020	1.54
	iTCF [45]	0.7014	0.8966	1.54
	DCSVD-S (baseline1)	0.6912	0.8936	1.68
	LSCF-S (baseline2)	$\underline{0.6838}$	$\underline{0.8762}$	1.54
	DLSCF-S (ours)	**0.6828**	**0.8749**	1.68

Bold and underline denote the best and second best performance.

TABLE 5
Compared Performance of Both Batch and SGD Algorithms on MovieLens10M Dataset

	Algorithm	MAE	RMSE	Memory
batch	cPMF [19]	0.6531	0.8354	4.75
	CMF-link [21]	0.6413	0.8326	4.75
	TCF(CMTF) [8]	0.6241	0.8213	4.75
	TCF(CSVD) [8]	0.6217	0.8153	4.99
	DCSVD-B [22] (conference)	0.6212	0.8212	5.51
	LSCF-B [22] (conference)	0.6154	0.8123	4.90
	DLSCF-B [22] (conference)	**0.6041**	0.8094	5.51
SGD	RSVD [20]	0.6438	0.8364	3.24
	CMF [21]	0.6334	0.8273	3.08
	iTCF [45]	0.6197	$\underline{0.8091}$	3.11
	DCSVD-S (baseline1)	0.6159	$\underline{0.8133}$	3.15
	LSCF-S (baseline2)	0.6133	0.8093	3.08
	DLSCF-S (ours)	$\underline{0.6121}$	**0.8088**	3.15

Bold and underline denote the best and second best performance.

To evaluate whether the improvement of DLSCF-B and DLSCF-S compared to other methods is significant, we conduct t-test and show the p value of each comparison under five datasets in Table 8. The lower the p value is, the more confident the difference is. Each comparison is run for 5 times. We observe that $p < 0.05$ in each comparison. It demonstrates the significance of the improvements.

(3) When comparing batch and SGD based solution of DLSCF, we could see that DLSCF-S is comparable or outperforms DLSCF-B. In MoviePilot, Netflix and MovieLens20M datasets, DLSCF-S achieves the lowest MAE and RMSE. In Flixter and MovieLens10, they are comparable. Similar observation could be achieved when comparing LSCF-B with LSCF-S, and comparing DCSVD-B with DCSVD-S. It demonstrates that with lower memory usage the proposed SGD could still achieve effective performance.

(4) In order to evaluate the effectiveness of low-rank sparse decomposition of DLSCF, we compare DLSCF with DCSVD, which is a variant of DLSCF by removing the low-rank sparsity constraint. We also compare the one layer model LSCF with CSVD. Similarly, CSVD can be regarded as a variant of LSCF but without the low-rank sparsity constraint. Experimental results show that in both SGD and batch based solutions, DLSCF achieves lower error rate than DCSVD and LSCF achieves lower error rate than CSVD. It demonstrates that under both multi-layer and one-layer structure, low-rank sparse decomposition is effective.

(5) To evaluate the effectiveness of the multi-layer model structure in cross-domain recommendation task, we also compare multi-layer models with their one-layer variant models. We compare DLSCF with LSCF and compare DCSVD with CSVD in both batch and SGD based solution. Experimental results show that DLSCF achieves lower error rate than LSCF and DCSVD also achieves lower error rate than CSVD, under both SGD and batch solutions. The experimental results demonstrate the effectiveness of the multi-layer structure.

In order to evaluate the effectiveness of our model on the data which are not simulated, we have added another setting that both target and auxiliary domains are not re-scaled, meaning that both domains are numerical. Actually, our model can be used not only for transferring binary data to numerical, but also for more general transfer learning problems, for example, transferring numerical rating to numerical rating. We evaluate our model on Netfilx dataset, and make both target and axillary domain numerical. In

TABLE 4
Compared Performance of Both Batch and SGD Algorithms on Flixter Dataset

	Algorithm	MAE	RMSE	Memory
batch	cPMF [19]	0.6479	0.8768	4.69
	CMF-link [21]	0.6456	0.8745	4.69
	TCF(CMTF) [8]	0.6394	0.8696	4.99
	TCF(CSVD) [8]	$\underline{0.6347}$	0.8632	5.32
	DCSVD-B [22] (conference)	0.6342	0.8624	5.51
	LSCF-B [22] (conference)	0.6348	0.8625	5.32
	DLSCF-B [22] (conference)	**0.6331**	$\underline{0.8621}$	5.51
SGD	RSVD [20]	0.6561	0.8814	3.94
	CMF [21]	0.6423	0.8710	3.91
	iTCF [45]	0.6373	0.8636	3.93
	DCSVD-S (baseline1)	0.6381	0.8631	3.95
	LSCF-S (baseline2)	0.6352	0.8623	3.86
	DLSCF-S (ours)	$\underline{0.6347}$	**0.8619**	3.95

Bold and underline denote the best and second best performance.

TABLE 6
Compared Performance of Both Batch and SGD Algorithms on MovieLens20M Dataset

	Algorithm	MAE	RMSE	Memory
batch	cPMF [19]	0.6532	0.9039	5.52
	CMF-link [21]	0.6511	0.8932	5.52
	TCF(CMTF) [8]	0.6432	0.8812	5.13
	TCF(CSVD) [8]	0.6445	0.8715	7.09
	DCSVD-B [22] (conference)	0.6364	0.8624	7.09
	LSCF-B [22] (conference)	0.6345	0.8594	7.09
	DLSCF-B [22] (conference)	$\underline{0.6315}$	0.8535	7.09
SGD	RSVD [20]	0.6572	0.9064	4.34
	CMF [21]	0.6492	0.8717	4.51
	iTCF [45]	0.6415	0.8627	4.51
	DCSVD-S (baseline1)	0.6382	0.8572	4.59
	LSCF-S (baseline2)	0.6331	$\underline{0.8511}$	4.53
	DLSCF-S (ours)	**0.6260**	**0.8419**	4.59

Bold and underline denote the best and second best performance.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on October 09,2021 at 03:35:47 UTC from IEEE Xplore. Restrictions apply.

TABLE 7
Compared Performance of Transferring Numerical to Numerical of Both Batch and SGD Algorithms on Netflix Dataset

	Algorithm	MAE	RMSE	Memory
batch	cPMF [19]	0.7094	0.9185	2.65
	CMF-link [21]	0.7084	0.9034	2.83
	TCF(CMTF) [8]	0.6946	0.8945	2.78
	TCF(CSVD) [8]	0.6877	0.8808	2.83
	DCSVD-B [22] (conference)	0.6854	0.8793	2.83
	LSCF-B [22] (conference)	0.6833	0.8750	2.83
	DLSCF-B [22] (conference)	<u>0.6714</u>	**0.8638**	2.97
SGD	RSVD [20]	0.7167	0.9157	1.54
	CMF [21]	0.7037	0.8985	1.54
	iTCF [45]	0.6984	0.8895	1.54
	DCSVD-S (baseline1)	0.6913	0.8837	1.72
	LSCF-S (baseline2)	0.6870	0.8747	1.68
	DLSCF-S (ours)	**0.6711**	<u>0.8662</u>	1.86

Bold and underline denote the best and second best performance.

this way, the axillary domain is not rescaled. We show the experimental results in Table 7.

From Table 7, we can see that when both target and auxiliary domain data are numerical and not simulated, our batch and SGD based methods, DLSCF-B and DLSCF-S, still achieve the best and second best performance. In terms of computational cost, SGD-based methods apply about 60 percent memory than batch based method, which is very similar as transferring binary data to numerical data in Netflix dataset in Table 3.

6.4.2 Results of Memory

We conduct the experiments on a computer with 48GiB memory, Intel i7 CPU with 3.50 GHz. We have reported the memory usage of the comparison methods on each dataset in Table 2 to Table 6.

We could see that all the SGD based methods generally take less memory than batch based methods. The difference among SGD or batch based methods is not large.

When we compare DLSCF-S with DLSCF-B, DLSCF-S takes less memory (about 60 percent) than DLSCF-B on all the datasets. Same phenomenon would be observed when comparing DCSVD-B with DCSVD-S, LSCF-B with LSCF-S. We could see the advantage of SDG based solution on the memory aspect facing same/similar model.

We would like to note that although the size of these benchmarks are already the top largest, in real world industry recommender such as Amazon.com, the number of users and items are considerably larger than these benchmarks. It may be even hard to load all the data into the memory at one time. Thus, we may face out of memory issue when using batch based methods. With SGD based method, we are able to save large memory (about 40 percent) and make the method computable.

6.5 Discussion

6.5.1 Discussion of Sparsity

We follow [8], [22] to discuss the effectiveness of solving data sparsity problem using the proposed DLSCF and comparison methods.

We randomly sample training set from T_R when the sparsity level is set at 0.2, 0.4, 0.6 and 0.8 percent in the target domain respectively. In Tables 9 and 10, we report the performance of non-transfer learning methods, batch based transfer learning methods and SGD based transfer learning methods, under Netflix and MovieLens10M. Three non-transfer learning methods PMF [19], SVD [46] and OptSpace [47] are included to evaluate the performance of transfer learning as [8]. The sparsity of auxiliary data $\tilde{R}$ is shown as Table 1, which is 2 percent of Netflix and 0.52 percent of MovieLens10M dataset. In Table 9, the results of non-transfer learning methods, batch based transfer learning methods are copied from [8], [22].

First, from Tables 9 and 10, we observe that in both Netflix and MovieLens10M datasets, under different sparsity levels, DLSCF-B and DLSCF-S achieve the best and second best performance. When comparing DLSCF-S and DLSCF-B, in Netflix dataset under sparsity level 0.6 and 0.8 percent, DLSCF-S outperforms DLSCF-B under both MAE and RMSE. Under sparsity level 0.2 and 0.4 percent, and under all the sparsity level in MovieLens10M dataset, the performances of DLSCF-S and DLSCF-B are comparable.

Second, Tables 9 and 10 show that almost all the transfer learning based methods achieve lower error rate compared with the non-transfer ones. For example, CMTF-link has already outperformed non-transfer methods PMF, SVD and OptSpace. It shows the effectiveness of solving the sparsity problem using transfer learning methods.

We also discuss the performance of DLSCF when the sparsity level of auxiliary domain data is set at 1, 2 and 3 percent. Fig. 5 shows the MAE and RMSE of DLSCF-B on MoviePilot dataset as an example. The observation is the same as [15]. Less sparse the target or auxiliary domain data usually achieves lower error rate.

TABLE 8
p Values of t-Test of Comparison Methods under RMSE Metric

Dataset	MoviePilot	Netflix	Flixter	MovieLens10M	MovieLens20M
DLSCF-B vs CMTF	2.27×10^{-5}	4.52×10^{-4}	6.32×10^{-4}	3.37×10^{-5}	7.76×10^{-5}
DLSCF-B vs CSVD	3.74×10^{-5}	4.65×10^{-4}	8.43×10^{-4}	2.96×10^{-4}	3.23×10^{-4}
DLSCF-B vs DCSVD-B	1.09×10^{-4}	0.0021	6.32×10^{-4}	7.32×10^{-4}	2.64×10^{-4}
DLSCF-B vs LSCF-B	0.0063	0.0346	0.0074	0.0023	0.0027
DLSCF-S vs CMF	3.24×10^{-4}	4.32×10^{-4}	0.0032	3.74×10^{-5}	0.0064
DLSCF-S vs iTCF	3.85×10^{-4}	0.0023	0.0085	8.832×10^{-4}	8.32×10^{-5}
DLSCF-S vs DCSVD-S	7.23×10^{-5}	7.42×10^{-4}	8.32×10^{-4}	3.75×10^{-4}	0.0047
DLSCF-S vs LSCF-S	0.0023	0.0075	7.45×10^{-4}	9.75×10^{-4}	4.32×10^{-4}

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on October 09,2021 at 03:35:47 UTC from IEEE Xplore. Restrictions apply.

TABLE 9
Discussion of Performance of Compared Methods under Different Sparsity Levels of Target Domain on Netflix

	Sparsity	0.2%		0.4%		0.6%		0.8%	
	Metrics	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
non-transfer	PMF [19]	0.8879	1.0779	0.8467	1.0473	0.8087	1.0205	0.7642	0.9691
	SVD [46]	0.8055	1.0202	0.7846	0.9906	0.7757	0.9798	0.7711	0.9741
	OptSpace [47]	0.8276	1.0676	0.7812	1.0089	0.7572	0.9750	0.7418	0.9543
transfer(batch)	cPMF	0.8491	1.0606	0.8147	1.0125	0.8122	1.0066	0.7864	0.9930
	CMF-link [21]	0.7994	1.0204	0.7508	0.9552	0.7365	0.9369	0.7295	0.9277
	TCF(CMTF) [8]	0.7589	0.9653	0.7195	0.9171	0.7031	0.8971	0.6962	0.8884
	TCF(CSVD) [8]	0.7405	0.9502	0.7080	0.9074	0.6948	0.8903	0.6877	0.8809
	DCSVD-B ([22] (conference)	0.7379	0.9466	0.7076	0.9067	0.6942	0.8892	0.6862	0.8793
	LSCF-B ([22] (conference)	0.7380	0.9470	0.7069	0.9062	0.6937	0.8890	0.6870	0.8780
	DLSCF-B ([22] (conference)	**0.7369**	0.9460	**0.7060**	0.9054	0.6929	0.8883	0.6854	0.8775
transfer(SGD)	RSVD [20]	0.7854	0.9857	0.7528	0.9564	0.7452	0.9486	0.7236	0.9201
	CMF [21]	0.7791	0.9735	0.7396	0.9472	0.7456	0.9232	0.7054	0.9020
	iTCF [45]	0.7583	0.9599	0.7236	0.9211	0.7208	0.9086	0.7014	0.8966
	DCSVD-S (baseline1)	0.7421	0.9473	0.7081	0.9049	0.6932	0.8874	0.6912	0.8936
	LSCF-S (baseline2)	0.7403	0.9442	0.7074	0.9047	0.6922	0.8866	0.6838	0.8762
	DLSCF-S (ours)	0.7376	**0.9413**	0.7067	**0.9028**	**0.6908**	**0.8854**	**0.6828**	**0.8749**

Bold and underline denote the best and second best performance.

6.5.2 Discussion of λ, β_1, β_2 in DLSCF-B

In this section, we discuss the impact of parameters λ, β_1, β_2 settings of DLSCF-B. In Fig. 3, we discuss λ, β_1, β_2 one by one through fixing the other parameters. We show an example in which the target domain sparsity level is 0.6 percent, the number of layers $p = 1$ and the layer size $d_1 = 10$ under MoviePilot dataset.

In Fig. 3a, we show RMSE and MAE when λ is across 0.05 to 1 by fixing $\beta_1 = 0.005$ and $\beta_2 = 0.00005$. We observe that DLSCF-B achieves the lowest RMSE and MAE when λ is set around 0.3. When λ is moved away from 0.3, both RMSE and MAE increase gradually. When $\lambda = 0$, RMSE is 0.9003 and MAE is 0.6839. In this case, the auxiliary domain tri-factorizing term $\|\tilde{Y} \odot (\tilde{R} - U\tilde{B}V^{\top})\|_F^2$ looses the impact.

In Fig. 3b, we show RMSE and MAE when β_1 is across 1×10^{-6} to 0.001 by fixing $\lambda = 0.3$ and $\beta_2 = 0.00005$. We observe that DLSCF-B achieves lower RMSE and MAE when β_1 is from 3×10^{-5} to 6×10^{-5}. It achieves lowest RMSE = 0.8642 and MAE = 0.6530 when $\beta_1 = 5 \times 10^{-5}$. Both RMSE and MAE increase gradually when moving β_1 away from 0.005.

In Fig. 3c, we show RMSE and MAE when β_2 is from 5×10^{-4} to 0.05 by fixing $\lambda = 0.3$ and $\beta_3 = 0.005$. We observe that DLSCF-B achieves the lowest error when $\beta_2 = 0.005$ under both RMSE and MAE, and we have RMSE = 0.8642 and MAE = 0.6530. When β_2 is bigger than 0.02, the error rate increases sharply.

6.5.3 Discussion of λ, β_1, β_2 in DLSCF-S

In this section, we discuss the impact of parameters λ, β_1, β_2 settings of DLSCF-S. We show the performance of DLSCF-S in the MoviePilot dataset in Fig. 4, when fixing $p = 1$, $d_1 = 10$ and the sparsity level of target domain data at 0.6. We have tried α_u and α_v from 0.1 to 5, and fixed $\alpha_u = 1$ and $\alpha_v = 1$.

TABLE 10
Discussion of Performance of Compared Methods under Different Sparsity Levels of Target Domain on MovieLens10M

	Sparsity	0.2%		0.4%		0.6%		0.8%	
	Metrics	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
non-transfer	PMF [19]	0.8064	0.9373	0.7368	0.9168	0.7046	0.8736	0.6943	0.8532
	SVD [46]	0.7943	0.9225	0.7143	0.8953	0.6825	0.8621	0.6835	0.8434
	OptSpace [47]	0.7891	0.9286	0.7153	0.8986	0.6799	0.8539	0.6713	0.8428
transfer(batch)	cPMF	0.7585	0.9077	0.6883	0.8698	0.6474	0.8304	0.6432	0.8275
	CMF-link [21]	0.7456	0.9054	0.6843	0.8685	0.6411	0.8321	0.6325	0.8236
	TCF(CMTF) [8]	0.7386	0.8965	0.6754	0.8574	0.6224	0.8184	0.6145	0.8143
	TCF(CSVD) [8]	0.7335	0.8964	0.6743	0.8534	0.6203	0.8151	0.6136	0.8147
	DCSVD-B ([22] (conference)	0.7254	0.8843	0.6636	0.8343	0.6142	0.8146	0.6014	0.8122
	LSCF-B ([22] (conference)	0.7012	0.8662	0.6546	0.8435	0.6121	0.8103	0.6032	0.8137
	DLSCF-B ([22] (conference)	**0.6886**	0.8485	0.6453	**0.8224**	**0.6011**	0.8065	0.5972	**0.7911**
transfer(SGD)	RSVD [20]	0.7496	0.9183	0.6864	0.8754	0.6430	0.8315	0.6267	0.8212
	CMF [21]	0.7278	0.9036	0.6785	0.8657	0.6258	0.8187	0.6193	0.8087
	iTCF [45]	0.7134	0.9076	0.6643	0.8584	0.6168	0.8046	0.6023	0.8027
	DCSVD-S (baseline1)	0.7143	0.8924	0.6639	0.8524	0.6144	0.8057	0.6035	0.8099
	LSCF-S (baseline2)	0.7024	0.8742	0.6524	0.8482	0.6122	0.8048	0.6036	0.8196
	DLSCF-S (ours)	0.6925	**0.8446**	**0.6356**	0.8256	0.6035	**0.8012**	**0.5943**	0.7962

Bold and underline denote the best and second best performance.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on October 09,2021 at 03:35:47 UTC from IEEE Xplore. Restrictions apply.

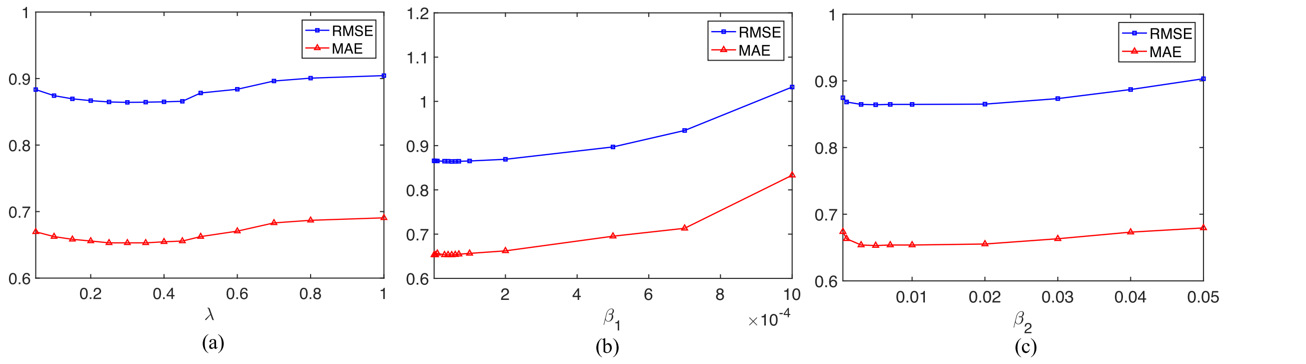


Fig. 3. Parameters λ, β_1, β_2 analysis of DLSCF-B. In figure (a), we show RMSE and MAE when λ is in the range from 0.05 to 1 by fixing $\beta_1 = 0.005$ and $\beta_2 = 0.00005$. In figure (b), we show RMSE and MAE when β_1 is from 1×10^{-6} to 0.001 by fixing $\lambda = 0.3$ and $\beta_2 = 0.00005$. In figure (c), we show RMSE and MAE when β_2 is from 5×10^{-4} to 0.05 by fixing $\lambda = 0.3$ and $\beta_3 = 0.005$.

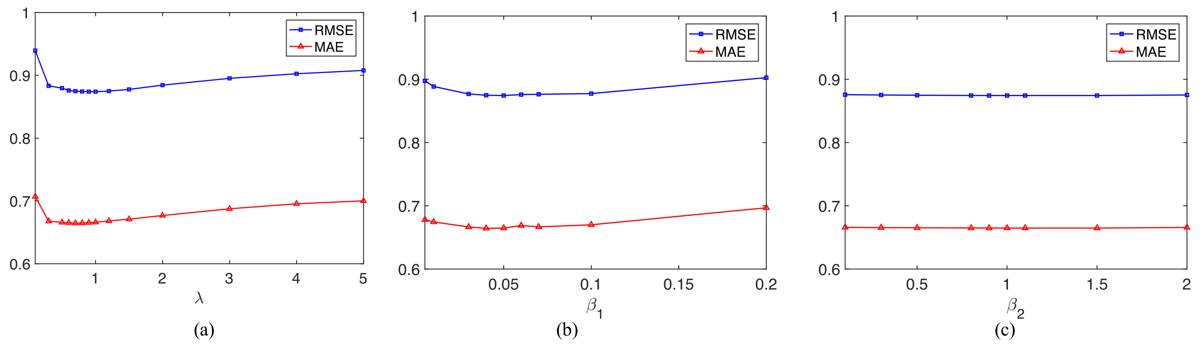


Fig. 4. Parameters λ, β_1, β_2 analysis of DLSCF-S. In figure (a), we present RMSE and MAE when λ is in the range from 0.1 to 5 with fixing $\beta_1 = 0.05$ and $\beta_2 = 1$. In figure (b), we show RMSE and MAE when β_1 is from 0.005 to 0.2 by fixing $\lambda = 0.8$ and $\beta_2 = 0.05$. In figure (c), we show RMSE and MAE when β_2 is from 0.1 to 2 by fixing $\lambda = 0.8$ and $\beta_3 = 1$.

We observe from Fig. 4a that DLSCF-S is with lower MAE and RMSE when λ is from 0.3 to 1, which shows that the prediction error of DLSCF-S is not sensitive to the λ. We achieve the lowest RMSE $= 0.8740$ at $\lambda = 0.9$ to 1, and lowest MAE $= 0.6645$ at $\lambda = 0.7$. Similar as in DLSCF-B, when λ is too small, the performance is not good. It is because very limited knowledge is borrowed from auxiliary domain, and the model is degraded to non-transfer learning model. When λ is very large, the impact of target domain data is decreased and the error gradually increases.

From Fig. 4b, we notice that DLSCF-S achieves lower error when β_1 is in the range of 0.03 to 0.07 under both RMSE and MAE. When β_1 goes further way from 0.05, the error rate increases gradually under both RMSE and MAE.

In Fig. 4c, we witness that under both RMSE and MAE, DLSCF-S is not sensitive to β_2. We achieve the lowest RMSE $= 0.8743$ and MAE $= 0.6648$ when $\beta_2 = 1$. When β_2 goes further away from 1, both RMSE and MAE increase gradually. When $\beta_2 = 0.3$, we have RMSE $= 0.8757$ and MAE $= 0.6655$. When $\beta_2 = 2$, we have RMSE $= 0.8748$ and MAE $= 0.6658$.

6.5.4 Discussion of Layer Size in DLSCF-B

In this section, we discuss the impact of layer size settings in batch based model DLSCF-B. Fig. 6 shows the performance of DLSCF-B under different layer size settings when fixing the sparsity level at 0.8 percent in MoviePilot dataset. The number of layers p is fixed at 2. We conduct the experiments under different layer size d_1 when the number of $d_2 = 10$ is fixed according to [15]. We vary the value of d_1 within $\{50, 100, 150, 200, 300, 400, 500, 1000\}$. DLSCF-B achieves lower MAE and RMSE when d_1 is around 200 to 400 as shown in the Fig. 6.

6.5.5 Discussion of Layer Size in DLSCF-S

In this section, we discuss the impact of layer size settings in the SGD based model DLSCF-S in Fig. 7.

We start from discussing the layer size in the one-layer DLSCF-S model (also named as LSCF-S) in Fig. 7a. Fig. 7a shows the performance of DLSCF-S under different layer size settings at sparsity level 0.6 percent in MoviePilot

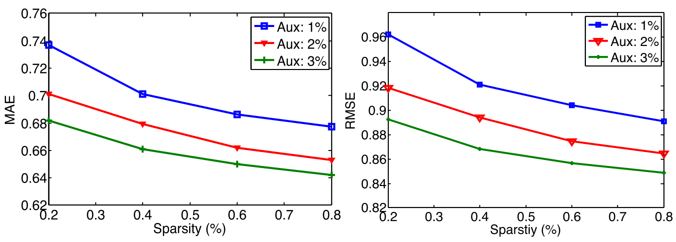


Fig. 5. Prediction error of DLSCF-B on MoviePilot dataset when auxiliary domain data is at different sparsity levels.

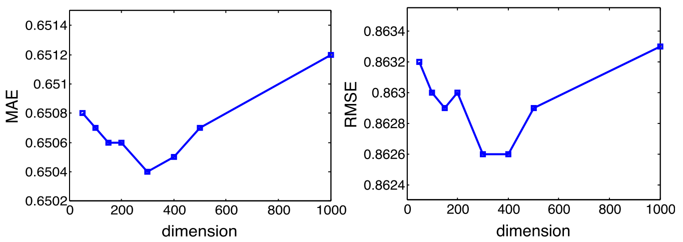
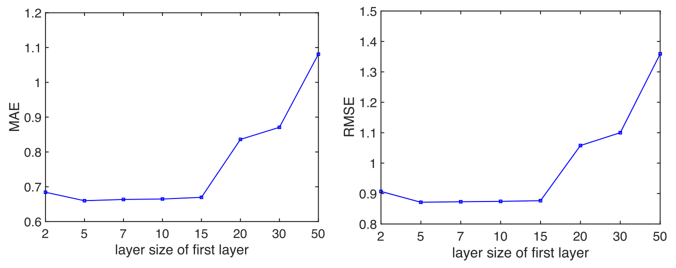
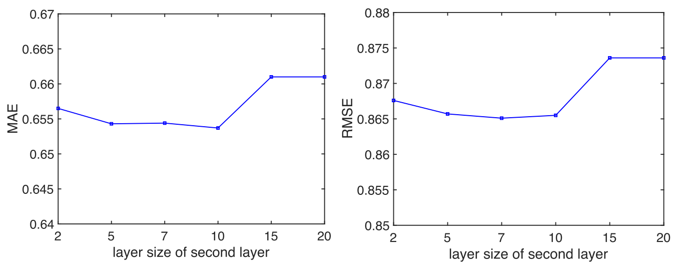


Fig. 6. Prediction error of DLSCF-B on MoviePilot dataset at different dimensions of the first layer.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on October 09,2021 at 03:35:47 UTC from IEEE Xplore. Restrictions apply.



(a) Prediction error in the first layer



(b) Prediction error in the second layer

Fig. 7. Prediction error of DLSCF-S on MoviePilot dataset at different dimensions of the one-layer model in (a) and two-layer model in (b).

dataset. We only use one layer ($p = 1$) and discuss the impact of layer size d_1. We vary the value of d_1 from 3 to 10. We could see that DLSCF-S achieves lower error when d_1 is around 5 to 10 under both MAE and RMSE. The lowest MAE is 0.6599 and the lowest RMSE is 0.8714 when $d_1 = 5$.

Second, we discuss the layer size in the two-layer model ($p = 2$). We fix $d_1 = 30$ and vary the value of d_2 within $\{2, 5, 7, 10, 15, 20\}$. In one-layer model, MAE = 0.8361, RMSE = 1.0578 when $d_1 = 30$, as shown in Fig. 7a. When we use two-layer model, the lowest MAE is 0.6537 when $d_2 = 10$ and the lowest RMSE is 0.8651 when $d_2 = 7$. When comparing the MAE and RMSE with the same layer size in Figs. 7a and 7b, all the MAE and RMSE in (b) are lower than in (a), which shows the effectiveness of the multi-layer structure.

7 Conclusions

In this paper, we presented a novel transfer learning framework Deep Low-rank Sparse Collective Factorization for heterogeneous recommendation. We transfered knowledge from binary rating data to facilitate the recommendation for numerical rating. We captured more shared rating patterns across two domains through a low-rank constraint on the common rating pattern and a group sparsity constraint on the domain-specific rating pattern. In this way, we can transfer more shared knowledge from auxiliary domain to target domain. Furthermore, we explored the deep structure of the model based on the hierarchical structure of the real-world recommender. We provided both batch and SGD based optimization algorithms. Experiments on five datasets showed the effectiveness of proposed algorithms.

Acknowledgments

This work is supported in part by the NSF IIS award 1651902 and U.S. Army Research Office Young Investigator Award W911NF-14-1-0218.

References

[1] N. N. Liu, E. W. Xiang, M. Zhao, and Q. Yang, "Unifying explicit and implicit feedback for collaborative filtering," in *Proc. 19th ACM Int. Conf. Inf. Knowl. Manage.*, 2010, pp. 1445–1448.
[2] W. Pan, N. N. Liu, E. W. Xiang, and Q. Yang, "Transfer learning to predict missing ratings via heterogeneous user feedbacks," in *Proc. IJCAI Int. Joint Conf. Artif. Intell.*, 2011, Art. no. 2318.
[3] S. Sahebi and P. Brusilovsky, "It takes two to tango: An exploration of domain pairs for cross-domain collaborative filtering," in *Proc. 9th ACM Conf. Recommender Syst.*, 2015, pp. 131–138.
[4] X. Xin, Z. Liu, C.-Y. Lin, H. Huang, X. Wei, and P. Guo, "Cross-domain collaborative filtering with review text," in *Proc. 24th Int. Conf. Artif. Intell.*, 2015, pp. 1827–1834.
[5] Y.-F. Liu, C.-Y. Hsu, and S.-H. Wu, "Non-linear cross-domain collaborative filtering via hyper-structure transfer," in *Proc. 32nd Int. Conf. Int. Conf. Mach. Learn.*, 2015, pp. 1190–1198.
[6] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Trans. Knowl. Discovery Data*, vol. 4, no. 1, 2010, Art. no. 1.
[7] Y. Zhang and J. Nie, "Probabilistic latent relational model for integrating heterogeneous information for recommendation," Technical report, School of Engineering, UCSC, 2010, https://pdfs.semanticscholar.org/6bca/8c601f338dd5f33e2e586c6735c58c20c306.pdf
[8] W. Pan and Q. Yang, "Transfer learning in heterogeneous collaborative filtering domains," *Artif. Intell.*, vol. 197, pp. 39–55, 2013.
[9] B. Li, Q. Yang, and X. Xue, "Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction," in *Proc. 21st Int. Joint Conf. Artif. Intell.*, 2009, pp. 2052–2057.
[10] B. Li, Q. Yang, and X. Xue, "Transfer learning for collaborative filtering via a rating-matrix generative model," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 617–624.
[11] S. Gao, H. Luo, D. Chen, S. Li, P. Gallinari, and J. Guo, "Cross-domain recommendation via cluster-level latent factor model," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2013, pp. 161–176.
[12] O. Moreno, B. Shapira, L. Rokach, and G. Shani, "TALMUD: Transfer learning for multiple domains," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, 2012, pp. 425–434.
[13] P. Cremonesi and M. Quadrana, "Cross-domain recommendations without overlapping data: Myth or reality?" in *Proc. 8th ACM Conf. Recommender Syst.*, 2014, pp. 297–300.
[14] C.-H. Lee, Y.-H. Kim, and P.-K. Rhee, "Web personalization expert with combining collaborative filtering and association rule mining technique," *Expert Syst. Appl.*, vol. 21, no. 3, pp. 131–137, 2001.
[15] S. Wang, J. Tang, Y. Wang, and H. Liu, "Exploring implicit hierarchical structures for recommender systems," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 1813–1819.
[16] K. Lu, G. Zhang, R. Li, S. Zhang, and B. Wang, "Exploiting and exploring hierarchical structure in music recommendation," in *Proc. Asia Inf. Retrieval Technol. Symp.*, 2012, pp. 211–225.
[17] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. Schuller, "A deep semi-NMF model for learning hidden representations," in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, pp. 1692–1700.
[18] H. Zhao, Z. Ding, and Y. Fu, "Multi-view clustering via deep matrix factorization," in *Proc. Assoc. Advancement Artif. Intell.*, 2017, pp. 2921–2927.
[19] A. Mnih and R. Salakhutdinov, "Probabilistic matrix factorization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2007, pp. 1257–1264.
[20] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 426–434.
[21] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 650–658.
[22] S. Jiang, Z. Ding, and Y. Fu, "Deep low-rank sparse collective factorization for cross-domain recommendation," in *Proc. ACM Multimedia Conf.*, 2017, pp. 163–171. [Online]. Available: http://doi.acm.org/10.1145/3123266.3123361
[23] F. Abel, E. Herder, G.-J. Houben, N. Henze, and D. Krause, "Cross-system user modeling and personalization on the social web," *User Model. User-Adapted Interaction*, vol. 23, no. 2/3, pp. 169–209, 2013.
[24] B. Shapira, L. Rokach, and S. Freilikhman, "Facebook single and cross domain data for recommendation systems," *User Model. User-Adapted Interaction*, vol. 23, no. 2/3, pp. 211–247, 2013.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on October 09,2021 at 03:35:47 UTC from IEEE Xplore. Restrictions apply.

[25] S. Berkovsky, T. Kuflik, and F. Ricci, "Mediation of user models for enhanced personalization in recommender systems," *User Model. User-Adapted Interaction*, vol. 18, no. 3, pp. 245–286, 2008.

[26] P. Cremonesi, A. Tripodi, and R. Turrin, "Cross-domain recommender systems," in *Proc. 11th Int. Conf. Data Mining Workshops*, 2011, pp. 496–503.

[27] A. Tiroshi, S. Berkovsky, M. A. Kaafar, T. Chen, and T. Kuflik, "Cross social networks interests predictions based ongraph features," in *Proc. 7th ACM Conf. Recommender Syst.*, 2013, pp. 319–322.

[28] M. Enrich, M. Braunhofer, and F. Ricci, "Cold-start management with cross-domain collaborative filtering and tags," in *Proc. Int. Conf. E-Commerce Web Technol.*, 2013, pp. 101–112.

[29] I. Fern ández-Tobías and I. Cantador, "Exploiting social tags in matrix factorization models for cross-domain collaborative filtering," in *Proc. 1st Workshop New Trends Content-Based Recommender Syst.*, 2014, pp. 34–41.

[30] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and C. Zhu, "Personalized recommendation via cross-domain triadic factorization," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 595–606.

[31] W. Pan, E. W. Xiang, N. N. Liu, and Q. Yang, "Transfer learning in collaborative filtering for sparsity reduction," in *Proc. 24th AAAI Conf. Artif. Intell.*, 2010, pp. 230–235.

[32] Y. Zhang, B. Cao, and D.-Y. Yeung, "Multi-domain collaborative filtering," in *Proc. 26th Conf. Uncertainty Artifi. Intell.*, 2010, pp. 725–732.

[33] B. Cao, N. N. Liu, and Q. Yang, "Transfer learning for collective link prediction in multiple heterogenous domains," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 159–166.

[34] Z. Ding, M. Shao, and Y. Fu, "Deep robust encoder through locality preserving low-rank dictionary," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 567–582.

[35] Z. Ding, M. Shao, and Y. Fu, "Latent low-rank transfer subspace learning for missing modality recognition," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 1192–1198.

[36] Z. Ding and Y. Fu, "Low-rank common subspace for multi-view learning," in *Proc. IEEE Int. Conf. Data Mining*, 2014, pp. 110–119.

[37] Z. Wang, S. Chang, Y. Yang, D. Liu, and T. S. Huang, "Studying very low resolution recognition using deep networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4792–4800.

[38] Y. Wu, J. Li, Y. Kong, and Y. Fu, "Deep convolutional neural network with independent softmax for large scale face recognition," in *Proc. ACM Multimedia Conf.*, 2016, pp. 1063–1067.

[39] S. Jiang, M. Shao, C. Jia, and Y. Fu, "Consensus style centralizing auto-encoder for weak style classification," in *AAAI*, 2016, pp. 1223–1229.

[40] M. Maleszka, B. Mianowska, and N. T. Nguyen, "A method for collaborative recommendation using knowledge integration tools and hierarchical structure of user profiles," *Knowl.-Based Syst.*, vol. 47, pp. 1–13, 2013.

[41] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM J. Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[42] J. Yang, W. Yin, Y. Zhang, and Y. Wang, "A fast algorithm for edge-preserving variational multichannel image restoration," *SIAM J. Imag. Sci.*, vol. 2, no. 2, pp. 569–592, 2009.

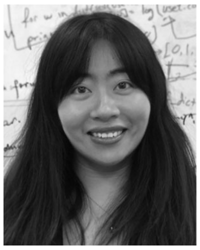
[43] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 69–77.

[44] J. Wang, S. C. Hoi, P. Zhao, and Z.-Y. Liu, "Online multi-task collaborative filtering for on-the-fly recommender systems," in *Proc. 7th ACM Conf. Recommender Syst.*, 2013, pp. 237–244.

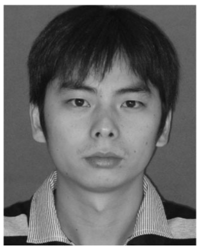
[45] W. Pan and Z. Ming, "Interaction-rich transfer learning for collaborative filtering with heterogeneous user feedback," *IEEE Intell. Syst.*, vol. 29, no. 6, pp. 48–54, Nov./Dec. 2014.

[46] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender system-a case study," Minnesota Univ. Minneapolis Dept. Comput. Sci., Rep. TR-00-043, 2000.

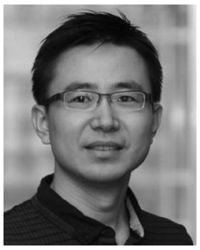
[47] R. Keshavan, A. Montanari, and S. Oh, "Matrix completion from noisy entries," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2009, pp. 952–960.



Shuhui Jiang received the BS and MS degrees in Xi'an Jiaotong University, Xi'an, China, in 2007 and 2011, respectively, and the PhD degree from the Department of Electrical and Computer Engineering, Northeastern University, in 2018. She was the recipient of the Dean's Fellowship of Northeastern University from 2014. She was a research intern with Adobe research lab, San Jose, in summer 2016. She is interested in machine learning, multimedia and computer vision. She has served as a reviewer for IEEE journals: the *IEEE Transactions on Neural Networks and Learning Systems*, the *IEEE Transactions on Multimedia* etc. She was the recipients of the best best paper candidate (ACM MM 2017).



Zhengming Ding (S'14-M'18) received the BEng degree in information security, the MEng degree in computer software and theory from the University of Electronic Science and Technology of China (UESTC), China, in 2010 and 2013, respectively, and the PhD degree from the Department of Electrical and Computer Engineering, Northeastern University, in 2018. He is a faculty member affiliated with Department of Computer, Information and Technology, Indiana University-Purdue University Indianapolis since 2018. His research interests include transfer learning, multi-view learning and deep learning. He received the National Institute of Justice Fellowship during 2016-2018. He was the recipient of the best paper award (SPIE 2016) and best paper candidate (ACM MM 2017). He is currently an associate editor of the *Journal of Electronic Imaging* (JEI). He is a member of the IEEE.



Yun Fu (S'07-M'08-SM'11-F'19) received the BEng degree in information engineering, the MEng degree in pattern recognition and intelligence systems from Xi'an Jiaotong University, China, respectively, the MS degree in statistics and the PhD degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign, respectively. He is an interdisciplinary faculty member affiliated with College of Engineering and the Khoury College of Computer and Information Sciences at Northeastern University since 2012. His research interests are machine learning, computational intelligence, big data mining, computer vision, pattern recognition, and cyber-physical systems. He has extensive publications in leading journals, books/book chapters and international conferences/workshops. He serves as associate editor, chairs, PC member and reviewer of many top journals and international conferences/workshops. He received seven Prestigious Young Investigator Awards from NAE, ONR, ARO, IEEE, INNS, UIUC, Grainger Foundation; nine Best Paper Awards from IEEE, IAPR, SPIE, SIAM; many major Industrial Research Awards from Google, Samsung, and Adobe, etc. He is currently an associate editor of the *IEEE Transactions on Neural Networks and Leaning Systems* (TNNLS). He is fellow of the IEEE, IAPR, OSA and SPIE, a lifetime distinguished member of ACM, lifetime member of AAAI and Institute of Mathematical Statistics, member of ACM Future of Computing Academy, Global Young Academy, AAAS, INNS and Beckman Graduate fellow during 2007-2008.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on October 09,2021 at 03:35:47 UTC from IEEE Xplore. Restrictions apply.