Heterogeneous Recommendation via Deep Low-rank Sparse Collective Factorization

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

Abstract—Real-world recommender usually makes use of heterogeneous types of user feedbacks—for example, binary ratings such as likes and dislikes and numerical rating such as 5-star grades. In this work, we focus on transferring knowledge from binary ratings to numerical ratings, facing more serious data sparsity problem. Conventional Collective Factorization methods usually assume that multiple domains share some common latent information across users and items. However, related domains may also share some knowledge of rating patterns. Furthermore, existing works may also fail to consider the hierarchical structures (i.e., genre, sub genre, detailed-category) in heterogeneous recommendation scenario. To address these challenges, in this paper, we propose a novel Deep Low-rank Sparse Collective Factorization (DLSCF) to facilitate the cross-domain recommendation. Specifically, the low-rank sparse decomposition is adopted to capture the shared rating patterns in multiple domains while splitting the domain-specific patterns. We also factorize the model in multiple layers to capture the affiliation relation between latent categories and latent 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 our proposed algorithms, by comparing them with several state-of-the-art batch and SGD based approaches.

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

1 Introduction

In real-world recommender, 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 for the easy operation in which they only need to click like or dislike button. Compared to binary ratings, 5-star ratings not only provide the like or dislike information, but also provide information of how much the customer likes or dislikes it, which should be more comprehensive to reflect the users' preference. However, in real world rating system, users may only rate a very limited number of items, especially 5-star ratings, while the item space is often very huge, which causes the data sparsity problem. Take Amazon product ratings as an example. A set of users may give binary ratings to product set B and 5-star ratings to product set C. B and C have a very small or no overlap. In this paper, we want to predict the 5-star ratings of product set B by transferring the rating knowledge from binary ratings. Although the scale of 0/1 ratings and 1-5 ratings are different, users' preferences towards categories are consistent across two rating strategies.

Cross-domain collaborative filtering (CDCF) is a powerful tool to address the data sparsity problem, which manages to adopt useful knowledge from other related domains [1], [2], [3], [4], [5]. In this paper, we solve the sparsity problem of numerical ratings (e.g., 5-star grades) through transferring binary rating (i.e., likes/dislikes) from auxiliary domain to numerical rating in target domain. Only a few previous works have exploited on this scenario: using "whether rate" data [1], [6] or "whether purchased" [7] to improve the numerical ratings prediction. Among the previous works, Liu et al. [1] paved the way of integrating the "whether rate" data to numerical form rating by investigation the cross domain collective matrix factorization (CMF) approach [8]. Pan at el. presented a framework named as Transfer by Collective Factorization (TCF) [2], [9] to transfer the shared latent user and item latent factors in both two domains. However, TCF neglects the shared knowledge in rating patterns. Recently cross-domain recommendation methods also work on transferring knowledge of similar rat-

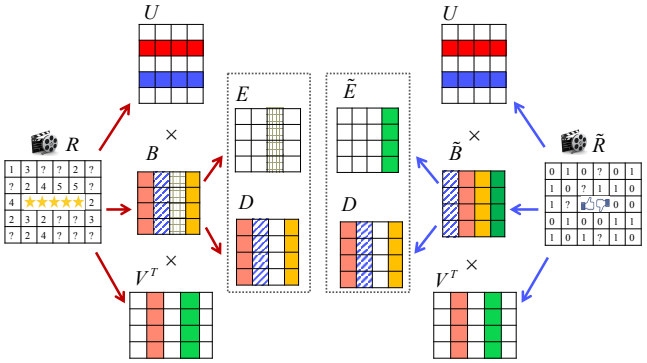


Figure 1. Illustration of the Low-rank Sparse Collective Factorization Model. R is a numerical rating matrix in the target domain and $\tilde{R}$ is a binary rating matrix in the auxiliary domain. U and V are user and item latent interests. Assume that two rating patterns B and $\tilde{B}$ share a common patterns D with low-rank constrains and also preserve domain-specific patterns E and $\tilde{E}$ with column sparse constraints.

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

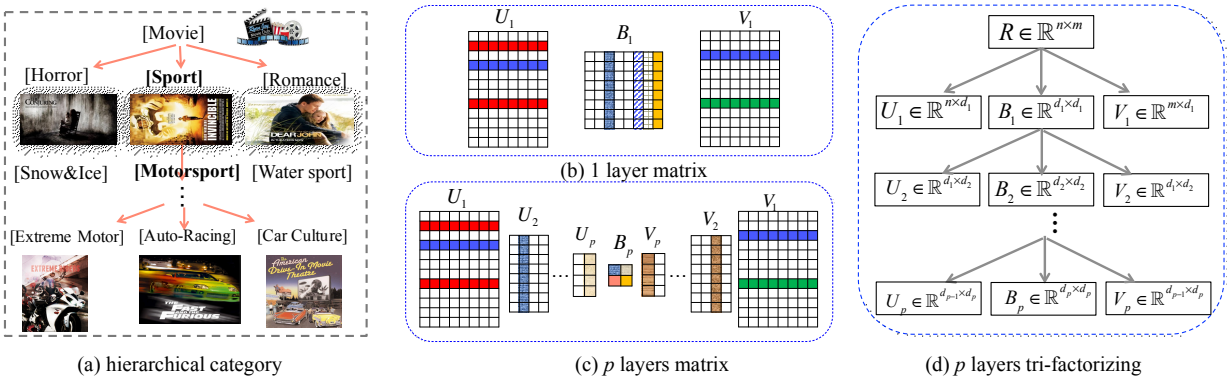


Figure 2. Implicit Hierarchical Structures via Deeply Tri-Factorizing the Rating Pattern Matrix of Target Domain. Figure (a) illustrates an example of the hierarchical structure in the movie recommendation system. We illustrate the matrix after tri-factorizing R in the first layer into U_1, B_1 and V_1 in figure (b), and p layer matrix $U_1...U_p$, $V_1...V_p$ and B_p in figure (c). The tri-factorization process is shown in figure (d), in which the rating patten B_i is further tri-factorized into U_{i+1}, B_{i+1} and V_{i+1} until the p-th layer. $d_1 \geq d_2... \geq ... \geq ...d_p$.

ing patterns, which have demonstrated that related domains do share a certain part of rating patterns [10], [11], [12], [13], [14], [15]. For example, Gao et al. [12] proposed a cluster-level latent feature model to learn the shared part of rating patterns of user groups.

Furthermore, previous works in cross-domain recommendation scenarios did not explore the hierarchical structures in the real-word recommendation system. For example, in Netflix DVD rental page, a hierarchical structure is applied to categorize movies as genre $\rightarrow$ sub-genre $\rightarrow$ detailed-category[1]. Based on the hierarchical structure in real-word recommendation system, exploiting the hierarchical latent factors of items or users attracted a lot of attentions by researchers recently [16], [17], and the hierarchical structure has demonstrated its effectiveness. Later, [18] formulated and optimized a deep version for matrix factorization. However, these works only focused on the single domain recommendation, instead of the cross-domain scenario. In cross-domain scenario, mostly, users' preferences are also presented a hierarchical structure. For example, if a user like sub-genres of Classical Music & Musicals genre, he/she may also like the sub-genres of Classical Movie in movie genres. How to learn the shared knowledge through the hierarchical structures in cross-domain scenario is a challenging but practical problem.

To address the above problems, in this paper, we propose a novel Deep Low-rank Sparse Collective Factorization (DLSCF) framework to enhance the cross-domain recommendation. We work on the same scenario as TCF [2], [9], where the auxiliary data is binary rating, e.g., like/dislike, while the target is numerical rating, e.g., 5-star gradings. First, in each single layer LSCF, as shown in Figure 1, we jointly factorize the rating matrices R and $\tilde{R}$ in target and auxiliary domain into three parts: a user-specific latent feature matrix U, an item-specific latent feature matrix V and two rating patterns B and $\tilde{B}$ in target and auxiliary domain respectively. To effectively uncover the common information shared by them, we propose to adopt the low-rank sparse decomposition to further decompose B and $\tilde{B}$. As shown in Figure 1, we assume that two rating patterns B and $\tilde{B}$ share a common pattern D and also preserve domain-specific patterns E and $\tilde{E}$, respectively. Intuitively, the low-rank structure of D is to drive the similar rating patterns of users in one group (i.e., users have similar preferences). The column sparsity constraint is to learn the domain-specific pattern E and $\tilde{E}$. Although the scale of 0/1 ratings and 1-5 ratings are different, users' preferences towards categories do not change much.

Secondly, according to the hierarchical structures in real-word recommender, we further tri-factorize the rating pattern matrices to obtain the user and item latent category affiliation matrices through the multi-layer implicit hierarchical structures (Figure 2). The latent category affiliation matrix indicates the affiliation relation between latent categories and latent sub-categories.

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 [9] are more widely used in this problem. It updates model parameters only once after scanning the whole data. However, in real-world recommendation task, 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 memory. Second, batch based methods are hard to deal with upcoming new users and items. SGD based methods are widely used in large-scale data problem such as deep learning methods and recommendation methods as regularized singular value decomposition (RSVD) [20] and collective matrix factorization (CMF) [8]. They usually randomly update sample by sample. In this way, we address these two challenges through a SGD based optimization algorithm. Extensive experiments on five real-world datasets show that our proposed model with both batch and SGD based optimization algorithms outperforms the state-of-the-art methods for transferring auxiliary like/dislike domain to the numerical rating domain for heterogeneous recommendation.

This paper is an extension of our previous conference work [21]. There are three major differences compared to the conference version. First, we propose a 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. Compared to DLSCF-B, DLSCF-S is able to do recommendation for the continuously arrived new users. Second, we add experiments

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

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 performance of compared methods.

2 Related Work

Cross-domain recommendation systems exploit knowledge (e.g., user preferences) from auxiliary/source domains to facilitate the personalized recommendations in a target domain. Cross-domain recommendation consists of two main categories: one is to *aggregate knowledge* from source domains to provide recommendations in a target domain. The other is to *link or transfer knowledge* between source and target domains and facilitate the target domain recommendation.

Generally, the knowledge aggregation methods have three strategies: the first is to merge user preferences [22]; the second is to mediate several user modeling data in various recommendation systems, e.g., user similarities and user neighborhoods [23] and the third is to combine single domain recommendations. For example, combining the single domain rating probability distributions [24] and the rating estimations.

The knowledge linkage and transfer methods also have three strategies: the first is to link domains according common information, which includes but not limited to semantic networks, item attributes, inter-domain correlations and item attributes [25], [26]; the second is to share the use or item latent features which could link source and target domains knowledge [2], [9], [27], [28], [29], [30]; the third is to explicitly or implicitly recover the common rating patterns in source and target domain [12], [13], [14], [15].

Among them, Li et al. used a latent feature space to capture the comment rating patterns of user ratings [10], [11]. However, Geo et al. pointed out that in practice, sometimes there may not always be a common rating pattern across the related domains [12]. Moreover, the effectiveness of the common rating pattern may be degraded by the diversity and specific features among different domains. They proposed latent feature model in cluster-level to learn the shared part of rating patterns of user groups to enhance the cross-domain recommendation. However, such works may only consider the shared rating patterns while ignoring the shared information in users and items. How to transfer both the latent user and item feature and user rating patterns remains a problem and few work has explored it, especially in the scenario of transferring knowledge in binary ratings to numerical ratings.

Most recently, among knowledge linkage and transfer methods, cross-domain recommendation models based on latent factor [31], [32] have attracted a lot of attentions. Because the domain relations of different domains could be automatically discovered and exploited. In order to capture the heterogeneous structures of user feedbacks, Pan et al. investigated a series of models where the matrix factorization is regularized by transferring the source domain knowledge in forms of latent features to the target domain [2], [9], [30]. Specifically, they proposed a Transfer by Collective Factorization (TCF) framework for jointly modeling the target domain numerical ratings with the source domain binary ratings.

TCF is the most related work to ours. However, there are two main differences between our work and TCF. First, we collectively learn rating patterns through a further low-rank sparse decomposition, which is different from TCF that learns rating patterns of each domain separately. In this way, we could uncover more shared rating patterns across two domains and integrate the domain-specific rating patterns into a group sparsity. The other difference is that we explore the hierarchical structure in real-word recommendation system to uncover more effective knowledge.

3 The Proposed Algorithm

In this section, first we provide the problem definition of our cross-domain recommendation scenario. Second, we introduce the preliminary knowledge. Third, we present the single-layer building block Low-rank Sparse Collective Factorization (LSCF) of our framework. Fourth, we extend the one layer model into deep structure according to the hierarchical structure of real-world recommendation system.

3.1 Problem Definition

In the target data, assume that there are n users and m items and q observed ratings. $R = [r^{(u,i)}]_{n\times m} \in \{1,2,3,4,5,?\}^{n\times m}$ denotes a 5-star numerical rating matrix. Note that in real-world application, the observed rating values in R can be any real numbers. The question mark "?" denotes the unobserved value in $\{1,2,3,4,5\}$ scales. A mask matrix $Y = [y^{(u,i)}]_{n\times m} \in \{0,1\}^{n\times m}$ is observed or not. The auxiliary domain rating as $\tilde{R} = [\tilde{r}^{(u,i)}]_{n\times m} \in \{0,1,?\}^{n\times m}$ is in binary rating form and each observed rating is shown as 1 and 0 presenting whether the user likes or dislikes this item respectively. Similarly, the question mark denoted the missing observation, which is in $\{0,1\}$. The proportion of missing value in auxiliary domain is usually smaller than or equivalent to target domain. The mask matrix in source domain is denoted as $\tilde{Y} = [\tilde{y}^{(u,i)}]_{n\times m} \in \{0,1\}^{n\times m}$. Our task is to recover the missing values in R from the knowledge of observed rating values in R and $\tilde{R}$. We could also recover the missing values in $\tilde{R}$, but it is out of the scope of this task.

3.2 Preliminary

PMF: Probabilistic Matrix Factorization (PMF) [19] models the preference data via two latent feature matrices,

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

where the target numerical rating matrix R is factorized into a user-specific latent feature matrix $U \in \mathbb{R}^{n\times d}$ and an item-specific latent feature matrix $V \in \mathbb{R}^{m\times d}$. Once we have obtained the latent feature matrices, we can predict the rating located at (u,i) via $\hat{r}_{ui} = U_{(u\cdot)}V_{(i\cdot)}^{\top}$, where $U_{(u\cdot)} \in \mathbb{R}^{1\times d}$ and $V_{(i\cdot)} \in \mathbb{R}^{1\times d}$ are user u's and item i's latent feature vectors, respectively. However, PMF does not deal with the cross-domain recommendation problem, and doesn't make use of auxiliary data.

CMF: Collective Matrix Factorization (CMF) [8] uses item-side auxiliary data via sharing the same item-specific

latent features. We use matrix notations to illustrate its idea on knowledge sharing,

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

where the auxiliary binary rating matrix $\tilde{R}$ is decomposed into a user-specific latent feature matrix $W \in \mathbb{R}^{n\times d}$ and an item-specific latent feature matrix $V \in \mathbb{R}^{m\times d}$. The knowledge encoded in the item-specific latent feature matrix V is shared in two factorization systems, and the two user-specific latent feature matrices W and U aren't shared. However, CMF only makes use of item-side auxiliary data.

TCF: Different from PMF and CMF, TCF [2], [9] adopts matrix tri-factorization model and decomposes the rating matrix into user and item latent factors and ratting patterns. TCF constrains the user and item preference terms as the same for two domains and we achieve that:

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

where U and V are user and item latent factors, which is shared by source and target domain. The B and $\tilde{B}$ present the rating patterns in source and target domain respectively, which contains the domain specific knowledge. I is the identity matrix. $\|\cdot\|_F$ is the Frobenius norm. $\odot$ is the dot production. However, model (3) would neglect the shared information between rating pattern B and $\tilde{B}$, since B and $\tilde{B}$ are two bases of two datasets, which should share partial information [10], [11], [12].

3.3 Low-rank Sparse Collective Factorization (LSCF)

To effectively capture the common information shared by them, we propose to adopt low-rank sparse decomposition to further decompose B and $\tilde{B}$ [33], [34], [35], [36]. As shown in Figure 1, we assume the two rating patterns B and $\tilde{B}$ share a common patterns D and also preserve domain-specific patterns E and $\tilde{E}$ [37]. We assume D is constrained to be low-rank. It is enlighten by Geo et al.'s finding that cluster-level based user groups share the rating pattern in two domains [12]. Intuitively, the low-rank structure of D is to drive the rating patterns of users in one group (i.e., users have similar preferences) similar. We also assume that the domain-specific patterns E and $\tilde{E}$ are with column sparsity constraints. The column sparsity constraint is to learn the domain-specific pattern E and $\tilde{E}$.

To this end, we propose to minimize the following objective function f as:

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

where D is the common part and $\text{rank}(\cdot)$ is the rank operator of a matrix. E and $\tilde{E}$ are two sparse individual parts, which are constrained with $l_{2,1}$ to be column sparse.

Compared to TCF based methods [2], [9] which only assume the shared knowledge in user and item latent feature, LSCF also explores the shared knowledge in rating pattern by decomposing B and $\tilde{B}$ into a common low-rank matrix D and the domain-specific E and $\tilde{E}$. To this end, LSCF could uncover more shared information between B and $\tilde{B}$.

3.4 Deep Low-rank Sparse Collective Factorization (DLSCF)

Deep and hierarchical structures have demonstrated its effectiveness in many real-word applications, such as recommendation system [16], [38], image recognition and classification [39], [40]. In recommendation system, for example, in Netflix DVD rental page, movies are usually categorized as a hierarchical structure as genre $\rightarrow$ sub-genre $\rightarrow$ detailed-category. As shown in Figure 2 (a), in the genre layer, movies are divided to different kinds of genres, such as Horror, Romance and Sports & Fitness. Under Sports & Fitness genre, movies are further split into sub-genres, such as Motorsports, Snow & Ice Sports, and Water Sports. Under the sub-genre Motorsports, movies could be further categorized into several classes, such as Auto Racing, Car Culture and Extreme Motorsports. Users' preferences also have hierarchical structures as items. For example, under the Music & Musicals genre, a user may have more specifically preference of Classical Music.

To capture the hierarchical structures in real-world recommendation system, recently, exploring the hierarchical items or users latent factors attracted a lot of attentions [16], [17], [41]. For example, assuming there are p layers, in [16], they explored a hierarchical structure of $R \approx UV^\top$ through factorizing user latent feature U and item latent feature V:

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

They achieve this hierarchical structure through factorizing U as $U \approx U_1U_2\cdots U_{p-1}U_p$ and factorizing V as $V \approx V_1V_2\cdots V_{p-1}V_p$, where U_i and V_i ($i \in \{1, 2, \cdots, p\}$). Note that in their work, they assume U_i and V_i ($i \in \{1, 2, \cdots, p\}$) are non-negative.

However, few work has explored the hierarchical structures in the cross-domain scenario. Although the model (3) achieved attractive performance in one singe domain, in the cross-domain scenario, it is improper to only explore the hierarchical structures through factorizing U and V as [16] for two main reasons. First at all, similar to [2], [9], [30], we tri-factorize $R \approx UBV^\top$, and do not constrain U and V to be nonnegative. We constrain U and V using $UU^\top = I, VV^\top = I$. Thus, we cannot directly factorize $U \approx U_1U_2$ or $V \approx V_1V_2$. Another challenge is that our cross-domain scenario is not only to uncover the shared latent user and item feature in sub-category layers, but to uncover the sub-category rating patterns in two domains. To address these challenges, we propose the Deep Low-rank Sparse Collective Factorization (DLSCF) by extending the one layer LSCF model in this subsection.

First, we introduce how to factorize target domain rating matrix R and auxiliary domain rating matrix $\tilde{R}$. We introduce how to explore the hierarchical structures in target domain first. As shown in Figure 2 (b), in the first layer, we tri-factorize the rating matrix $R \in \mathbb{R}^{n\times m}$ to three matrices: $R \approx U_1B_1V_1^\top$, where $U_1 \in \mathbb{R}^{n\times d_1}$, $B_1 \in \mathbb{R}^{d_1\times d_1}$ and $V_1 \in \mathbb{R}^{m\times d_1}$. The item latent feature matrix V_1 indicates

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2894137, IEEE Transactions on Pattern Analysis and Machine Intelligence

the affiliation of m items to d_1 latent detailed categories. The user latent feature matrix U_1 indicates the affiliation of n users to d_1 latent detailed categories. Note that the number of categories in the first layer is much larger than the number of categories in deeper layers.

In the second layer, we keep U_1 and V_1 unchanged and further tri-factorize B_1 to $U_2B_2V_2^\top$, where $U_2 \in \mathbb{R}^{d_1\times d_2}$, $B_2 \in \mathbb{R}^{d_2\times d_2}$ and $V_2 \in \mathbb{R}^{d_1\times d_2}$. d_2 is the number of latent sub-categories. The latent category affiliation matrix of user and item latent features in the second layer are denoted as U_2 and V_2 respectively. They capture the affiliation relation between detailed categories and latent sub-categories in the first and second layer. Thus, we have:

$$R \approx U_1U_2B_2V_2^\top V_1^\top. \tag{6}$$

In the p-layer, as shown in Figure 2 (c) and (d), we keep $U_1, \cdots, U_{p-1}$ and $V_1, \cdots, V_{p-1}$, and tri-factorize B_{p-1} to $U_pB_pV_p^\top$. The category number in the p-th layer is denoted as d_p. Similar as U_2 and V_2 in the second layer, U_p and V_p capture the affiliation relation between categories in the $(p-1)$-th and p-th layer. And we have

$$R = U_1U_2\cdots U_pB_pV_p^\top\cdots V_2^\top V_1^\top. \tag{7}$$

In the auxiliary domain, we also factorize the rating matrix through progressively tri-factorizing $\tilde{B}_{i-1}$ into $\tilde{B}_{i-1} = U_i\tilde{B}_iV_i^\top$. In the p-th layer, we get the deep factorization of $\tilde{R}$ as:

$$\tilde{R} = U_1U_2\cdots U_p\tilde{B}_pV_p^\top\cdots V_2^\top V_1^\top. \tag{8}$$

In transfer learning scenario, we jointly learn user/item latent feature U_1/V_1 and the latent category affiliation matrix U_i and V_i $i \in \{2, 3, \cdots, p\}$, with constraint of sharing user/item latent feature and the latent category affiliation matrix in two domains. Also, similar to one layer model, we also assume that two ratting patterns B_p and $\tilde{B}_p$ share a part of common low-rank pattern D and also have domain-specific patterns E and $\tilde{E}$ respectively. To this end, we extend the one layer objective function in Eq. (25) into the following p layer deep structure objective function f_p as:

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

4 Batch based Solution (DLSCF-B)

We first provide batch based solutions of how to optimize the proposed DLSCF model and named as DLSCF-B. The optimization algorithm mainly consists of two steps. First, both the target numerical rating matrix and the auxiliary binary rating matrix are factorized, with the user-specific constraint and the item-specific constraint. Second, we learn the rating patterns in each domain with the low-rank constraint on the common part and group sparse constraints on the domain-specific part.

Since there are more than one variables to be optimized, it is hard to update them jointly [35], [36]. However, we could solve the variables one by one, meaning we optimize one variable by fixing others.

Updating U_i

We follow the same strategy in [9] to learn U_i and V_i. Note that the constraints of U_i and V_i are the same in [9] and ours. First, we remove the constraints in Eq. (9) since they do not contain U_i. Eq. (9) could be rewritten as:

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

where A_i, H_i and $\tilde{H}_i$, $1\le i\le p$, are defined as:

$$A_i = \begin{cases} U_1U_2\cdots U_{i-1}, & \text{if } i\neq 1, \\ I, & \text{if } i = 1. \end{cases} \tag{11}$$

And

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

And

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

We have the gradients as

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

Then the variable U_i can be learned via a gradient descent algorithm on the Grassmann manifold,

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

where we further define $t_1 = Y\odot(R-A_iU_iH_i)$, $\tilde{t}_1 = \tilde{Y}\odot(\tilde{R}-A_iU_i\tilde{H}_i)$, $t_2 = Y\odot(A_i\nabla_{U_i}H_i)$ and $\tilde{t}_2 = \tilde{Y}\odot(A_i\nabla_{U_i}\tilde{H}_i)$. Then, the learning rate $\gamma_{U_i} = \dfrac{-\mathrm{tr}(t_1^\top t_2)-\lambda\mathrm{tr}(\tilde{t}_1^\top\tilde{t}_2)}{\mathrm{tr}(t_2^\top t_2)+\lambda\mathrm{tr}(\tilde{t}_2^\top\tilde{t}_2)}$, where $\mathrm{tr}(\cdot)$ is the trace operator of a matrix.

Updating V_i

The update rule for V_i is similar as U_i. The other variables except V_i are fixed. The optimization problem for V_i could be rewriten as follows, when removing terms that are irrelevant to V_i:

$$f_p = \frac{1}{2}\|Y\odot(R-C_iV_iF_i)\|_F^2+\frac{\lambda}{2}||\tilde{Y}\odot(\tilde{R}-\tilde{C}_iV_iF_i)|_F^2, \tag{16}$$

where F_i, C_i and $\tilde{C}_i$, $1\le i\le p$, are defined as:

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

And

$$C_i = \begin{cases} U_1\cdots U_pB_pV_p^\top\cdots V_{i+1}^\top, & \text{if } i\neq p, \\ U_i\cdots U_pB_p, & \text{if } i = p. \end{cases} \tag{18}$$

And

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

We have the gradients as

$$\frac{\partial f_p}{\partial V_i} = C_i^\top(Y\odot(C_iV_iF_i-R))F_i^\top+\lambda C_i^\top(\tilde{Y}\odot(C_iV_i\tilde{F}_i-\tilde{R}))\tilde{F}_i^\top. \tag{20}$$

Then the variable V_i can be also learned via the gradient descent algorithm on the Grassmann manifold,

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

where γ_{V_i} could be learned in the same way as γ_{U_i}.

Updating B_p and $\tilde{B}_p$

The rule for updating B_p and $\tilde{B}_p$ are similar. Take updating B_p as an example. We remove other irrelevant parts to B_p as:

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

where $U = U_1 \cdots U_p$ and $V = V_1 \cdots V_p$. B_p can be estimated exactly through a corresponding least square SVM problem. We define $\varrho = \mathrm{vec}(B_p) = [B_{1,1}^\top, \cdots, B_{1,d}^\top]^\top \in \mathbb{R}^{d^2 \times 1}$ is a big vector concatenated from the columns of matrix B_p, and $\rho = \mathrm{vec}(D + E) = \mathrm{vec}(\bar{D}) = [\bar{D}_1^\top, \cdots, \bar{D}_d^\top]^\top \in \mathbb{R}^{d^2 \times 1}$. The instances can be constructed as $\{(\vartheta_{u,i}, r_{u,i})\}$ with $y_{u,i} = 1$. $\theta_{u,i} = \mathrm{vec}(U^{(u)^\top} V^{(i)}) \in \mathbb{R}^{d^2 \times 1}$, where $U^{(u)}$ is the u-th row of U and $V^{(i)}$ is the i-th row of V. Hence, we obtain the following least-square SVM problem:

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

where I is the identity matrix. Note that we could apply the existing off-the-shelf tools to solve B_p or ρ efficiently by considering it as a linear compact operator. Similarly $\tilde{B}_p$ can be solved in this way.

Updating D:

We remove other irrelevant parts to D as:

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

where the rank minimization problem is replaced with minimizing the nuclear norm problem [33], [35], [36]. And problem (24) can be effectively solved by the singular value thresholding (SVT) operator [42].

Updating E and $\tilde{E}$

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

which can be solved by the shrinkage operator [43]. While $\tilde{E}$ can also be addressed in the same way.

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

4.1 Complexity Analysis of Batch based Solution

There are three main time-consuming operators in our algorithm: (1) updating U_i and V_i ($i = 1, \cdots, p$), (2) updating B_p and $\tilde{B}_p$, (3) updating D. The time complexity of the first two steps is similar to [9], but [9] only focused on the situation of one layer structure. In our deep structure, we sum up the time complexity of updating U_i and V_i, $i \in \{1, ..., p\}$. The complexity of the first two steps is $O\big(K(\max(q, \tilde{q})d_1^3 + \sum_{i=2}^{p} d_{i-1}d_i + d_p^6)\big)$. K is denoted as the number of iterations which needed by the algorithm to converge. $q, \tilde{q}(q, \tilde{q} > n, m)$ is the number of observed ratings in R and $\tilde{R}$ respectively. n is number of all the users and m is the number of all the items. d_i is the number of latent features. The third step is time-consuming to the full SVD operator on each iterative optimization for nuclear norm [33], [36]. Since $D \in \mathbb{R}^{d_p \times d_p}$, the SVD costs $O(Kd_p^3)$ for K iterations. To sum up, our algorithm takes about $O\big(K(\max(q, \tilde{q})d_1^3 + \sum_{i=2}^{p} d_{i-1}d_i + d_p^6 + d_p^3)\big)$. As we can observe, the first two steps are much more time-consuming compared to our low-rank part. Meanwhile, our low-rank constraint really helps a lot in the predictions, which will be shown in the experiments.

The time complexity of compared methods are: PMF is $O(Kqd^2 + K\max(n, m)d^3)$, cPMF is $O(Kq\tilde{c}d^2 + K\max(n, m)d^3)$ [19] and TCF is $O(K\max(q, \tilde{q})d^3 + Kd^6)$ [9]. The time complexity of our proposed method is comparable to TCF. Although the time complexity of ours will be higher especially when the hierarchical structure is very deep, from our analysis, two layers would be enough to obtain the state-of-the-art performance.

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

Batch based algorithms such as PMF [19] and TCF [9] update model parameters only once after scanning the whole data, which might not be applicable for real world large data for two reasons. First, updating all the user and item factors may take a lot memory for large number of users and items. Second, batch based methods are hard to deal with upcoming new users and items.

Stochastic methods such as regularized singular value decomposition (RSVD) [20] and collective matrix factorization (CMF) [8] are empirically much more efficient than alternative batch-style algorithms. However, the prediction accuracy of RSVD and CMF might not be adequate when compared with batch based methods such as TCF, especially when the users' feedback are heterogeneous. There are also some efficient distributed or online collaborative filtering algorithms such as distributed stochastic gradient descent [44] and online multitask collaborative filtering [45], but they're designed for homogeneous user feedback instead of the heterogenous ones studied in this article.

To address these challenges, in this section, we introduce the Stochastic Gradient Descent (SGD) 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 would be huge matrices when R and $\tilde{R}$ are large-scale. Thus, we only update U_1 and V_1 in a Stochastic Gradient Descent method. And further, we release the orthogonal constraints on U_1 and V_1 to a l_2 regularizer on them.

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

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

In Eq. (26), we could see that only the size of U_1 and V_1 are effected by the number of users and items and could be very large. The sizes of U_i, V_i, $i \in \{2,\cdots,p\}$, B_p, $\tilde{B}_p$, D, E, $\tilde{E}$ are predefined and are not very large. 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 methods.

Updating U_1:

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

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

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

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 $G^{(u)} = \sum_{v=1}^{m}(y^{(uv)}AA^\top + \lambda\tilde{y}^{(uv)}\tilde{A}\tilde{A}^\top) + \alpha^{(u)}\sum_{v=1}^{m}(y^{(uv)} + \lambda\tilde{y}^{(uv)})I$, and $b^{(u)} = \sum_{v=1}^{m}(y^{(uv)}r^{(uv)}A^\top + \lambda\tilde{y}^{(uv)}\tilde{r}^{(uv)}\tilde{A}^\top)$.

We can see that $U_1^{(u\cdot)}$ in Eq. (28) is independent of all other users' latent features given B and V, thus we can obtain the user-specific latent feature matrix U_1 analytically.

Updating V_1:

Similarly, given B, $U_1,...,U_p$, $V_2,...,V_p$ the latent feature vector $V_1^{(v\cdot)}$ of each item v can be estimated in a closed form. Let $U_1^{(u\cdot)}...U_p^{(u\cdot)}B_pV_p^{(v\cdot)\top}...V_2^{(v\cdot)\top} = A$ and let $U_1^{(u\cdot)}...U_p^{(u\cdot)}\tilde{B}_pV_p^{(v\cdot)\top}...V_2^{(v\cdot)\top} = \tilde{A}$, and thus the whole item-specific latent feature matrix V_1 can be obtained analytically.

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

where $G^{(v)} = \sum_{u=1}^{n}(y^{(uv)}A^\top A + \lambda\tilde{y}^{(uv)}\tilde{A}^\top\tilde{A}) + \alpha_v\sum_{u=1}^{n}(y^{(uv)} + \lambda\tilde{y}^{(uv)})I$, 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 our 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 could avoid SVD operation on 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 also solve the variables one by one, which is optimizing one variable by fixing others. For each iteration, instead of updating U_1 or V_1 as a whole matrix, we only update one row $U_1^{(u\cdot)}$ 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 the describe of the datasets, evaluation metrics and comparison methods. The we present the evaluation results on both accuracy and memory aspect. Then, we evaluate the influence of parameters.

6.1 Datasets and Settings

We evaluate our proposed method on five benchmark recommendation datasets collected from real-world, (1) Moviepilot, (2) Netflix, (3) Flixter, (4) MovieLens10M and (5) MovieLens20M. Since these is no existing dataset designed for the same scenario as ours (i.e., one numerical and one binary), we follow existing works [9], [46] 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 contain more than 4.5×10^6 ratings, given by more than 1.0×10^5 users on around 2.5×10^4 movies. We first randomly extract a $2,000 \times 2,000$ dense rating matrix R from the Moviepilot data. We randomly split R into training and test sets, T_R, T_E, with 50% ratings, respectively. T_R, $T_E \subset (u, i, r_{u,i}) \in N \times N \times [1,5] | 1 \le u \le n, 1 \le i \le m$. T_E is kept unchanged, while different (average) number of observed ratings for each user, 4, 8, 12, 16, are randomly sampled from T_R for training, with different sparsity $(u, i, y_{u,i}, y_{u,i}/n/m)$ levels of 0.2%, 0.4%, 0.6% and 0.8% correspondingly. The sparsity means the average rated items by user dividing the number of all the items. For example, the sparsity 0.2% means one user has rated 0.2% items. 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. We randomly pick 40 observed ratings on average from T_R for each user to construct the auxiliary data matrix $\tilde{R}$. Thus, the sparsity of auxiliary data is 2%. We extracted the second dataset Netflix in the same way as oviepilot. The size of rating matrix R in Netflix is $5,000 \times 5,000$. By default, we apply the sparsity levels at 0.8% following [46] for these two datasets.

To simulate heterogeneous auxiliary and target data, we adopt a pre-processing approach on $\tilde{R}$, by relabeling ratings

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2894137, IEEE Transactions on Pattern Analysis and Machine Intelligence

Table 1
Previous usage and statistic of datasets Netflix, MoviePilot, Flixter, MovieLens10M and MovieLens20M. 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 rating), auxiliary(#aux rating), and test data(#test rating). The sparsity means the average rated items by user dividing the number of all the items.

	previous usage	#user	#item	# target rating	target sparsity	#aux rating	aux sparsity	#test rating	test sparsity
Netflix	both	5,000	5,000	45,000	$\leq 1\%$	500,000	2%	2,824,204	11.27%
MoviePilot	Batch	2,000	2,000	30,000	$\leq 1\%$	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%

with value $r_{u,i} \leq 3$ in $\tilde{R}$ as 0 (dislike), and then ratings with value rui ≥ 3 as 1 (like). The overlap between $\tilde{R}$ and $R(u, i, y_{u,i}, \tilde{y}_{u,i}/n/m)$ is 0.026%, 0.062%, 0.096% and 0.13% correspondingly. The overlap between $\tilde{R}$ and R is 0.035%, 0.070%, 0.10% and 0.14% respectively. 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 extracted the third dataset from Flixter[2] following the same setting in [46]. This data contain 8.2×10^6 ratings given by 1.5×10^5 users on 4.9×10^4 products. The range of ratings is {0.5, 1, 1.5,...,5}. The data contains five copies of target, auxiliary, and test data. For each copy of auxiliary data, we convert ratings smaller than 4 to "dislike" and ratings larger than 4 to "like", to simulate the binary feedback.

We extracted the fourth dataset from MovieLens10M[3] in the same way as [46]. 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 preprocess the MovieLens10M rating data similar as that of the Flixter data to generate five copies of target, auxiliary, and test data.

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 [9], [21] evaluate the performance on Netflix and MoviePilot. SGD based methods [46] evaluate the performance only on Netflix, Filxter and MovieLens10M dataset. MovieLens20M has not been applied on this problem yet.

6.2 Evaluation Metrics

We follow [9], [21], [46] and adopt the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) as evaluation metrics as:

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

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

where $r_{u,i}$ and $\hat{r}_{u,i}$ are the ground truth and predicted ratings, respectively. And $|T_E|$ is the number of test ratings.

Same as [9], in all experiments, the required numbers of observed ratings are generated from T_R randomly, and averaged by 3 trials.

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.

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

6.3.1 Batch based Methods

We compare batch based DLSCF with four batch based transfer learning methods:

cPMF [19]: Constrained Probabilistic Matrix Factorization (PMF). PMF models the preference data via two latent feature matrices, user-specific latent feature matrix and item-specific latent feature matrix, as shown in Eq. (1). [9] integrated the auxiliary data of PMF in their experiments.

CMF-link [8]: CMF with logistic link function. CMF [8] is proposed for jointly factorizing two matrices with the constraints of sharing item-specific latent features, as shown in 2. However, in this problem setting, both users and items are aligned. To alleviate the data heterogeneity in CMF, [9] embed a logistic link function in the auxiliary data matrix factorization in their experiments.

TCF(CMTF), **TCF(CSVD)** [2], [9]: Transfer by Collective Factorization (TCF) constructs a shared latent space collectively and learns the data-dependent effect separately. It is able to capture the data dependent effect when sharing the data-independent knowledge. Two variants 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 [9]. We strictly follow the settings of [9].

In order to evaluate the effectiveness of the hierarchical structure and the low-rank sparse decomposition, we also compare with two baseline methods: DCSVD and LSCF. We named 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 [21] (conference): We keep the deep structure of DLSCF, but remove the low-rank sparse constraint and set $\beta_1 = \beta_2 = 0$. The objective function is shown as following. This method could also be regarded as a deep version of CSVD.

LSCF-B [21] (conference): The method is the one layer structure of DLSCF as introduced in Section 3.3.

DLSCF-B [21] (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 study the effectiveness of our SGD based DLSCF, DLSCF-S, with some state-of-the-art methods, including regularized singular value decomposition (RSVD) [20], collective matrix factorization (CMF) [8] and interaction-rich transfer by collective factorization (iTCF) [46].

RSVD [20]: RSVD approximates an observed target grade score via learning some latent variables of the corresponding user and item, which works well for data of grade scores.

CMF [8]: CMF extends RSVD via sharing items' latent variables for the target grade scores and the auxiliary binary ratings.

iTCF [46]: iTCF further extends CMF via introducing interactions between users' latent variables, which was reported to be more accurate than CMF.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2894137, IEEE Transactions on Pattern Analysis and Machine Intelligence

Table 2
Compared performance of both batch and SGD algorithms on MoviePilot dataset. Bold and underline denote the best and second best performance.

Algorithm		MAE	RMSE	Memory
batch	cPMF [19]	0.7462	0.9599	1.92
	CMF-link [8]	0.6905	0.9072	2.40
	TCF(CMTF) [9]	0.6776	0.8875	1.92
	TCF(CSVD) [9]	0.6612	0.8744	2.40
	DCSVD-B [21](conference)	0.6590	0.8733	2.88
	LSCF-B [21](conference)	0.6530	0.8644	2.40
	DLSCF-B [21](conference)	0.6504	0.8626	2.88
SGD	RSVD [20]	0.7081	0.9043	1.76
	CMF [8]	0.6843	0.8954	1.76
	iTCF [46]	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

Table 3
Compared performance of both batch and SGD algorithms on Netflix dataset. Bold and underline denote the best and second best performance.

Algorithm		MAE	RMSE	Memory
batch	cPMF [19]	0.7642	0.9691	2.50
	CMF-link [8]	0.7295	0.9277	2.78
	TCF(CMTF) [9]	0.6962	0.8884	2.50
	TCF(CSVD) [9]	0.6877	0.8809	2.78
	DCSVD-B [21](conference)	0.6862	0.8793	2.78
	LSCF-B [21](conference)	0.6870	0.8780	2.78
	DLSCF-B [21](conference)	0.6854	0.8775	2.83
SGD	RSVD [20]	0.7236	0.9201	1.54
	CMF [8]	0.7054	0.9020	1.54
	iTCF [46]	0.7014	0.8966	1.54
	DCSVD-S (baseline1)	0.6912	0.8936	1.68
	LSCF-S (baseline2)	<u>0.6838</u>	<u>0.8762</u>	1.54
	DLSCF-S (ours)	**0.6828**	**0.8749**	1.68

Table 4
Compared performance of both batch and SGD algorithms on Flixter dataset. Bold and underline denote the best and second best performance.

Algorithm		MAE	RMSE	Memory
batch	cPMF [19]	0.6479	0.8768	4.69
	CMF-link [8]	0.6456	0.8745	4.69
	TCF(CMTF) [9]	0.6394	0.8696	4.99
	TCF(CSVD) [9]	<u>0.6347</u>	0.8632	5.32
	DCSVD-B [21](conference)	0.6342	0.8624	5.51
	LSCF-B [21](conference)	0.6348	0.8625	5.32
	DLSCF-B [21](conference)	**0.6331**	<u>0.8621</u>	5.51
SGD	RSVD [20]	0.6561	0.8814	3.94
	CMF [8]	0.6423	0.8710	3.91
	iTCF [46]	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)	<u>0.6347</u>	**0.8619**	3.95

We also implement SGD based DCSVD-S and LSCF-S for two reasons. First, we evaluate the effectiveness of the hierarchical structure and the low-rank sparse decomposition in SGD based optimization algorithm. Second, we compare the 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 and set $\beta_1 = \beta_2 = 0$ 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.

Table 5
Compared performance of both batch and SGD algorithms on MovieLens10M dataset. Bold and underline denote the best and second best performance.

Algorithm		MAE	RMSE	Memory
batch	cPMF [19]	0.6531	0.8354	4.75
	CMF-link [8]	0.6413	0.8326	4.75
	TCF(CMTF) [9]	0.6241	0.8213	4.75
	TCF(CSVD) [9]	0.6217	0.8153	4.99
	DCSVD-B [21](conference)	0.6212	0.8212	5.51
	LSCF-B [21](conference)	0.6154	0.8123	4.90
	DLSCF-B [21](conference)	**0.6041**	0.8094	5.51
SGD	RSVD [20]	0.6438	0.8364	3.24
	CMF [8]	0.6334	0.8273	3.08
	iTCF [46]	0.6197	<u>0.8091</u>	3.11
	DCSVD-S (baseline1)	0.6159	0.8133	3.15
	LSCF-S (baseline2)	0.6133	0.8093	3.08
	DLSCF-S (ours)	<u>0.6121</u>	**0.8088**	3.15

Table 6
Compared performance of both batch and SGD algorithms on MovieLens20M dataset. Bold and underline denote the best and second best performance.

Algorithm		MAE	RMSE	Memory
batch	cPMF [19]	0.6532	0.9039	5.52
	CMF-link [8]	0.6511	0.8932	5.52
	TCF(CMTF) [9]	0.6432	0.8812	5.13
	TCF(CSVD) [9]	0.6445	0.8715	7.09
	DCSVD-B [21](conference)	0.6364	0.8624	7.09
	LSCF-B [21](conference)	0.6345	0.8594	7.09
	DLSCF-B [21](conference)	<u>0.6315</u>	0.8535	7.09
SGD	RSVD [20]	0.6572	0.9064	4.34
	CMF [8]	0.6492	0.8717	4.51
	iTCF [46]	0.6415	0.8627	4.51
	DCSVD-S (baseline1)	0.6382	0.8572	4.59
	LSCF-S (baseline2)	0.6331	<u>0.8511</u>	4.53
	DLSCF-S (ours)	**0.6260**	**0.8419**	4.59

Table 7
Compared performance of transferring numerical to numerical of both batch and SGD algorithms on Netflix dataset. Bold and underline denote the best and second best performance.

Algorithm		MAE	RMSE	Memory
batch	cPMF [19]	0.7094	0.9185	2.65
	CMF-link [8]	0.7084	0.9034	2.83
	TCF(CMTF) [9]	0.6946	0.8945	2.78
	TCF(CSVD) [9]	0.6877	0.8808	2.83
	DCSVD-B [21](conference)	0.6854	0.8793	2.83
	LSCF-B [21](conference)	0.6833	0.8750	2.83
	DLSCF-B [21](conference)	<u>0.6714</u>	**0.8638**	2.97
SGD	RSVD [20]	0.7167	0.9157	1.54
	CMF [8]	0.7037	0.8985	1.54
	iTCF [46]	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

6.4 Experimental Results

The results on test data 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.

Different trade-off parameters $\lambda \in [0.1, 5]$, $\beta_1 \in [0, 1]$, $\beta_2 \in [0, 1]$ are tried. For DCSVD and DLSCF, different layers $p \in \{1, 2, 3, 4, 5\}$ are tried. In DLSCF-S, we have tried α_u and α_v from 0.1 to 5, and set $\alpha_u = 1$ and $\alpha_v = 1$. The parameters and convergence condition are set through cross-validation, which randomly samples n ratings from training data T_R. We found DLSCF usually achieves better performance when p=2 or 3, which agrees with the real-world category structure. From Tables 2 to 6, we could observe that:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2894137, IEEE Transactions on Pattern Analysis and Machine Intelligence

Table 8
p values of t-test of comparison methods under RMSE metric.

Dataset	MoviePilot	Netflix	Flixter	MovieLens10	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}

Table 9
Discussion of performance of compared methods under different sparsity levels of target domain on Netflix. Bold and underline denote the best and second best performance.

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 [47]	0.8055	1.0202	0.7846	0.9906	0.7757	0.9798	0.7711	0.9741
	OptSpace [48]	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 [8]	0.7994	1.0204	0.7508	0.9552	0.7365	0.9369	0.7295	0.9277
	TCF(CMTF) [9]	0.7589	0.9653	0.7195	0.9171	0.7031	0.8971	0.6962	0.8884
	TCF(CSVD) [9]	0.7405	0.9502	0.7080	0.9074	0.6948	0.8903	0.6877	0.8809
	DCSVD-B([21](conference)	0.7379	0.9466	0.7076	0.9067	0.6942	0.8892	0.6862	0.8793
	LSCF-B([21](conference)	0.7380	0.9470	0.7069	0.9062	0.6937	0.8890	0.6870	0.8780
	DLSCF-B([21](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 [8]	0.7791	0.9735	0.7396	0.9472	0.7456	0.9232	0.7054	0.9020
	iTCF [46]	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	<u>0.9442</u>	0.7074	<u>0.9047</u>	<u>0.6922</u>	<u>0.8866</u>	<u>0.6838</u>	<u>0.8762</u>
	DLSCF-S (ours)	<u>0.7376</u>	**0.9413**	<u>0.7067</u>	**0.9028**	**0.6908**	**0.8854**	**0.6828**	**0.8749**

Table 10
Discussion of performance of compared methods under different sparsity levels of target domain on MovieLens10M. Bold and underline denote the best and second best performance.

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 [47]	0.7943	0.9225	0.7143	0.8953	0.6825	0.8621	0.6835	0.8434
	OptSpace [48]	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 [8]	0.7456	0.9054	0.6843	0.8685	0.6411	0.8321	0.6325	0.8236
	TCF(CMTF) [9]	0.7386	0.8965	0.6754	0.8574	0.6224	0.8184	0.6145	0.8143
	TCF(CSVD) [9]	0.7335	0.8964	0.6743	0.8534	0.6203	0.8151	0.6136	0.8147
	DCSVD-B([21](conference)	0.7254	0.8843	0.6636	0.8343	0.6142	0.8146	0.6014	0.8122
	LSCF-B([21](conference)	0.7012	0.8662	0.6546	0.8435	0.6121	0.8103	0.6032	0.8137
	DLSCF-B([21](conference)	**0.6886**	<u>0.8485</u>	<u>0.6453</u>	**0.8224**	**0.6011**	<u>0.8065</u>	<u>0.5972</u>	**0.7911**
transfer(SGD)	RSVD [20]	0.7496	0.9183	0.6864	0.8754	0.6430	0.8315	0.6267	0.8212
	CMF [8]	0.7278	0.9036	0.6785	0.8657	0.6258	0.8187	0.6193	0.8087
	iTCF [46]	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)	<u>0.6925</u>	**0.8446**	**0.6356**	<u>0.8256</u>	<u>0.6035</u>	**0.8012**	**0.5943**	<u>0.7962</u>

6.4.1 Results of accuracy

(1) Among all the batch based methods, our conference DLSCF-B achieves the lowest error under MAE and RMSE metric, including the state-of-the-art batch based method TCF(CSVD) and our two baseline methods DCSVD-B and LSCF-B.

(2) Among all the SGD based methods, our proposed method DLSCF-S in this journal extension achieves the lowest error, comparing to the state-of-the-art SGD based method iTCF and our two baseline methods DCSVD-S and LSCF-S.

We conduct t-test to evaluate whether the improvement is significant. The lower the significance level, which is presented with p value, the more confident the difference between comparison methods. Table 8 presents p values of t-test between our methods and comparison methods under all the datasets. We run each method for 5 times. Table 8 demonstrates the significance of the improvements (i.e., p <0.05) of both DLSCF-B and DLSCF-S.

(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,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2894137, IEEE Transactions on Pattern Analysis and Machine Intelligence

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) To evaluate the effectiveness of the low-rank sparsity constraint, we compare DLSCF with DCSVD and LSCF with CSVD in both batch and SGD based solution. Both DLSCF and DCSVD adopt the multi-layer structure, while both LSCF and CSVD adopt one-layer structure. Thus, the difference is mainly caused by the ways of sharing information between two domains. DCSVD and CSVD assume that each domain has a specific rating pattern matrix, while DLSCF and LSCF apply the low-rank sparse constraint on the rating patterns of two domains. In both SGD and batch based methods, DLSCF outperforms DCSVD and LSCF outperforms CSVD, which demonstrates the effectiveness of the proposed low-rank sparsity constraint.

(5) To evaluate the effectiveness of the multi-layer structure, we also compare DLSCF with LSCF and DCSVD with CSVD in both batch and SGD based solution. DLSCF is the multi-layer version LSCF and DCSVD is the multi-layer version CSVD. In both SGD and batch based methods, DLSCF outperforms LSCF and DCSVD outperforms CSVD. Thus, we could safely consider that the multi-layer structure plays a major role of achieving better performance.

In order to evaluate the effectiveness of our method on the data which are not simulated, we have added another setting that both target and auxiliary domains are not re-scaled, meaning both domains are numerical. Actually, our model not only could be used 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 (5 stars). In this way, the axillary domain is not simulated. We show the experimental results in Table 7.

From Table 7, we can see that when both source and auxiliary domain data are numerical and not simulated, our batch or SGD 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% 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 are not large.

When we compare DLSCF-S with DLSCF-B, DLSCF-S takes less memory (about 60%) 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 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%) and make the method computable.

6.5 Discussion

6.5.1 Discussion of sparsity

We follow [9], [21] to discuss the effectiveness of solving data sparsity problem of proposed DLSCF and comparison methods.

Firstly, we randomly sample training set from T_R, with sparsity levels at 0.2%, 0.4%, 0.6% and 0.8% in target domain respectively. In Table 9 and 10, we report the performance of non-transfer learning methods, batch based transfer learning methods and SGD based transfer learning methods, under dataset Netflix and MovieLens10M. Three non-transfer learning methods as [9]: PMF [19], SVD [47] and OptSpace [48] are included to evaluate the performance of transfer learning based methods. The sparsity of auxiliary data $\tilde{R}$ is shown as Table 1, which is 2% of Netflix and 0.52% of MovieLens10M dataset. In Table 9, the results of non-transfer learning methods, batch based transfer learning methods are copied from [9], [21].

First, from Tables 9 and 10, we could 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%, DLSCF-S outperforms DLSCF-B under both MAE and RMSE. Under sparsity level 0.2% and 0.4%, 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 methods outperform the non-transfer ones. The transfer learning method CMTF-link already works better than PMF, SVD and OptSpace. It demonstrates that the transfer learning method is effective to solve the sparsity problem in this task.

We also discuss the effectiveness of DLSCF in different sparsity levels in auxiliary domain data in at 1%, 2% and 3%. Figure 5 shows the MAE and RMSE of DLSCF-B on MoviePilot as an example. We have the same observation as [16], which shows the effectiveness of method of selective transferring via noise reduction.

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

In this section, we discuss the impact of settings of parameter λ, β_1, β_2 in DLSCF-B. Figure 3 shows an example in MoviePilot dataset when the sparsity level of target domain data is 0.6 %, p=1, and d_1=10.

In Figure 3 (a), we could notice that RMSE achieves the lowest result when $\lambda = 0.3$. Around 0.3, when $\lambda = 0.35$, RMSE is 0.8645. When $\lambda = 0.25$, RMSE is 0.8648. MAE achieves the lowest result when $\lambda = 0.25$, 0.3 and 0.35. Both RMSE and MAE increase gradually when λ is far away from 0.3. When $\lambda = 0.4$, RMSE is 0.865 and MAE is 0.6546. When $\lambda = 0.2$, RMSE is 0.8658 and MAE is 0.6558. When $\lambda = 0$, meaning $\|\tilde{Y} \odot (\tilde{R} - U\tilde{B}V^\top)\|_F^2$ term looses the impact, RMSE is 0.9003 and MAE is 0.6839.

In Figure 3 (b), we could see 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 $\beta_1 = 5 \times 10^{-5}$. When β_1 is far away from 0.005, RMSE and MAE increase gradually.

In Figure 3 (c), we could see that both RMSE and MAE achieve the lowest result when $\beta_2 = 0.005$, and we have RMSE = 0.8642 and MAE = 0.6530. Both RMSE and MAE increase sharply when β_2 is bigger than 0.02.

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

In this section, we discuss the impact of settings of parameter λ, β_1, β_2 of DLSCF-S. Figure 4 shows an example in MoviePilot dataset when the sparsity level of target domain data is 0.6 %, p=1, and d_1=10. We have tried α_u and α_v from 0.1 to 5, and fixed $\alpha_u = 1$ and $\alpha_v = 1$.

In Figure 4 (a), we notice that both MAE and RMSE achieve the lower error when λ is from 0.3 to 1, which shows that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2894137, IEEE Transactions on Pattern Analysis and Machine Intelligence

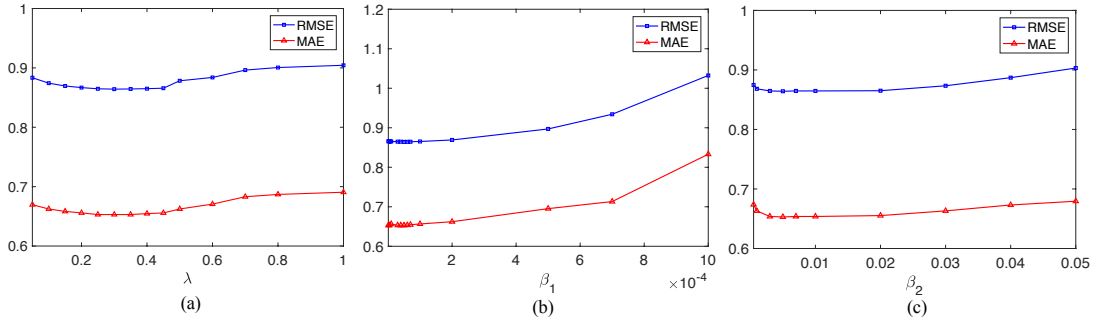


Figure 3. Parameter analysis of DLSCF-B. In figure (a), we fix $\beta_1 = 0.005$, $\beta_2 = 0.00005$ and show the RMSE and MAE when λ from 0.05 to 1. In figure (b), we fix $\lambda = 0.3$, $\beta_2 = 0.00005$, and show the RMSE and MAE when β_1 from 1×10^{-6} to 0.001. In figure (c), we fix $\lambda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

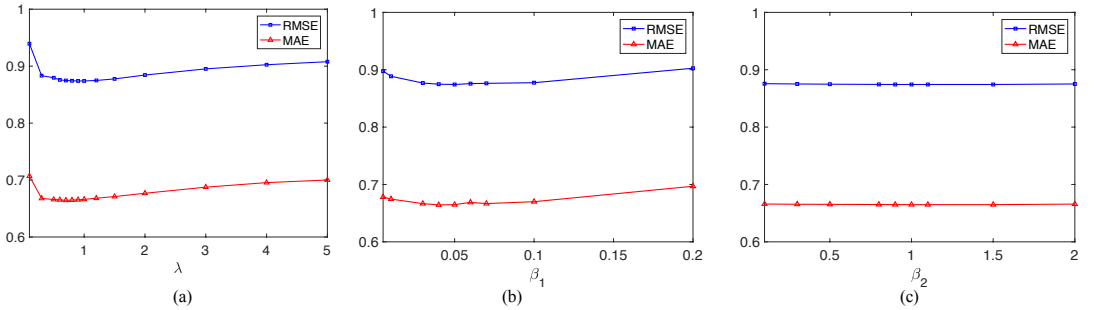


Figure 4. Parameter analysis of DLSCF-S. In figure (a), we fix $\beta_1 = 0.05$, $\beta_2 = 1$ and show the RMSE and MAE when λ from 0.1 to 5. In figure (b), we fix $\lambda = 0.8$, $\beta_2 = 0.05$, and show the RMSE and MAE when β_1 from 0.005 to 0.2. In figure (c), we fix $\lambda = 0.8$, $\beta_3 = 1$, and show the RMSE and MAE when β_2 from 0.1 to 2.

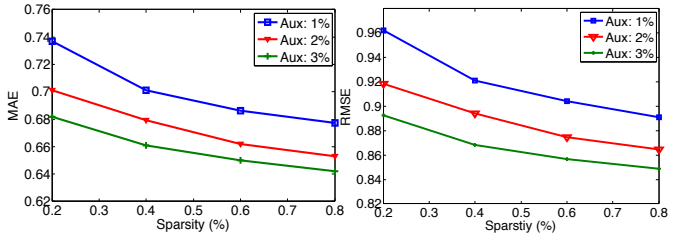


Figure 5. Prediction error of DLSCF-B on MoviePilot at different sparsity levels with auxiliary domain data.

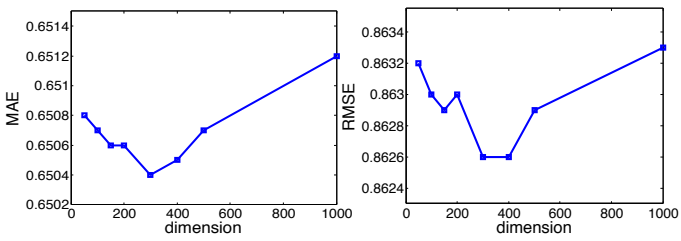
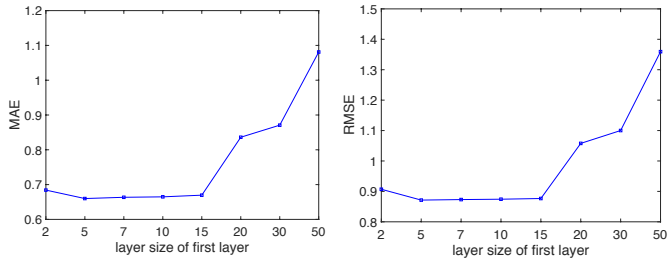
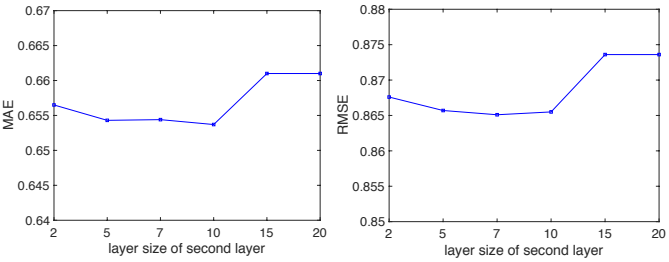


Figure 6. Prediction error of DLSCF-B on MoviePilot at different dimensions of layer.



(a) Prediction error in the first layer



(b) Prediction error in the second layer

Figure 7. Prediction error of DLSCF-S on MoviePilot at different dimensions of first and second layer.

the performance is not sensitive to the parameter. 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.

In Figure 4 (b), we could witness that both RMSE and MAE achieve the lower error when λ is from 0.03 to 0.07. RMSE and MAE increase gradually when β_1 is far way from 0.05.

In Figure 4 (c), we could witness that both RMSE and MAE are not sensitive to β_2. We achieve the lowest RMSE = 0.8743 and MAE = 0.6648 when $\beta_2 = 1$. RMSE and MAE increase gradually when β_2 is far way from 1. 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 settings of layer size. Figure 6 shows the performance under settings of different layer sizes of DLSCF in MoviePilot dataset at sparsity

level 0.8%. We set the number of layers p at 2 and discuss the impact of layer size d_1 by fixing the number of d_2 as [16]. We set $d_2 = 10$ and vary the value of d_1 within $\{50, 100, 150, 200, 300, 400, 500, 1000\}$. As shown in the Figure 6, under both MAE and RMSE, DLSCF-B achieves better performance when d_1 is around 200 to 400.

6.5.5 Discussion of layer size in DLSCF-S

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

We start from discussing the layer size in one-layer DLSCF-S model (same as LSCF-S) in Figure 7 (a). Figure 7 (a) shows the performance under settings of different layer sizes of DLSCF in MoviePilot dataset at sparsity level 0.6%. 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 under both MAE and RMSE, our method achieves better performance when d_1 is around 5 to 10. The lowest MAE is 0.6599 and the lowest RMSE is 0.8714 when $d_1 = 5$.

Second, we discuss the layer size in two-layer DLSCF-S model ($p = 1$). We fix the $d_1 = 30$ and vary the value of d_2 within $\{2, 5, 7, 10, 15, 20\}$. When we only use one layer model and set $d_1 = 30$ as shown in Figure 7 (a), MAE = 0.8361, RMSE = 1.0578. 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 Figure 7 (a) and (b), all the MAE and RMSE in (b) are lower than in (a), which shows the effectiveness of the multilayer structure.

7 Conclusions

In this paper, we presented a novel transfer learning framework for heterogeneous recommendation, named as Deep Low-rank Sparse Collective Factorization (DLSCF), to transfer knowledge from binary ratings data. With a low-rank constraint on the common part and a group sparsity constraint on the domain-specific part, we could capture more shared patterns across two domains to transfer more knowledge from auxiliary domain to target domain. In addition, according to the hierarchical structure of the real-world recommendation system, we further explored the deep structure based on the one layer Low-rank Sparse Collective Factorization model. We provided both batch and SGD based optimization algorithms. Experiments on five datasets showed effectiveness of proposed algorithms.

Acknowledgment

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 *Proceedings of the 19th ACM international conference on Information and knowledge management*, 2010.

[2] W. Pan, N. N. Liu, E. W. Xiang, and Q. Yang, "Transfer learning to predict missing ratings via heterogeneous user feedbacks," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, no. 3, 2011, p. 2318.

[3] S. Sahebi and P. Brusilovsky, "It takes two to tango: An exploration of domain pairs for cross-domain collaborative filtering," in *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 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 *IJCAI*, 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 *ICML*, 2015, pp. 1190–1198.

[6] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 4, no. 1, p. 1, 2010.

[7] Y. Zhang and J. Nie, "Probabilistic latent relational model for integrating heterogeneous information for recommendation," Technical report, School of Engineering, UCSC, Tech. Rep., 2010.

[8] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 650–658.

[9] W. Pan and Q. Yang, "Transfer learning in heterogeneous collaborative filtering domains," *Artificial intelligence*, vol. 197, pp. 39–55, 2013.

[10] B. Li, Q. Yang, and X. Xue, "Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction." in *IJCAI*, vol. 9, 2009, pp. 2052–2057.

[11] ——, "Transfer learning for collaborative filtering via a rating-matrix generative model," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 617–624.

[12] S. Gao, H. Luo, D. Chen, S. Li, P. Gallinari, and J. Guo, "Cross-domain recommendation via cluster-level latent factor model," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2013, pp. 161–176.

[13] O. Moreno, B. Shapira, L. Rokach, and G. Shani, "Talmud: transfer learning for multiple domains," in *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012, pp. 425–434.

[14] P. Cremonesi and M. Quadrana, "Cross-domain recommendations without overlapping data: myth or reality?" in *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 2014, pp. 297–300.

[15] C.-H. Lee, Y.-H. Kim, and P.-K. Rhee, "Web personalization expert with combining collaborative filtering and association rule mining technique," *Expert Systems with Applications*, vol. 21, no. 3, pp. 131–137, 2001.

[16] S. Wang, J. Tang, Y. Wang, and H. Liu, "Exploring implicit hierarchical structures for recommender systems," in *Proceedings of 24th International Joint Conference on Artificial Intelligence*, 2015.

[17] K. Lu, G. Zhang, R. Li, S. Zhang, and B. Wang, "Exploiting and exploring hierarchical structure in music recommendation," in *Information Retrieval Technology*. Springer, 2012, pp. 211–225.

[18] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. Schuller, "A deep semi-nmf model for learning hidden representations," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1692–1700.

[19] A. Mnih and R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in neural information processing systems*, 2007, pp. 1257–1264.

[20] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 426–434.

[21] S. Jiang, Z. Ding, and Y. Fu, "Deep low-rank sparse collective factorization for cross-domain recommendation," in *Proceedings of the 2017 ACM on Multimedia Conference, MM 2017, Mountain View, CA, USA, October 23-27, 2017*, 2017, pp. 163–171. [Online]. Available: http://doi.acm.org/10.1145/3123266.3123361

[22] F. Abel, E. Herder, G.-J. Houben, N. Henze, and D. Krause, "Cross-system user modeling and personalization on the social web," *User Modeling and User-Adapted Interaction*, vol. 23, no. 2-3, pp. 169–209, 2013.

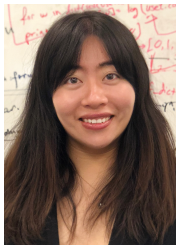
[23] B. Shapira, L. Rokach, and S. Freilikhman, "Facebook single and cross domain data for recommendation systems," *User Modeling and User-Adapted Interaction*, vol. 23, no. 2-3, 2013.

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

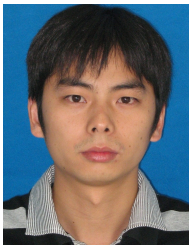
[25] P. Cremonesi, A. Tripodi, and R. Turrin, "Cross-domain recommender systems," in *11th International Conference on Data Mining Workshops*. IEEE, 2011, pp. 496–503.

[26] A. Tiroshi, S. Berkovsky, M. A. Kaafar, T. Chen, and T. Kuflik, "Cross social networks interests predictions based ongraph features," in *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 2013, pp. 319–322.

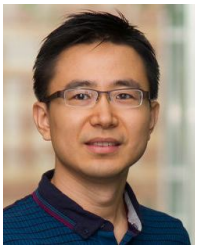
[27] M. Enrich, M. Braunhofer, and F. Ricci, "Cold-start management with cross-domain collaborative filtering and tags," in *E-Commerce and Web Technologies*, 2013.
[28] I. Fernández-Tobías and I. Cantador, "Exploiting social tags in matrix factorization models for cross-domain collaborative filtering," in *Proceedings of the 1st Workshop on New Trends in Content-based Recommender Systems, Foster City, California, USA*, 2014, pp. 34–41.
[29] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and C. Zhu, "Personalized recommendation via cross-domain triadic factorization," in *Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2013, pp. 595–606.
[30] W. Pan, E. W. Xiang, N. N. Liu, and Q. Yang, "Transfer learning in collaborative filtering for sparsity reduction." in *AAAI*, vol. 10, 2010, pp. 230–235.
[31] Y. Zhang, B. Cao, and D.-Y. Yeung, "Multi-domain collaborative filtering," *arXiv preprint arXiv:1203.3535*, 2012.
[32] B. Cao, N. N. Liu, and Q. Yang, "Transfer learning for collective link prediction in multiple heterogenous domains," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 159–166.
[33] Z. Ding, M. Shao, and Y. Fu, "Deep robust encoder through locality preserving low-rank dictionary," in *European Conference on Computer Vision*. Springer, 2016, pp. 567–582.
[34] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 663–670.
[35] Z. Ding, M. Shao, and Y. Fu, "Latent low-rank transfer subspace learning for missing modality recognition," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014, pp. 1192–1198.
[36] Z. Ding and Y. Fu, "Low-rank common subspace for multi-view learning," in *IEEE International Conference on Data Mining*. IEEE, 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4792–4800.
[38] S. Jiang, Y. Wu, and Y. Fu, "Deep bi-directional cross-triplet embedding for cross-domain clothing retrieval," in *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016, pp. 52–56.
[39] Y. Wu, J. Li, Y. Kong, and Y. Fu, "Deep convolutional neural network with independent softmax for large scale face recognition," in *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016, pp. 1063–1067.
[40] S. Jiang, M. Shao, C. Jia, and Y. Fu, "Consensus style centralizing auto-encoder for weak style classification." in *AAAI*, 2016, pp. 1223–1229.
[41] M. Maleszka, B. Mianowska, and N. T. Nguyen, "A method for collaborative recommendation using knowledge integration tools and hierarchical structure of user profiles," *Knowledge-Based Systems*, vol. 47, pp. 1–13, 2013.
[42] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.
[43] J. Yang, W. Yin, Y. Zhang, and Y. Wang, "A fast algorithm for edge-preserving variational multichannel image restoration," *SIAM Journal on Imaging Sciences*, vol. 2, no. 2, pp. 569–592, 2009.
[44] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 69–77.
[45] J. Wang, S. C. Hoi, P. Zhao, and Z.-Y. Liu, "Online multi-task collaborative filtering for on-the-fly recommender systems," in *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 2013, pp. 237–244.
[46] W. Pan and Z. Ming, "Interaction-rich transfer learning for collaborative filtering with heterogeneous user feedback," *IEEE Intelligent Systems*, vol. 29, no. 6, pp. 48–54, 2014.
[47] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender system-a case study," DTIC Document, Tech. Rep., 2000.
[48] R. Keshavan, A. Montanari, and S. Oh, "Matrix completion from noisy entries," in *Advances in Neural Information Processing Systems*, 2009, pp. 952–960.



Shuhui Jiang received the B.S. and M.S. degrees in Xi'an Jiaotong University, Xi'an, China, in 2007 and 2011, respectively. She is now pursuing her PHD degree in School of Electrical and Computer Engineering, Northeastern University (Boston, USA). She was the recipient of the Dean's Fellowship of Northeastern University from 2014. She is interested in machine learning, multimedia and computer vision. She has served as the reviewers for IEEE journals: IEEE Transactions on Neural Networks and Learning Systems etc. She was a research intern with Adobe research lab, San Jose, US, in summer 2016.



Zhengming Ding (S'14-M'18) received the B.Eng. degree in information security and the M.Eng. degree in computer software and theory from University of Electronic Science and Technology of China (UESTC), China, in 2010 and 2013, respectively. He received the Ph.D. degree from the Department of Electrical and Computer Engineering, Northeastern University, USA 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 machine learning and computer vision. Specifically, he devotes himself to develop scalable algorithms for challenging problems in transfer learning and deep learning scenario. He received the National Institute of Justice Fellowship during 2016-2018. He was the recipients 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 IEEE.



Yun Fu (S'07-M'08-SM'11) received the B.Eng. degree in information engineering and the M.Eng. degree in pattern recognition and intelligence systems from Xi'an Jiaotong University, China, respectively, and the M.S. degree in statistics and the Ph.D. 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 College of Computer and Information Science 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 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.