Can Movies and Books Collaborate?
Cross-Domain Collaborative Filtering for Sparsity Reduction

Bin Li[1], Qiang Yang[2] and Xiangyang Xue[1]
[1]School of Computer Science, Fudan University, Shanghai, China
[2]Department of Computer Science & Engineering,
Hong Kong University of Science & Technology, Hong Kong, China
libin@fudan.edu.cn, qyang@cse.ust.hk, xyxue@fudan.edu.cn

Abstract

The sparsity problem in collaborative filtering (CF) is a major bottleneck for most CF methods. In this paper, we consider a novel approach for alleviating the sparsity problem in CF by transferring user-item rating patterns from a dense auxiliary rating matrix in other domains (e.g., a popular movie rating website) to a sparse rating matrix in a target domain (e.g., a new book rating website). We do not require that the users and items in the two domains be identical or even overlap. Based on the limited ratings in the target matrix, we establish a bridge between the two rating matrices at a cluster-level of user-item rating patterns in order to transfer more useful knowledge from the auxiliary task domain. We first compress the ratings in the auxiliary rating matrix into an informative and yet compact cluster-level rating pattern representation referred to as a codebook. Then, we propose an efficient algorithm for reconstructing the target rating matrix by expanding the codebook. We perform extensive empirical tests to show that our method is effective in addressing the data sparsity problem by transferring the useful knowledge from the auxiliary tasks, as compared to many state-of-the-art CF methods.

1 Introduction

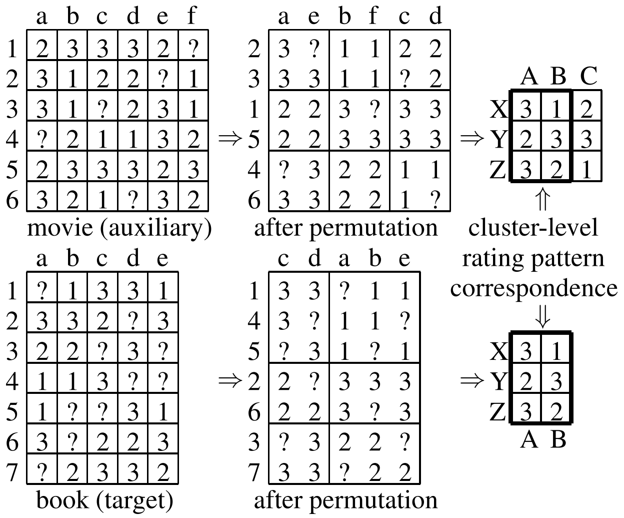
Collaborative filtering (CF) in recommender systems boils down to analyzing the tabular data, i.e., the user-item rating matrix. Memory-based methods [Resnick *et al.*, 1994; Sarwar *et al.*, 2005] aim at finding like-minded users in the rating matrix to predict the active user's ratings. Model-based methods [Hofmann and Puzicha, 1999; Si and Jin, 2003] model the user-item rating patterns based on the observed ratings in the rating matrix. Recently, the matrix factorization approach [Srebro and Jaakkola, 2003] has also been introduced. Most of these methods are based on the observed ratings in a rating matrix.

However, in real-world recommender systems, users can rate a very limited number of items, so the rating matrix is always extremely sparse. The available rating data that can be used for k-NN search, probabilistic modeling, or matrix factorization are radically insufficient. The sparsity problem has become a major bottleneck for most CF methods.

Although we can not fabricate more observed ratings in the considered rating matrix, we may borrow useful knowledge from another rating matrix in a different domain. Consider the following case: A new book rating website has opened. Due to a lack of visiting in the beginning, very few ratings can be used for collaborative filtering. Now suppose that we already have a dense movie rating matrix available on a popular movie rating website. Then, we ask, can we establish a bridge between the two rating matrices and transfer useful rating patterns from the movie rating matrix to the book rating matrix? Since movies and books are somewhat related (they have some correspondence in genre and the users of the both rating websites may reflect similar social aspects [Coyle and Smyth, 2008]), we believe that the rating matrices in different domains can share similar user-item rating patterns. Thus, transfer of knowledge can be beneficial. Furthermore, this knowledge-transfer idea can be generalized to any related real-world domains. For example, movies, books, and music are related in entertainment; mobile phones, notebook PCs, and digital cameras are related in electronic products.

Fig. 1 illustrates the underlying correspondence of the user-item rating patterns between two toy rating matrices, where we take movie and book domains for example. By permutating the rows (users) and columns (items) in the two matrices, the underlying correspondence can be discovered: User groups $\{2,3\}, \{1,5\}, \{4,6\}$ and item groups $\{a,e\}, \{b,f\}$ in the movie rating matrix correspond to user groups $\{1,4,5\}, \{2,6\}, \{3,7\}$ and item groups $\{c,d\}, \{a,b,e\}$ in the book rating matrix, respectively. In the real world, the user group $\{X,Y,Z\}$ may be students, professors, and engineers, while the item group $\{A,B\}$ may be comedies and dramas. Based on such a correspondence, the user-item rating patterns can be transferred from one rating matrix to another.

In this paper, we consider how to alleviate the sparsity problem in collaborative filtering by transferring user-item rating knowledge from one task to other related tasks. We call the task of interest the "target task" and the related task the "auxiliary task". The target task is represented as a sparse $p \times q$ rating matrix, $\mathbf{X}_{tgt}$, containing few observed ratings and would only result in poor prediction results. Meanwhile, we also get an auxiliary task from another domain, which is related to the target one (the aforementioned correspondence



	a	b	c	d	e	f
1	2	3	3	3	2	?
2	3	1	2	2	?	1
3	3	1	?	2	3	1
4	?	2	1	1	3	2
5	2	3	3	3	2	3
6	3	2	1	?	3	2

movie (auxiliary)

⇒

	a	e	b	f	c	d
2	3	?	1	1	2	2
3	3	3	1	1	?	2
1	2	2	3	?	3	3
5	2	2	3	3	3	3
4	?	3	2	2	1	1
6	3	3	2	2	1	?

after permutation

⇒

	A	B	C
X	3	1	2
Y	2	3	3
Z	3	2	1

⇑

cluster-level rating pattern correspondence

⇓

	a	b	c	d	e
1	?	1	3	3	1
2	3	3	2	?	3
3	2	2	?	3	?
4	1	1	3	?	?
5	1	?	?	3	1
6	3	?	2	2	3
7	?	2	3	3	2

book (target)

⇒

	c	d	a	b	e
1	3	3	?	1	1
4	3	?	1	1	?
5	?	3	1	?	1
2	2	?	3	3	3
6	2	2	3	?	3
3	?	3	2	2	?
7	3	3	?	2	2

after permutation

⇒

	A	B
X	3	1
Y	2	3
Z	3	2

Figure 1: The correspondence of the user-item rating patterns between two toy rating matrices. The missing values are denoted by '?'. The book rating matrix has a subset cluster-level rating patterns of the movie rating matrix.

underlies the relatedness and the rating scales in both rating matrices should be consistent, e.g., $1 - 5$) and has a dense $n \times m$ rating matrix, $\mathbf{X}_{aux}$. In this paper, we show how to learn informative and yet compact cluster-level user-item rating patterns from the auxiliary rating matrix and transfer them to the target rating matrix, to enable better prediction results in the target task. We refer to this collection of patterns to be transferred as a "codebook", which is a $k \times l$ ($k < n, k < p, l < m, l < q$) cluster-level user-item rating matrix that encodes the user-item clusters in $\mathbf{X}_{aux}$. By assuming the user-item rating patterns in $\mathbf{X}_{tgt}$ is similar to $\mathbf{X}_{aux}$, we can reconstruct the target rating matrix by expanding the codebook. If the codebook is expressive enough to describe various user-item rating patterns, the reconstructed rating matrix can fit the true target rating matrix well.

2 Codebook Construction

In collaborative filtering, users with similar tastes or items with similar attributes usually behave very similarly. For example, in Fig. 1, users in X prefer items in A to B, and items in B are liked by users in Y but disliked by users in X. This observation suggests that users and items can be clustered into groups. If users and items can be well clustered, a much more compact user-item rating matrix, which only comprises the representatives of all the user/item clusters, is able to summarize and compress the original redundant rating matrix. As a result, we only need the cluster indicators for the users and items to recover the original rating matrix. This is why the transferred knowledge is called a "codebook".

Definition 2.1 (Codebook). *Codebook is a $k \times l$ matrix which compresses the cluster-level user-item rating patterns of k user clusters and l item clusters in the original rating matrix.*

Ideally, if the users/items in the same clusters are identical, we only need to select one user/item pair from each user/item cluster to form the codebook. However, in real-world cases, the users/items in the same cluster cannot be identical. A natural choice is to use the cluster centroid as the prototype for each cluster. To this end, we need to simultaneously cluster the rows (users) and columns (items) of the rating matrix. For codebook construction, what we need are only the user and item cluster indicators, and for this we can choose any co-clustering algorithms. In this paper, we adopt the orthogonal nonnegative matrix tri-factorization (ONMTF) algorithm [Ding *et al.*, 2006], since it is proved to be equivalent to a two-way K-means clustering algorithm and easy to implement. The auxiliary rating matrix $\mathbf{X}_{aux}$ can then be tri-factorized as follows

$$\min_{\mathbf{U}\geq 0, \mathbf{V}\geq 0, \mathbf{S}\geq 0} \left\|\mathbf{X}_{aux} - \mathbf{U}\mathbf{S}\mathbf{V}^\top\right\|_F^2 \quad (1)$$
$$\text{s.t.} \quad \mathbf{U}^\top\mathbf{U} = \mathbf{I}, \mathbf{V}^\top\mathbf{V} = \mathbf{I},$$

where $\mathbf{U} \in \mathbb{R}_+^{n\times k}$, $\mathbf{V} \in \mathbb{R}_+^{m\times l}$, $\mathbf{S} \in \mathbb{R}_+^{k\times l}$ ($\mathbb{R}_+$ means nonnegative real number), and $\|\cdot\|_F$ denotes the Frobenius norm. Due to the nonnegativity and orthogonality constraints for $\mathbf{U}$ and $\mathbf{V}$, each row of $\mathbf{U}$ and $\mathbf{V}$ can have only one nonnegative entry, which is the cluster indicator for the user/item in that row. The constrained optimization problem (1) can be solved efficiently by using equations (28–30) in [Ding *et al.*, 2006]. Due to space limitations, we skip them here.

The current form of the user/item cluster indicators, $\mathbf{U}$ and $\mathbf{V}$, may not have interpretable meanings for the user/item memberships. For simplicity, in this paper we use binary data to represent $\mathbf{U}$ and $\mathbf{V}$ by setting the nonnegative entry in each row to be 1 and the others to be 0. Then, these binary user/item cluster indicators form a (hard membership) matrix, denoted by $\mathbf{U}_{aux}$ and $\mathbf{V}_{aux}$, for the auxiliary rating matrix.

We can construct the codebook $\mathbf{B}$ as follows

$$\mathbf{B} = [\mathbf{U}_{aux}^\top\mathbf{X}_{aux}\mathbf{V}_{aux}] \oslash [\mathbf{U}_{aux}^\top\mathbf{1}\mathbf{1}^\top\mathbf{V}_{aux}], \quad (2)$$

where $\oslash$ means entry-wise division. Eq. (2) means averaging all the ratings in each user-item co-cluster as an entry in the codebook, i.e., the cluster-level rating pattern. The details for codebook construction are summarized in Algorithm 1.

From Eq. (2), the codebook $\mathbf{B}$ is either the user basis of the row space of $\mathbf{X}_{aux}\mathbf{V}_{aux}$ or the item basis of the column space of $\mathbf{U}_{aux}^\top\mathbf{X}_{aux}$. Therefore, the codebook can be viewed as the "two-sided basis" for $\mathbf{X}_{aux}$ such that $\mathbf{X}_{aux}$ can be recovered by duplicating the rows and columns of the codebook.

A remaining task is to select the user/item cluster numbers, k and l. Too many clusters may comprise redundant information such that more computational cost is incurred during the codebook construction and transfer. Having too few clusters may make it insufficient to encode the user/item data and may cause the algorithm to miss much useful information. Therefore, a suitable size for the codebook should be not only expressive enough to compress most information in the data but also compact enough for computational efficiency.

3 Codebook Transfer

After obtaining the codebook, the user-item rating patterns can be transferred from $\mathbf{X}_{aux}$ to $\mathbf{X}_{tgt}$. By assuming that there

Algorithm 1 Codebook Construction

```
Input: An $n \times m$ auxiliary rating matrix $\mathbf{X}_{aux}$; the numbers of user and item clusters $k$ and $l$.
Output: A $k \times l$ codebook $\mathbf{B}$ learned from $\mathbf{X}_{aux}$.
Randomly initialize $\mathbf{U}^{(0)}$, $\mathbf{V}^{(0)}$, and $\mathbf{S}^{(0)}$ in Eq. (1).
for $t \leftarrow 1, \ldots, T$ do
    Update $\mathbf{U}^{(t-1)}$, $\mathbf{V}^{(t-1)}$, $\mathbf{S}^{(t-1)}$ using Eqs. (28–30) in [Ding et al., 2006], and obtain $\mathbf{U}^{(t)}$, $\mathbf{V}^{(t)}$, $\mathbf{S}^{(t)}$.
end for
Allocate spaces for $\mathbf{U}_{aux}$ and $\mathbf{V}_{aux}$.
for $i \leftarrow 1, \ldots, n$ do
    $\hat{j} = \arg\max_{j \in \{1,\ldots,k\}} \{\mathbf{U}_{ij}\}$.
    $[\mathbf{U}_{aux}]_{i\hat{j}} \leftarrow 1$; $[\mathbf{U}_{aux}]_{ij} \leftarrow 0$ for $j \in \{1, \ldots, k\}/\hat{j}$.
end for
for $i \leftarrow 1, \ldots, m$ do
    $\hat{j} = \arg\max_{j \in \{1,\ldots,l\}} \{\mathbf{V}_{ij}\}$.
    $[\mathbf{V}_{aux}]_{i\hat{j}} \leftarrow 1$; $[\mathbf{V}_{aux}]_{ij} \leftarrow 0$ for $j \in \{1, \ldots, l\}/\hat{j}$.
end for
Calculate the codebook $\mathbf{B}$ using Eq. (2).
```

exists implicit correspondence between the user/item clusters of the auxiliary task and those of the target task (like the illustration in Fig. 1), $\mathbf{X}_{tgt}$ can then be reconstructed by expanding the codebook; i.e., this can be done by duplicating certain rows and columns in the codebook. The duplication of the i-th row/column in the codebook means that there are a set of users/items in $\mathbf{X}_{tgt}$ that behaves like the i-th user/item cluster prototype in $\mathbf{X}_{aux}$.

The reconstruction process of $\mathbf{X}_{tgt}$ expands the codebook as it reduces the differences based on a certain loss function (in this paper, we adopt the quadratic loss function) between the observed ratings in $\mathbf{X}_{tgt}$ and the corresponding entries in the reconstructed rating matrix. Here we employ a binary weighting matrix $\mathbf{W}$ of the same size as $\mathbf{X}_{tgt}$ to mask the unobserved entries, where $\mathbf{W}_{ij} = 1$ if $[\mathbf{X}_{tgt}]_{ij}$ is rated and $\mathbf{W}_{ij} = 0$ otherwise. Consequently, only the quadratic losses at the observed entries in $\mathbf{X}_{tgt}$ are taken into account in the objective function

$$\min_{\substack{\mathbf{U}_{tgt} \in \{0,1\}^{p \times k} \\ \mathbf{V}_{tgt} \in \{0,1\}^{q \times l}}} \left\| [\mathbf{X}_{tgt} - \mathbf{U}_{tgt}\mathbf{B}\mathbf{V}_{tgt}^\top] \circ \mathbf{W} \right\|_F^2 \qquad (3)$$
$$\text{s.t.} \quad \mathbf{U}_{tgt}\mathbf{1} = \mathbf{1}, \mathbf{V}_{tgt}\mathbf{1} = \mathbf{1},$$

where $\circ$ denotes the entry-wise product. The constraints indicate that each row in $\mathbf{U}_{tgt}$ or $\mathbf{V}_{tgt}$ is the cluster membership of the user or item. In this work, we only consider the case that each user/item can only belong to one user/item cluster. Thus, $\mathbf{U}_{tgt}$ and $\mathbf{V}_{tgt}$ can only take binary values $\{0, 1\}$, and only one '1' can be in each row of $\mathbf{U}_{tgt}$ and $\mathbf{V}_{tgt}$. The optimization problem (3) becomes a binary integer programming problem, which is known to be NP-hard. In the sequel, we propose an efficient algorithm to search for the local minimum solution (Algorithm 2).

Proposition 3.1. *Algorithm 2 monotonically decreases the objective function (3).*

Proof. Algorithm 2 alternatively updating $\mathbf{U}_{tgt}$ and $\mathbf{V}_{tgt}$, so we only need to prove that, at the t-th ($t > 0$) iterative round, the following two inequalities always hold

Algorithm 2 Codebook Transfer

```
Input: The $p \times q$ target rating matrix $\mathbf{X}_{tgt}$; the $p \times q$ weighting matrix $\mathbf{W}$; the $k \times l$ codebook $\mathbf{B}$.
Output: The filled-in $p \times q$ target rating matrix $\tilde{\mathbf{X}}_{tgt}$.
Allocate spaces for $\mathbf{U}_{tgt}$ and $\mathbf{V}_{tgt}$.
for $i \leftarrow 1, \ldots, m$ do
    Randomly select $\hat{j}$ from $\{1, \ldots, l\}$.
    $[\mathbf{V}_{tgt}^{(0)}]_{i\hat{j}} \leftarrow 1$; $[\mathbf{V}_{tgt}^{(0)}]_{ij} \leftarrow 0$ for $j \in \{1, \ldots, l\}/\hat{j}$.
end for
for $t \leftarrow 1, \ldots, T$ do
    for $i \leftarrow 1, \ldots, p$ do
        $\hat{j} = \arg\min_j \left\|[\mathbf{X}_{tgt}]_{i*} - [\mathbf{B}[\mathbf{V}_{tgt}^{(t-1)}]^\top]_{j*}\right\|^2_{\mathbf{W}_{i*}}$.
        $[\mathbf{U}_{tgt}^{(t)}]_{i\hat{j}} \leftarrow 1$; $[\mathbf{U}_{tgt}^{(t)}]_{ij} \leftarrow 0$ for $j \in \{1, \ldots, k\}/\hat{j}$.
    end for
    for $i \leftarrow 1, \ldots, q$ do
        $\hat{j} = \arg\min_j \left\|[\mathbf{X}_{tgt}]_{*i} - [\mathbf{U}_{tgt}^{(t)}\mathbf{B}]_{*j}\right\|^2_{\mathbf{W}_{*i}}$.
        $[\mathbf{V}_{tgt}^{(t)}]_{i\hat{j}} \leftarrow 1$; $[\mathbf{V}_{tgt}^{(t)}]_{ij} \leftarrow 0$ for $j \in \{1, \ldots, l\}/\hat{j}$.
    end for
end for
Calculate the filled-in rating matrix $\tilde{\mathbf{X}}_{tgt}$ using Eq. (8).
```

$$\left\|[\mathbf{X}_{tgt} - \mathbf{U}_{tgt}^{(t-1)}\mathbf{B}[\mathbf{V}_{tgt}^{(t-1)}]^\top] \circ \mathbf{W}\right\|_F^2 \geq$$
$$\left\|[\mathbf{X}_{tgt} - \mathbf{U}_{tgt}^{(t)}\mathbf{B}[\mathbf{V}_{tgt}^{(t-1)}]^\top] \circ \mathbf{W}\right\|_F^2 \qquad (4)$$

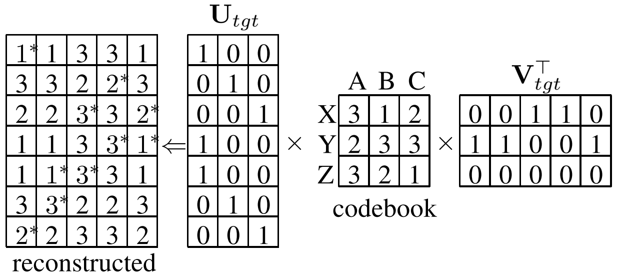
and

$$\left\|[\mathbf{X}_{tgt} - \mathbf{U}_{tgt}^{(t)}\mathbf{B}[\mathbf{V}_{tgt}^{(t-1)}]^\top] \circ \mathbf{W}\right\|_F^2 \geq$$
$$\left\|[\mathbf{X}_{tgt} - \mathbf{U}_{tgt}^{(t)}\mathbf{B}[\mathbf{V}_{tgt}^{(t)}]^\top] \circ \mathbf{W}\right\|_F^2. \qquad (5)$$

We first prove Eq. (4). When $\mathbf{U}_{tgt}$ is updated, $\mathbf{V}_{tgt}$ is fixed.

$$\begin{aligned}
& \left\|[\mathbf{X}_{tgt} - \mathbf{U}_{tgt}^{(t-1)}\mathbf{B}[\mathbf{V}_{tgt}^{(t-1)}]^\top] \circ \mathbf{W}\right\|_F^2 \\
= \quad & \sum_{i=1}^{n}\sum_{j=1}^{k}[\mathbf{U}_{tgt}^{(t-1)}]_{ij}\left\|[\mathbf{X}_{tgt}]_{i*} - [\mathbf{B}[\mathbf{V}_{tgt}^{(t-1)}]^\top]_{j*}\right\|^2_{\mathbf{W}_{i*}} \\
= \quad & \sum_{i=1}^{n}\left\|[\mathbf{X}_{tgt}]_{i*} - [\mathbf{B}[\mathbf{V}_{tgt}^{(t-1)}]^\top]_{\delta([\mathbf{U}_{tgt}^{(t-1)}]_{i*})*}\right\|^2_{\mathbf{W}_{i*}} \qquad (6) \\
\geq \quad & \sum_{i=1}^{n}\left\|[\mathbf{X}_{tgt}]_{i*} - [\mathbf{B}[\mathbf{V}_{tgt}^{(t-1)}]^\top]_{\delta([\mathbf{U}_{tgt}^{(t)}]_{i*})*}\right\|^2_{\mathbf{W}_{i*}} \qquad (7) \\
= \quad & \left\|[\mathbf{X}_{tgt} - \mathbf{U}_{tgt}^{(t)}\mathbf{B}[\mathbf{V}_{tgt}^{(t-1)}]^\top] \circ \mathbf{W}\right\|_F^2
\end{aligned}$$

where $[\cdot]_{i*}$ (or $[\cdot]_{*i}$) denotes the i-th row (or column) in a matrix, and $\|\mathbf{x}\|^2_{\mathbf{W}_{i*}} = \mathbf{x}^\top \text{diag}(\mathbf{W}_{i*})\mathbf{x}$ is the weighted l_2-norm. The indicator function $\delta(\cdot)$ returns the index of the largest component in a vector. Eq. (6) is obtained based on the fact that each user only belongs to one cluster (i.e., only one '1' in $[\mathbf{U}_{tgt}^{(t-1)}]_{i*}$). Thus, we can replace the sum of $[\mathbf{U}_{tgt}^{(t-1)}]_{i*}$-weighted quadratic losses by only one quadratic loss indexed by the indicator function $\delta([\mathbf{U}_{tgt}^{(t-1)}]_{i*})$, which tells that the



$$\underset{\text{reconstructed}}{\begin{array}{|c|c|c|c|c|}\hline 1^* & 1 & 3 & 3 & 1 \\ \hline 3 & 3 & 2 & 2^* & 3 \\ \hline 2 & 2 & 3^* & 3 & 2^* \\ \hline 1 & 1 & 3 & 3^* & 1^* \\ \hline 1 & 1^* & 3^* & 3 & 1 \\ \hline 3 & 3^* & 2 & 2 & 3 \\ \hline 2^* & 2 & 3 & 3 & 2 \\ \hline \end{array}} \Leftarrow \overset{\mathbf{U}_{tgt}}{\begin{array}{|c|c|c|}\hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline 1 & 0 & 0 \\ \hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline \end{array}} \times \underset{\text{codebook}}{\begin{array}{c|c|c|c|} & \text{A} & \text{B} & \text{C} \\ \hline \text{X} & 3 & 1 & 2 \\ \hline \text{Y} & 2 & 3 & 3 \\ \hline \text{Z} & 3 & 2 & 1 \\ \hline \end{array}} \times \overset{\mathbf{V}_{tgt}^{\top}}{\begin{array}{|c|c|c|c|c|}\hline 0 & 0 & 1 & 1 & 0 \\ \hline 1 & 1 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}}$$

Figure 2: Target rating matrix reconstruction. The entries with '*' are the filled-in missing values; the ones without '*' are identical to those in the original rating matrix (loss is 0).

cluster to which the i-th user (i.e., $[\mathbf{X}_{tgt}]_{i*}$) belongs at the $(t-1)$-th iterative round. The inequality between Eq. (6) and Eq. (7) can be obtained, at the t-th iterative round, by finding the nearest cluster centroid in $[\mathbf{B}[\mathbf{V}_{tgt}^{(t-1)}]^{\top}]_{j*}, j = 1, \ldots, k$, for the i-th user based on the distance metric $\| \cdot \|^2_{\mathbf{W}_{i*}}$.

Eq. (5) can also be proved in the same way as Eq. (4). Alternatively updating $\mathbf{U}_{tgt}$ and $\mathbf{V}_{tgt}$ can monotonically reduce the value of the objective function (3) and make it converge to a local minimum. □

Algorithm 2 is very efficient. It takes a computational complexity of $\mathcal{O}(T(pk + ql))$ and can always converge to a local minimum in few rounds; in our empirical tests in Section 5, it usually takes less than ten rounds ($T < 10$). Thus, one can run the algorithm multiple times with different random initializations for $\mathbf{V}_{tgt}$ to search a better local minimum.

It is worth noting that the value of the objective function (3) is also an index of the relatedness between the target and auxiliary tasks. The smaller the value is, the better the rating patterns are corresponded to each other. In contrast, a large value implies a weak correspondence which may lead to potential negative transfer [Rosenstein *et al.*, 2005].

Once the user and item membership (cluster indicator) matrices $\mathbf{U}_{tgt}$ and $\mathbf{V}_{tgt}$ are obtained, we can reconstruct the target rating matrix $\mathbf{X}_{tgt}$ by duplicating the rows and columns of the codebook using $\mathbf{U}_{tgt}\mathbf{B}\mathbf{V}_{tgt}^{\top}$. Fig. 2 illustrates the target rating matrix reconstruction for the book rating matrix (viewed as the target task) in Fig. 1 by expanding the codebook learned from the movie rating matrix (viewed as the auxiliary task) in Fig. 1. Then the filled-in target rating matrix, $\tilde{\mathbf{X}}_{tgt}$, is defined as

$$\tilde{\mathbf{X}}_{tgt} = \mathbf{W} \circ \mathbf{X}_{tgt} + [\mathbf{1} - \mathbf{W}] \circ [\mathbf{U}_{tgt}\mathbf{B}\mathbf{V}_{tgt}^{\top}], \qquad (8)$$

which can be used as the training data set for memory-based collaborative filtering.

We summarize the proposed cross-domain collaborative filtering method via codebook transfer (CBT) as the following three steps: 1) Construct the codebook $\mathbf{B}$ from $\mathbf{X}_{aux}$ using Algorithm 1; 2) based on $\mathbf{B}$, fill in the missing values in $\mathbf{X}_{tgt}$ using Algorithm 2 and get $\tilde{\mathbf{X}}_{tgt}$; 3) take $\tilde{\mathbf{X}}_{tgt}$ as the training data set for memory-based collaborative filtering.

4 Related Work

In the proposed method, user-item rating patterns are learned from a dense auxiliary rating matrix and transferred to a sparse target rating matrix in the form of a codebook, which are two-sided prototypes (basis) of both users and items. This knowledge transfer fashion is similar to the self-taught learning (STL) [Raina *et al.*, 2007] in transfer learning area. STL first learns an abundant data representation (basis) from a large collection of auxiliary data. It then represents the test data as the linear combination of a set of basis. A major difference from our work is that STL transfers one-sided data representation (in row spaces) while our method transfers two-sided data representation (in both row and column space). Furthermore, the application domain of STL is image analysis, whereas we focus on collaborative filtering.

Codebook modeling is also related to model-based collaborative filtering methods, such as two-sided clustering (TSC) model [Hofmann and Puzicha, 1999] and flexible mixture model (FMM) [Si and Jin, 2003]. TSC is defined as

$$P(r, i, j) = P(i)P(j)P(r|u_i, v_j)\phi(u_i, v_j),$$

where u_i denotes the user cluster to which the i-th user belongs, v_j the item cluster to which the j-th item belongs, and $\phi(\cdot, \cdot)$ the cluster association parameter. The entry $\mathbf{B}_{uv}$ in the codebook perform the same role as $P(r|u, v)\phi(u, v)$ in TSC. FMM is also a probabilistic model defined as

$$P(r, i, j) = \sum_{u,v} P(i|u)P(j|v)P(r|u, v)P(u)P(v),$$

where u denotes user cluster, v item cluster, and r rating. The entry $\mathbf{B}_{uv}$ in the codebook can explicitly correspond to the joint probability, $P(r|u, v)P(u)P(v)$, in FMM. In the above models, the user-item rating patterns are modeled in the probabilistic framework; whereas in this work, for the simplicity of transfer, user-item rating patterns are modeled using hard-membership clustering rather than probabilistic models.

In collaborative filtering, several research works have been done on sparsity reduction and similarity enhancement. [Wilson *et al.*, 2003] exploits the implicit similarity knowledge via item-item association rule mining and case-based reasoning. [Xue *et al.*, 2005] smooths the rating matrix by the average rating values within user clusters. [George and Merugu, 2005] models rating patterns by simultaneously clustering users and items. [Chen *et al.*, 2007] also simultaneously clusters users and items and fuses the user-based and item-based search results. However, these methods restrict themselves to learning in the target task domain only. In contrast, we aim at establishing the correspondence between a target task and an auxiliary task, in order to seek more useful knowledge from the auxiliary task.

5 Experiments

In this section, we focus on investigating whether additional knowledge can be really gained from the auxiliary rating matrix in other domains to remedy the sparse rating matrices in the target domain. We aim to validate that transferring useful information from a dense auxiliary rating matrix is more effective than mining the limited knowledge in the sparse target rating matrix itself.

5.1 Data sets

We use the following data sets in our empirical tests.

- EachMovie[1] (Auxiliary task): A movie rating data set comprising 2.8 million ratings (scales 1–6) by 72,916 users on 1628 movies. Since $\mathbf{X}_{aux}$ is assumed to be denser than $\mathbf{X}_{tgt}$, we extract a sub-matrix to simulate the auxiliary task by selecting 500 users and 500 movies with most ratings (rating ratio 54.6%), and the missing values for each user is simply filled in with her average rating. For a rating-scale consistency with the target tasks, we replace 6 with 5 in the rating matrix to make the rating scales from 1 to 5.
- MovieLens[2] (Target task): A movie rating data set comprising 100,000 ratings (scales 1–5) by 943 users on 1682 movies. We randomly select 500 users with more than 40 ratings and 1000 movies (rating ratio 11.6%).
- Book-Crossing[3] (Target task): A book rating data set comprising more than 1.1 million ratings (scales 1–10) by 278,858 users on 271,379 books. We randomly select 500 users and 1000 books with most ratings (rating ratio 3.02%). We also normalize the rating scales from 1 to 5.

5.2 Compared Methods

Our proposed method aims at filling in the missing values in the rating matrix to alleviate the sparsity problem in the similarity computation between training and test users. Thus, our method is compared with the state-of-the-art methods which also focus on missing-value problems:

- PCC (baseline): Do not handle the missing values.
- CBS (Scalable cluster-based smoothing [Xue *et al.*, 2005]): Take the average of the observed ratings in the same user cluster to fill in the missing values. The number of user clusters is set to 50.
- WLR (Weighted low-rank approximation [Srebro and Jaakkola, 2003]): Fill in the missing values with corresponding entries in the reconstructed low-rank matrix. The dimension of the latent space is set to 50.
- CBT (Codebook transfer, the proposed method): Fill in the missing values with corresponding entries in the reconstructed rating matrix by expanding the codebook. The numbers of both user and item clusters are set to 50.

After filling in the missing values in the two target rating matrices, we evaluate the knowledge gains by different methods in the same framework, i.e., to find the top K nearest neighbors ($K = 20$) based on the Pearson correlation coefficients (PCC) similarity to predict test user ratings.

5.3 Evaluation Protocol

Following [Xue *et al.*, 2005], we evaluate the algorithm performance under different configurations. The first 100, 200, and 300 users in the both target data sets are used for training, respectively, and the last 200 users for testing. For each test user, three different sizes of the observed ratings (Given5, Given10, Given15) are provided for computing PCC and the remaining ratings are used for evaluation. Note that in our experiments, the given observed rating indices are randomly selected 10 times, so that the reported results in Table 1 and 2 are the average results over ten splits.

Table 1: MAE on MovieLens (average over 10 splits)

Training Set	Method	Given5	Given10	Given15
ML100	PCC	0.930	0.883	0.873
	CBS	0.874	0.845	0.839
	WLR	0.915	0.875	0.890
	CBT	**0.840**	**0.802**	**0.786**
ML200	PCC	0.905	0.878	0.878
	CBS	0.871	0.833	0.828
	WLR	0.941	0.903	0.883
	CBT	**0.839**	**0.800**	**0.784**
ML300	PCC	0.897	0.882	0.885
	CBS	0.870	0.834	0.819
	WLR	1.018	0.962	0.938
	CBT	**0.840**	**0.801**	**0.785**

Table 2: MAE on Book-Crossing (average over 10 splits)

Training Set	Method	Given5	Given10	Given15
BX100	PCC	0.677	0.710	0.693
	CBS	0.664	0.655	0.641
	WLR	1.170	1.182	1.174
	CBT	**0.614**	**0.611**	**0.593**
BX200	PCC	0.687	0.719	0.695
	CBS	0.661	0.644	0.630
	WLR	0.965	1.024	0.991
	CBT	**0.614**	**0.600**	**0.581**
BX300	PCC	0.688	0.712	0.682
	CBS	0.659	0.655	0.633
	WLR	0.842	0.837	0.829
	CBT	**0.605**	**0.592**	**0.574**

The evaluation metric we adopt is Mean Absolute Error (MAE): $(\sum_{i \in T} |r_i - \tilde{r}_i|)/|T|$, where T denotes the set of test ratings, r_i is ground truth and $\tilde{r}_i$ is predicted rating. A smaller value of MAE means a better performance.

5.4 Results

The comparison results are reported in Table 1 and 2. As expected, our method clearly outperforms all the other compared methods under all the testing configurations on both target data sets. CBS performs slightly better than the baseline, which implies that the users in the same cluster do share common rating patterns. WLR performs even worse than the baseline because of the extreme sparsity of the two target rating matrices, which can lead to a poor low-rank approximation and introduce noise instead of knowledge. The experimental results show that our proposed method CBT is more effective in alleviating the sparsity problem by transferring useful knowledge from the dense auxiliary rating matrix, as

[1] http://www.cs.cmu.edu/~lebanon/IR-lab.htm

[2] http://www.grouplens.org/node/73

[3] http://www.informatik.uni-freiburg.de/~cziegler/BX/

compared to the methods which aim at mining the knowledge in the sparse target rating matrix from a single target domain.

It is also worth noting that, with the number of given observed ratings increasing (from 5 to 15), the performance improvements of our method are most notable among the compared methods. Given one more observed rating for a test user, the amount of improvement indicates how much novel useful knowledge that has been introduced by the corresponding column in the target rating matrix (e.g., given the second rating of the test user observable, the second column in the rating matrix then can be used for computing PCC for this test user). Since most of the ratings in the target rating matrix for computing PCC are filled-in values, we can conclude that our method can fill in the most informative values, owing to the use of knowledge transfer.

5.5 Discussion

Although our proposed CBT method can clearly outperform the other compared methods on the both target data sets, we can see that there still exists some room for further performance improvements. We consider that a crucial problem that holds back further improvements lies in the inherent problem of the data sets; i.e., the users and items in the rating matrices may not always be able to be grouped into high quality clusters. We observe that the average ratings of the three data sets are far larger than the medians (given the median being 3, the average ratings are 3.66, 3.58, and 4.21 for the three data sets, respectively). This may be caused by the fact that the items with the most ratings are usually the most popular ones. In other words, users are willing to rate their favorite items to recommend them to others, but have little interest to rate the items they dislike. Given that no clear user and item groups can be discovered for these cases, it is hard to learn a good cluster-level rating pattern representation.

6 Conclusion

In this paper, we presented a novel cross-domain collaborative filtering method via codebook-based knowledge transfer (CBT) for recommender systems, which can transfer useful knowledge from the auxiliary rating matrix in some other domains to remedy the sparsity of the rating matrix in a target domain. The knowledge is transferred in the form of a codebook, which is learned from an auxiliary rating matrix by compressing the cluster-level user-item rating patterns into an informative and yet compact representation. The sparse target rating matrix can thus be reconstructed by expanding the codebook. The experimental results show that codebook transfer can clearly outperform the many state-of-the-art methods. This can validate that, in collaborative filtering, additional useful information indeed can be gained from the auxiliary tasks in other domains to aid the target task.

Acknowledgments

Bin Li and Qiang Yang are supported by Hong Kong CERG Grant 621307; Bin Li and Xiangyang Xue are supported in part by Shanghai Leading Academic Discipline Project (No. B114) and NSF of China (No. 60873178).

References

[Chen *et al.*, 2007] Gang Chen, Fei Wang, and Changshui Zhang. Collaborative filtering using orthogonal nonnegative matrix tri-factorization. In *Proc. of the IEEE Int'l Conf. on Data Mining Workshops*, pages 303–308, 2007.

[Coyle and Smyth, 2008] Maurice Coyle and Barry Smyth. Web search shared: Social aspects of a collaborative, community-based search network. In *Proc. of the 15th Int'l Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 103–112, 2008.

[Ding *et al.*, 2006] Chris Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix tri-factorizations for clustering. In *Proc. of the 12th ACM SIGKDD Int'l Conf.*, pages 126–135, 2006.

[George and Merugu, 2005] Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *Proc. of the Fifth IEEE Int'l Conf. on Data Mining*, pages 625–628, 2005.

[Hofmann and Puzicha, 1999] Thomas Hofmann and Jan Puzicha. Latent class models for collaborative filtering. In *Proc. of the 16th Int'l Joint Conf. on Artificial Intelligence*, pages 688–693, 1999.

[Raina *et al.*, 2007] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proc. of the 24th Int'l Conf. on Machine Learning*, pages 759–766, 2007.

[Resnick *et al.*, 1994] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proc. of the ACM Conf. on Computer Supported Cooperative Work*, pages 175–186, 1994.

[Rosenstein *et al.*, 2005] Michael T. Rosenstein, Zvika Marx, and Leslie Pack Kaelbling. To transfer or not to transfer. In *Proc. of the NIPS Workshop on Inductive Transfer: 10 Years Later*, 2005.

[Sarwar *et al.*, 2005] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. of the 10th Int'l World Wide Web Conf.*, pages 285–295, 2005.

[Si and Jin, 2003] Luo Si and Rong Jin. Flexible mixture model for collaborative filtering. In *Proc. of the 20th Int'l Conf. on Machine Learning*, pages 704–711, 2003.

[Srebro and Jaakkola, 2003] Nathan Srebro and Tommi Jaakkola. Weighted low-rank approximations. In *Proc. of the 20th Int'l Conf. on Machine Learning*, pages 720–727, 2003.

[Wilson *et al.*, 2003] David C. Wilson, Barry Smyth, and Derry O'sullivan. Sparsity reduction in collaborative recommendation: A case-based approach. *Int'l Journal of Pattern Recognition and Artificial Intelligence*, 17(5):863–884, 2003.

[Xue *et al.*, 2005] Gui-Rong Xue, Chenxi Lin, Qiang Yang, Wensi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proc. of the 28th SIGIR Conf.*, pages 114–121, 2005.