Matching Users and Items Across Domains to Improve the Recommendation Quality

Chung-Yi Li
Department of Computer Science and Information Engineering,
National Taiwan University, Taiwan
r00922051@csie.ntu.edu.tw

Shou-De Lin
Department of Computer Science and Information Engineering,
National Taiwan University, Taiwan
sdlin@csie.ntu.edu.tw

ABSTRACT

Given two homogeneous rating matrices with some overlapped users/items whose mappings are unknown, this paper aims at answering two questions. First, can we identify the unknown mapping between the users and/or items? Second, can we further utilize the identified mappings to improve the quality of recommendation in either domain? Our solution integrates a latent space matching procedure and a refining process based on the optimization of prediction to identify the matching. Then, we further design a transfer-based method to improve the recommendation performance. Using both synthetic and real data, we have done extensive experiments given different real life scenarios to verify the effectiveness of our models. The code and other materials are available at http://www.csie.ntu.edu.tw/~r00922051/matching/

Categories and Subject Descriptors

I.2.6 [**ARTIFICIAL INTELLIGENCE**]: Learning—*Parameter learning*

Keywords

Collaborative Filtering; Transfer Learning; Matrix Factorization

1. INTRODUCTION

Nowadays, recommendation systems are widely deployed for not only e-business services but also brick and mortar stores. In general there are two possible strategies in designing a recommendation system: content-based [10] and collaborative filtering (CF) based [5] approaches. The former relies on the content and profile information of users and items to perform recommendation, while the latter relies mainly on the ratings of items given by users to identify similar users and items for recommendation. This paper mainly focuses on the analysis of CF-based models and assumes only rating data are available.

Collecting sufficient amount of rating data has been recognized as a critical factor in designing a successful CF-based recommendation system. As CF-based recommendation systems become increasingly popular, it is not hard to imagine more and more rating data would become available to models for different businesses. One might wonder what kind of useful knowledge one can extract from several different rating sets of completely independent services.

For example, a newly established online song requesting service, which possesses some internal ratings about its users' preferences to songs, can take advantage of the publicly available rating data from other similar services (e.g. Yahoo! music service). The reason is that we know there exists some overlapping between users and/or items in the two services, only the exact mappings are unknown. We call the newly established service "the target domain" and the other service "the source domain". One would then appreciate a model that, based only on the two sets of rating data, is able to identify not only the user mappings but the item mappings between the two domains. Such connection can then be used to enrich the understanding of the users and items in the target domain. One would be even happier if the model can exploit the likely "marginally correct" correspondence it found previously to boost the performance of a CF system built in the target domain. In our experiments, we provide 8 effective scenarios (e.g. whether user sets are completely overlapped, partially overlapped, or completely independent; or whether the ratings are disjoint across matrices or not) to verify the usefulness of such model.

Formally, we assume there are two homogeneous rating matrices: a target matrix $\mathbf{R}_1$ and an auxiliary data matrix $\mathbf{R}_2$. The rows and columns represent users and items, and each entry in the matrix represents the rating of an item given by a user. In these matrices, the two user sets and two item sets are overlapped to some extent, but we do not know how to associate the users and the items between the matrices. We want to answer the following two questions:

1. Given only $\mathbf{R}_1$ and $\mathbf{R}_2$, is it possible to find out the mapping of users and the mapping of items between them?

2. Given the noisy mapping we obtained, is it possible to transfer information from $\mathbf{R}_2$ to improve the rating prediction in $\mathbf{R}_1$?

The underlying assumption we made is that the rating behavior of both domains are similar; we call such rating data the "homogeneous". The problems sound hard because the user and item correspondences are both unknown, which prevents us from exploiting any supervised learning algorithm. If one side is already matched (e.g. if the items are matched), then one can use approaches like nearest neighbor to match the other side (e.g. users), as discussed in [13]. Unfortunately, such method cannot be applied when the columns and rows of the rating matrices both have gone through an unknown permutation process.

KDD'14, August 24–27, 2014, New York, NY, USA.

http://dx.doi.org/10.1145/2623330.2623657.

Fortunately, the matching problem is not impossible to solve. Recall the underlying assumption of collaborative filtering: similar users shall rate items similarly and similar items shall receive similar ratings from users. Viewing collaborative filtering as a matrix completion process, the above assumption actually suggests the rows/columns of the rating matrix are indeed dependent. That is, the matrix is of low-rank. The low-rank assumption has been validated by experiments. Owing to this assumption, researchers have proposed a family of strategies based on matrix factorization (MF), which has proven to be one of the most successful solutions for collaborative filtering [6, 1].

Extending from the concept of CF, our key hypothesis is that if two users rated some items similarly in one domain, they shall rate similar items similarly in the other domain. The low-rank assumption on both matrices plays a key role to the solution we propose in this paper. Our idea is to find a way to transform the incomplete matrices $\mathbf{R}_1$ and $\mathbf{R}_2$ into low-rank approximation, and match users/items in the latent space. One plausible solution is to perform matrix factorization on $\mathbf{R}_1$ and $\mathbf{R}_2$ then try to match users/items in the latent space. Figure 1 shows an illustrative example.

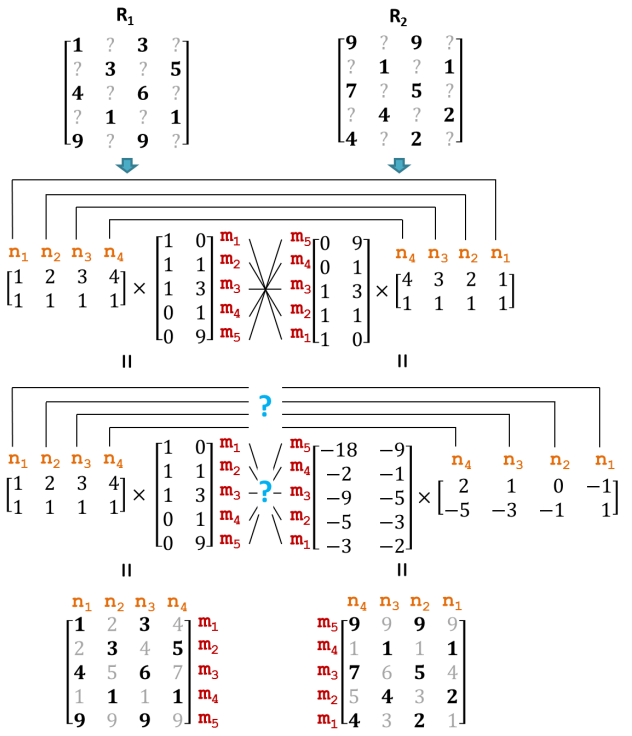


Figure 1: User and Item Matching on Low-Rank Matrices

At the first glance, it seems non-trivial to find the relationship between $\mathbf{R}_1$ and $\mathbf{R}_2$. Assuming that both matrices are of low-rank, they can be factorized into a user-latent matrix and item-latent matrix. Then it might be possible to match users based on the two user-latent matrices, as shown in the first decomposition in Figure 1. Unfortunately, matrix factorization is not unique, so $\mathbf{R}_1$ and $\mathbf{R}_2$ can be factorized using different latent space, prohibiting any further matching of users or items (see the second decomposition in Figure 1).

Another plausible idea is to exploit the idea of singular value decomposition (SVD) on the rating matrices $\mathbf{R}_1$ and $\mathbf{R}_2$, and then match users and items based on the singular vectors. This paper proposes a trick that allows us to obtain singular value decomposition on an incomplete matrix. However, the solution of SVD is still not unique, i.e. sign difference may exist, and the decomposition is unstable when noise occurs. We then propose a greedy-based algorithm that solves the sign problem and searches the nearest neighbors in the latent space. Then, we propose to adjust the matching results through a process that minimizes the prediction error on the existing ratings of $\mathbf{R}_1$ using $\mathbf{R}_2$, making the matching results more resistant to noise. Experiments show that our method can accurately find the correspondence given clean low-rank data, while achieving much higher matching accuracy than other competitors if given real, noisy data. Finally, based on the discovered matching outcome, we propose a transfer learning approach that transfers the rating information from $\mathbf{R}_2$ to $\mathbf{R}_1$ to boost the performance of the recommendation system in $\mathbf{R}_1$. We conduct extensive experiments including a variety of scenarios to verify the effectiveness of our model.

2. RELATED WORK

2.1 User Identification

User identification has been studied under different settings by different research communities. In natural language processing, author identification [2] is formulated as a text categorization problem, in which a supervised learning model is built based on features extracted from the authors' manuscripts. In social network analysis, Narayanan and Shmatikov [14] propose to link two social networks and de-anonymize users by first identifying some seed nodes and then iteratively propagating the mapping.

Some approaches have recently been proposed to link users from different social media sites. Zafarani and Liu propose MOBIUS for cross-media user identification [20]. Based on features extracted from user names, they learn a classification model to determine whether an account belongs to a certain person. Liu et al. [9] associates users with an unsupervised approach by calculating the rareness of the names. When a rare name (such as pennystar881) occurs on different websites, it is very likely that the two accounts are owned by the same person. Yuan et al. [19] propose another unsupervised approach to link users. They find that some users may explicitly display their other accounts on the profile page or disclose the account links when they share a post across websites. They design an algorithm to automatically capture all such information to link users.

Most of the studies differ from us because our model relies on rating data, and we do not assume there are any labeled pairs available for training. The closest work to us is [13], in which Narayanan and Shmatikov study how to de-anonymize users from rating data. In their setup the item mapping is known, which makes the task easier than the one we are trying to solve. Moreover, our model embeds itself a second objective: to improve the rating prediction in the target matrix.

2.2 Transfer Learning in Collaborative Filtering Given Correspondence

Many models have been proposed for transfer learning in collaborative filtering [17, 11, 18, 21, 15, 16, 4, 3]. The common goal is to transfer information across several data matrices. Most of the models assume that there is a one-to-one correspondence between users or between items across domains.

For example, in collective matrix factorization [17], there is a rating matrix (users by movies) and a label matrix (genres by movies) for transferring. The shared movie side then becomes the bridge for transferring, as they assume the latent factors of the movies are similar. Similar ideas can be applied on matrices of different time

frames [18], on binary versus numerical matrices [15, 16], or on a rating matrix and a social relation matrix [11, 4].

To sum up, the above models transfer information across two or more heterogeneous data matrices based on restrictions that the latent factors of the shared user side or item side are similar. Our setting is quite the opposite: the data matrices are homogeneous while both the user and item correspondence are unknown.

2.3 Transfer Learning in Collaborative Filtering When Correspondence is Unknown

We only find two models that transfer information between rating matrices without exploiting user and item correspondences and will discuss them in details here.

Codebook transfer (CBT) [7] and rating-matrix generative model (RMGM) [8] transfer information between two rating matrices without assuming user correspondence or item correspondence. The basic assumption of the two models are of the form:

$$\mathbf{R}_1 \approx \mathbf{U}_1\mathbf{B}\mathbf{V}_1^T,$$
$$\mathbf{R}_2 \approx \mathbf{U}_2\mathbf{B}\mathbf{V}_2^T,$$

where $\mathbf{B}$ is a shared K by K matrix. The major difference between these models and our model is that these models do not assume user correspondence or item correspondence between $\mathbf{R}_1$ and $\mathbf{R}_2$. Instead, they assume that $\mathbf{R}_1$ and $\mathbf{R}_2$ have homogeneous rating pattern; they enforce constraints on $\mathbf{U}$'s and $\mathbf{V}$'s while assuming the homogeneous pattern matrix, $\mathbf{B}$, is shared across $\mathbf{R}_1$ and $\mathbf{R}_2$. However, the constraints make the optimization process complicated and limit the expressive power of the model.

In CBT, $\mathbf{U}$ and $\mathbf{V}$ are constrained to be 0-1 matrices, and there can only be one entry with value 1 in each row. After adding the constraints, $\mathbf{R} \approx \mathbf{U}\mathbf{B}\mathbf{V}^T$ can be viewed as a co-clustering process that simultaneously divide users and items into groups. Assume a user is in i-th user group and an item is in j-th item group, then the rating of such user and item (r_{ij} in $\mathbf{R}$) is predicted to be the value of corresponding group rating value in $\mathbf{B}$ (b_{ij}). Thus, the formulation of CBT is actually associating groups of users (items) in $\mathbf{R}_1$ with groups of users (items) in $\mathbf{R}_2$.

However, the hard clustering constraint greatly reduces the expressive power of CBT; many lower-rank matrices cannot be factored under such constraint. Furthermore, to solve CBT, the optimization process requires the auxiliary matrix $\mathbf{R}_2$ to be fully observed, or it has to fill in missing entries with data mean before factorization. Since there are much more missing entries than observed ones in a sparse rating matrix, filling in the missing values manually can seriously bias the model.

RMGM relaxes the constraints in CBT from hard clustering to soft clustering by using a probabilistic graphical model, and it does not require $\mathbf{R}_2$ to be fully observed. To be more specific, the joint probability defined in RMGM is

$$P(r, m^{(i)}, n^{(i)}, C_m, C_n) = P(m^{(i)}|C_m)P(n^{(i)}|C_n)P(r|C_m, C_n)P(C_m)P(C_n),$$

and the rating prediction is

$$\sum_r r \sum_{k_1,k_2} P(r|C_m = k_1, C_n = k_2)P(C_m = k_1|m^{(i)})P(C_n = k_2|n^{(i)}),$$

where $m^{(i)}$ and $n^{(i)}$ are user and item in i-th domain, and C_m and C_n are the cluster for the user and item, respectively. If we rewrite the rating prediction as

$$\sum_{k_1,k_2} \left(P(C_m = k_1|m^{(i)})P(C_n = k_2|n^{(i)}) \sum_r rP(r|C_m = k_1, C_n = k_2) \right),$$

the term $\sum_r rP(r|C_m, C_n)$ is similar to $\mathbf{B}$ in CBT. $P(C_m|m^{(i)})$ and $P(C_n|n^{(i)})$ are similar to $\mathbf{U}_i$ and $\mathbf{V}_i$, respectively. When the rating prediction is perfect, $\mathbf{R} = \mathbf{U}_i\mathbf{B}\mathbf{V}_i^T$.

Even though RMGM relaxes the hard clustering constraints in CBT, constraints still exist: the elements in each row of $\mathbf{U}$ and $\mathbf{V}$ must be nonnegative and sum up to 1. This again limits the expressive power of the model and complicate the optimization task. Besides, the objective function of RMGM is the self-defined likelihood. It deviates from common evaluation measures such as root mean square error. Consider the following example:

$$\mathbf{R} = \begin{bmatrix} 1 & 1 \\ 3 & 3 \\ 2 & 2 \end{bmatrix}, \mathbf{U} = P(C_m|m^{(i)}) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0.5 & 0.5 \end{bmatrix}, \mathbf{V}^T = P(C_n|n^{(i)})^T = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix},$$

$$\mathbf{B} = \textstyle\sum_r rP(r|C_m, C_n) = 1 \times \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} + 2 \times \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} + 3 \times \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 3 & 3 \end{bmatrix}.$$

When we evaluate this model by root mean square error, the error is 0 because $\mathbf{R}$ is exactly the product of the three matrices. However, the data likelihood defined by RMGM is also 0 because the term $P(r = 2|C_m, C_n)$ is always 0. Thus, RMGM will try to find other parameters to fit the data, but they are not necessarily minimizing the prediction error. Nevertheless, RMGM is considered as the state-of-the-art and will be our main competitor in the evaluation.

In conclusion, past models do not try to solve correspondence and impose extra constraints in the factorization equation. Our model directly solves the correspondence problem and leverages such information to transfer across matrices.

3. METHODOLOGY

Given two partially observed rating matrices $\mathbf{R}_1$ and $\mathbf{R}_2$, we made the following assumptions:

1. There are some common users and common items across $\mathbf{R}_1$ and $\mathbf{R}_2$, but we do not know the correspondence.
2. The two matrices represent the same homogeneous domain. That is, if we can link the common users/items (rows/columns) between $\mathbf{R}_1$ and $\mathbf{R}_2$, we can merge them into one single matrix (Figure 2), and such matrix is likely to be low-rank which allows the collaborative filtering models to perform recommendation.

To find out the correspondence between $\mathbf{R}_1$ and $\mathbf{R}_2$, intuitively we want to solve

$$\mathbf{R}_1 \approx \mathbf{G}_{\text{user}}\mathbf{R}_2\mathbf{G}_{\text{item}}^T,$$

where $\mathbf{G}_{\text{user}}$ and $\mathbf{G}_{\text{item}}$ are 0-1 matrices that represent user and item correspondences. The symbol $\approx$ implies the corresponding entries in $\mathbf{R}_1$ and $\mathbf{G}_{\text{user}}\mathbf{R}_2\mathbf{G}_{\text{item}}^T$ shall be as similar as possible. The apparent challenge is that the rating matrices are partially observed, and very few entries are observed in both $\mathbf{R}_1$ and $\mathbf{R}_2$. Thus, we want to modify the above equation to deal with this problem.

We want to exploit the low-rank property of the rating matrices to solve the problem. To do so, we replace the original rating matrix by its fully filled low-rank approximation (i.e. no missing values). We propose the following two objective functions:

1. $\hat{\mathbf{R}}_1 \approx \mathbf{G}_{\text{user}}\hat{\mathbf{R}}_2\mathbf{G}_{\text{item}}^T$
2. $\mathbf{R}_1 \approx \mathbf{G}_{\text{user}}\hat{\mathbf{R}}_2\mathbf{G}_{\text{item}}^T$

We want to find $\mathbf{G}$'s that satisfies the above criteria. The symbol $\hat{\mathbf{R}}$ stands for the low-rank approximation of $\mathbf{R}$. In Equation 1, we hope to map the low-rank approximation $\mathbf{R}_2$ to the low-rank approximation of $\mathbf{R}_1$. In Equation 2, we hope the observed entries

$$\mathbf{R}_1 = \begin{bmatrix} \mathbf{1} & ? & ? & ? \\ ? & \mathbf{2} & \mathbf{3} & ? \\ \mathbf{1} & ? & ? & ? \\ ? & \mathbf{2} & \mathbf{3} & \mathbf{4} \end{bmatrix}, \quad \mathbf{R}_2 = \begin{bmatrix} \mathbf{1} & \mathbf{2} & ? & ? \\ \mathbf{1} & ? & ? & \mathbf{4} \\ ? & ? & \mathbf{3} & ? \\ ? & \mathbf{2} & ? & \mathbf{4} \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} \mathbf{1} & \mathbf{2} & ? & ? \\ \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} \\ \mathbf{1} & ? & \mathbf{3} & ? \\ ? & \mathbf{2} & \mathbf{3} & \mathbf{4} \end{bmatrix}$$

Figure 2: if the correspondence between $\mathbf{R}_1$ and $\mathbf{R}_2$ are known (e.g. the i-th column/row of $\mathbf{R}_1$ are mapped to the i-th column/row of $\mathbf{R}_2$), then $\mathbf{R}_1$ and $\mathbf{R}_2$ can be merged into one denser matrix.

in $\mathbf{R}_1$ can be matched by the corresponding entries of the low-rank approximation of $\mathbf{R}_2$. In theory we can swap $\mathbf{R}_1$ and $\mathbf{R}_2$ in Equation 2, but we chose to match the observation of $\mathbf{R}_1$ since it is the target matrix which we want to improve the rating prediction later on.

Optimizing the first objective function is easier but yields less accurate solutions, while the second objective function is more reliable but harder to solve. Therefore, our complete matching algorithm solves the two equations sequentially: the matching obtained from the first equation is used as initialization and to reduce the search space of second objective. We describe our solution to Equation 1 in Section 3.1. The main idea is to conduct singular value decomposition on the incomplete matrix $\hat{\mathbf{R}}$. We also design a greedy search on the latent factors for user matching and item matching. Section 3.2 solves Equation 2 in a way similar to two-block coordinate descent. At last, in Section 3.3, we describe our transfer learning model that exploits the matching result to boost the quality of rating prediction on the target domain.

In the following discussion, we do not constrain correspondence matrix $\mathbf{G}$ to be pure permutation matrix to simplify the optimization process. We only impose constraints on each row – such that one entry is 1 and the rest are all 0s – but not on columns. In other words, for each user/item in $\mathbf{R}_1$, we want to find its most similar neighbor in $\mathbf{R}_2$, rather than enforce the 1-to-1 mapping between users/items. This relaxation also makes our model more general as it can then handle situations with low or even no overlapping across domains.

3.1 Latent Space Matching

3.1.1 *Singular Value Decomposition*

The fundamental idea of our latent space matching approach is based on the uniqueness of singular value decomposition (SVD). Given a completely observed matrix $\mathbf{R}$, the singular value decomposition is $\mathbf{R} = \mathbf{U}\mathbf{D}\mathbf{V}^T$, where $\mathbf{D}$ is diagonal and $\mathbf{U}$ and $\mathbf{V}$ are unitary. We know that $\mathbf{D}$ is unique for every $\mathbf{R}$ when d_i's are sorted. Moreover, when all the singular values are distinct, the singular vectors ($\mathbf{U}$ and $\mathbf{V}$) are unique except sign differences. We can always multiply a column of U and the corresponding column of V by -1 and they become another valid solution for singular value decomposition.

The sign difference is indeed an important issue which we will address later. Assuming the sign problem is solved, it is possible to obtain a unique latent vector to represent any user or item. Then we can solve the user/item matching problem by conducting nearest neighbor search on the latent factors.

3.1.2 *From Matrix Factorization to Singular Value Decomposition*

Conventional singular value decomposition solvers cannot be applied to decompose the partially observed rating matrix. In collaborative filtering, matrix factorization (MF)[6, 12] – an unconstrained two-factor decomposition – is usually used. Here we provide a simple yet efficient process to transform unconstrained matrix factorization to a form of singular value decomposition.

Let $\mathbf{R}$ be a rating matrix and $\mathbf{W}$ an indicator matrix, where $\mathbf{W}_{ij} = 1$ means user i rated item j or 0 otherwise. Let M denote the number of users, N be the number of items, and K be the parameter that estimates the rank. We omit the regularization term for clarity. Then the objective of matrix factorization is to find $\mathbf{P}_{M\times K}$ and $\mathbf{Q}_{N\times K}$ such that

$$\min_{P,Q} \|\mathbf{W} \odot \left(\mathbf{R} - \mathbf{P}\mathbf{Q}^T\right)\|^2_{\text{Fro}},$$

where $\odot$ is the Hadamard (entrywise) product and Fro stands for Frobenius norm. The $\mathbf{P}$ and $\mathbf{Q}$ are usually solved by stochastic gradient descent or alternating least square [6].

Singular value decomposition is composed of $\mathbf{U}_{M\times K}$, $\mathbf{D}_{K\times K}$ and $\mathbf{V}_{N\times K}$ such that

$$\begin{aligned} \min_{U,V,D} \quad & \|\mathbf{W} \odot \left(\mathbf{R} - \mathbf{U}\mathbf{D}\mathbf{V}^T\right)\|^2_{\text{Fro}} \\ \text{subject to} \quad & \mathbf{U}^T\mathbf{U} = \mathbf{I}, \mathbf{V}^T\mathbf{V} = \mathbf{I}, \mathbf{D} \text{ is diagonal} \end{aligned}$$

Our trick starts from performing factorization on $\mathbf{R}$ to obtain $\mathbf{P}$ and $\mathbf{Q}$, and then transform $(\mathbf{P}, \mathbf{Q})$ to $(\mathbf{U}, \mathbf{D}, \mathbf{V})$ by applying conventional SVD on $\mathbf{P}$ and $\mathbf{Q}$. Note that $\mathbf{P}$ and $\mathbf{Q}$ are not partially observed so we can exploit conventional methods to obtain the decomposition. [1] The process is illustrated in Figure 3. Assume ($\mathbf{U}_P$, $\mathbf{D}_P$, $\mathbf{V}_P$) is the SVD of $\mathbf{P}$ and ($\mathbf{U}_Q$, $\mathbf{D}_Q$, $\mathbf{V}_Q$) is the SVD of $\mathbf{Q}$. Let ($\mathbf{U}_X$, $\mathbf{D}_X$, $\mathbf{V}_X$) be the SVD of $(\mathbf{D}_P\mathbf{V}_P^T\mathbf{V}_Q\mathbf{D}_Q^T)$. Then,

$$\mathbf{P}\mathbf{Q}^T = \mathbf{U}_P\mathbf{D}_P\mathbf{V}_P^T\left(\mathbf{U}_Q\mathbf{D}_Q\mathbf{V}_Q^T\right)^T = \mathbf{U}_P\mathbf{U}_X\mathbf{D}_X\mathbf{V}_X^T\mathbf{U}_Q^T.$$

We have obtained the SVD for $\mathbf{P}\mathbf{Q}^T$ as $\mathbf{U}\mathbf{D}\mathbf{V}^T$, where $\mathbf{U} = \mathbf{U}_P\mathbf{U}_X$, $\mathbf{V} = \mathbf{U}_Q\mathbf{V}_X$ and $\mathbf{D} = \mathbf{D}_X$.

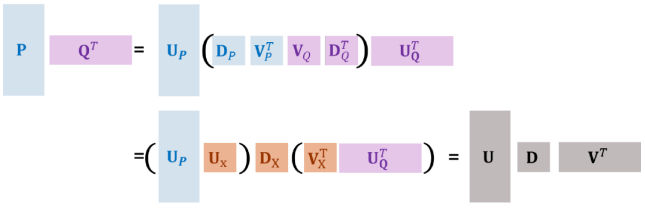


Figure 3: From MF to SVD

One practical issue to be aware of is that SVD normalizes the norm of each column of $\mathbf{U}$ and $\mathbf{V}$ to 1, so the values of $\mathbf{U}$ and $\mathbf{V}$ tend to be smaller for larger matrices. Since $\mathbf{R}_1$ and $\mathbf{R}_2$ may be of different sizes, it is necessary to scale the values in $\mathbf{U}$ and $\mathbf{V}$ to make sure later on when we want to compare $\mathbf{U}_1$ with $\mathbf{U}_2$ and $\mathbf{V}_1$ with $\mathbf{V}_2$, they are of the same scale. $\mathbf{U}$, $\mathbf{D}$ and $\mathbf{V}$ are re-scaled as

$$\mathbf{U} = \mathbf{U} \times \sqrt{M}, \mathbf{D} = \mathbf{D} \times \sqrt{M} \times \sqrt{N}, \mathbf{V} = \mathbf{V} \times \sqrt{N},$$

where M and N are scalars representing the number of users and items respectively.

[1] the matrix is highly asymmetric so thin SVD (economical SVD) is used and the decomposition only takes seconds.

3.1.3 *Objective Function of Latent Space Matching*

Remember our first goal is to find $\mathbf{G}_{\text{user}}$ and $\mathbf{G}_{\text{item}}$ such that

$$\hat{\mathbf{R}}_1 \approx \mathbf{G}_{\text{user}} \hat{\mathbf{R}}_2 \mathbf{G}_{\text{item}}^T.$$

From the previous section, we have obtained the low-rank approximation of two matrices as $\hat{\mathbf{R}}_1 = \mathbf{U}_1\mathbf{D}_1\mathbf{V}_1^T$ and $\hat{\mathbf{R}}_2 = \mathbf{U}_2\mathbf{D}_2\mathbf{V}_2^T$. The above equation becomes

$$\mathbf{U}_1\mathbf{D}_1\mathbf{V}_1^T \approx \mathbf{G}_{\text{user}}\mathbf{U}_2\mathbf{D}_2\mathbf{V}_2^T\mathbf{G}_{\text{item}}^T.$$

Then, we can decouple the user/item matching problem because of the uniqueness of singular value decomposition. When the decomposition is unique, the latent factors of the same user in $\mathbf{R}_1$ and $\mathbf{R}_2$ should be similar, and the latent factors of the same item should be similar, too. Thus, we can decouple the above equation into two matching problems:

$$\min_{\mathbf{G}_{\text{user}},\mathbf{S}_{\text{user}}} \|\mathbf{U}_1\mathbf{D}_1^{0.5} - \mathbf{G}_{\text{user}}\mathbf{U}_2\mathbf{D}_2^{0.5}\mathbf{S}_{\text{user}}\|_{\text{Fro}}^2$$

$$\min_{\mathbf{G}_{\text{item}},\mathbf{S}_{\text{item}}} \|\mathbf{V}_1\mathbf{D}_1^{0.5} - \mathbf{G}_{\text{item}}\mathbf{V}_2\mathbf{D}_2^{0.5}\mathbf{S}_{\text{item}}\|_{\text{Fro}}^2$$

We have mentioned that singular value decomposition is unique except the sign of column vectors. Thus, in the equation we add a sign matrix $\mathbf{S}$, a small diagonal matrix (K by K) where each diagonal element is either 1 or -1. Since the formulations of user matching and item matching are identical, here we will discuss user side only.

When $\mathbf{S}$ is fixed, the previous equation becomes

$$\min_{\mathbf{G}_{\text{user}}} \|\mathbf{Z}_1 - \mathbf{G}_{\text{user}}\mathbf{Z}_2\|_{\text{Fro}}^2,$$

where

$$\mathbf{Z}_1 = \mathbf{U}_1\mathbf{D}_1^{0.5}, \mathbf{Z}_2 = \mathbf{U}_2\mathbf{D}_2^{0.5}\mathbf{S}.$$

Now all rows of $\mathbf{G}$ become independent because we only enforce constraints on rows but not columns. Thus, it becomes a simple nearest neighbor search problem. For each row in $\mathbf{Z}_1$, the goal is to select the most similar row in $\mathbf{Z}_2$ (illustrated in Figure 4). It takes $\Theta(M_1M_2K)$ time to solve $\mathbf{G}$, where M_1 and M_2 are the number of users in $\mathbf{R}_1$ and $\mathbf{R}_2$, respectively.

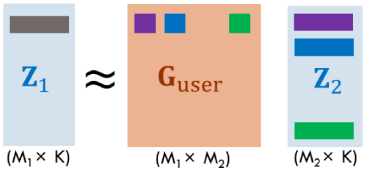


Figure 4: Subroutine when $\mathbf{S}$ is fixed – Nearest Neighbor Search. The purple entry in $\mathbf{G}$ becomes 1 when the purple row in $\mathbf{Z}_2$ is the nearest neigbor to the grey row in $\mathbf{Z}_1$.

Unfortunately, for the problem we are facing, the sign matrix is unknown and there are 2^K possible combinations. We therefore design a greedy approach to search for a plausible solution. Assume $d_1 \geq d_2 \geq \cdots \geq d_K$, where d_i is a singular value. In the k-th iteration, we try both $s_k = 1$ and $s_k = -1$ and solve $\mathbf{G}$ twice. We choose the s_k and $\mathbf{G}$ that results in a smaller objective value in nearest neighbor. After determining s_k, we move on to the next iteration and solve s_{k+1}. We illustrate how to solve $\mathbf{S}$ in Figure 5.

Ideally, we can simply use the $\mathbf{G}$ obtained in the last iteration as our matching result. However, smaller singular values and their singular vectors are more noisy, resulting in a noisy $\mathbf{G}$. That says,

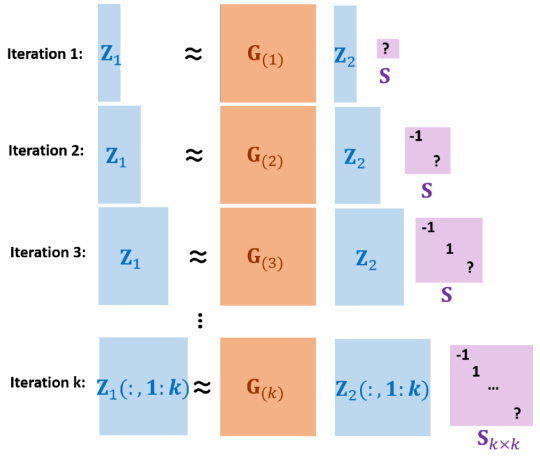


Figure 5: Sign Solving Process

picking the matching function $\mathbf{G}$ in the final iteration might not be the best choice. Here we propose a simple strategy to perform such "dimension selection" to pick the most suitable $\mathbf{G}_{\text{user}}$ and $\mathbf{G}_{\text{item}}$. Our goal is to select one $\mathbf{G}$ from $\mathbf{G}_{(1:K)}$, where the $\mathbf{G}_{(1:K)}$ is the output in every iteration while solving $\mathbf{S}$. We collect all pairs of $\mathbf{G}_{\text{user}}$ and $\mathbf{G}_{\text{item}}$, and then select the pair that leads to the best rating prediction in $\mathbf{R}_1$. We choose the $\mathbf{G}_{\text{user}}$ and $\mathbf{G}_{\text{item}}$ that minimize

$$\|\mathbf{W}_1 \odot \left(\mathbf{R}_1 - \mathbf{G}_{\text{user}}\mathbf{U}_2\mathbf{D}_2\mathbf{V}_2\mathbf{G}_{\text{item}}^T\right)\|_{\text{Fro}}^2.$$

The complete procedure for our latent space matching is described in Algorithm 1.

Finally, we enter the discussion about time complexity. In the MATCHING function of Algorithm 1, we actually maintain a distance table T that records all pair distance between $\mathbf{Z}_1$ and $\mathbf{Z}_2$ for fast look-up. After k iterations, T records the distance calculated using the latent dimension 1 to k, so in the next iteration we can quickly update the distance by considering the latent dimension $k+1$ only. Thus, the complete time complexity of the MATCHING function in Algorithm 1 is still $\Theta(M_1M_2K)$ for user matching and $\Theta(N_1N_2K)$ for item matching. The dimension selection part takes $\Theta(K \times L \times K) = \Theta(K^2L)$, where L is the number of ratings in $\mathbf{R}_1$.

3.1.4 *Remarks on Latent Space Matching*

The main advantage of the proposed latent space matching method is that it decouples user/item matching problem. In the ideal case where the matrix is low-rank, latent space matching is capable of producing perfect matching (our experiment will show this). However, there are some potential concerns when noise exists. First, latent space matching is based on matrix factorization, which is only an approximation of the original rating matrix. Second, when some singular values are close and the intensity of noise increases, the corresponding singular vectors are mixed together. In such degenerate cases, singular value decomposition is not unique.

It is known that singular value decomposition is equivalent to principle component analysis when the data mean is zero. We can illustrate the degenerate case in Figure 6(a) and Figure 6(b). Figure 6(a) is an example where the singular value decomposition is unique. The major axis of the ellipse represents the first singular vector and the minor axis represents the second singular vector. In contrast, Figure 6(b) shows a degenerate case. We cannot identify the major axis in the graph and the singular value decomposition of the data matrix is not unique.

Algorithm 1 Latent Space Matching

Require: $\mathbf{U}_1, \mathbf{D}_1, \mathbf{V}_1, \mathbf{U}_2, \mathbf{D}_2, \mathbf{V}_2$ and $\mathbf{R}_1$
Ensure: $\mathbf{G}^*_{\text{user}}$ and $\mathbf{G}^*_{\text{item}}$
  function NN($\mathbf{Z}_1, \mathbf{Z}_2, T$)
    ▷ perform nearest neighbor search
    ▷ T is the distance cache for speed-up
    $\alpha = \min_{\mathbf{G}} \|\mathbf{Z}_1 - \mathbf{G}\mathbf{Z}_2\|^2_{\text{Fro}}$
    $\mathbf{G} = \underset{\mathbf{G}}{\text{argmin}} \|\mathbf{Z}_1 - \mathbf{G}\mathbf{Z}_2\|^2_{\text{Fro}}$
    return $(\alpha, \mathbf{G})$
  end function
  function MATCHING($\mathbf{Z}_1, \mathbf{Z}_2$)
    ▷ T is a cache for speed-up
    initialize all elements of the distance table T to 0
    for k = 1 to K **do**
      let $s_k = +1$
      $(\alpha_+, \mathbf{G}_+) = \text{NN}(\mathbf{Z}_{1\ (:,1:k)}, \mathbf{Z}_{2\ (:,1:k)}\mathbf{S}_{(1:k,1:k)}, \text{T})$
      let $s_k = -1$
      $(\alpha_-, \mathbf{G}_-) = \text{NN}(\mathbf{Z}_{1\ (:,1:k)}, \mathbf{Z}_{2\ (:,1:k)}\mathbf{S}_{(1:k,1:k)}, \text{T})$
      if $\alpha_+ \leq \alpha_-$ **then**
        $\mathbf{G}_{(k)} = \mathbf{G}_+, s_k = +1$
      else
        $\mathbf{G}_{(k)} = \mathbf{G}_-, s_k = -1$
      end if
      update T according to $\mathbf{G}_{(k)}$
    end for
    return $\mathbf{G}_{(1)}, \cdots, \mathbf{G}_{(K)}$
  end function
  $\mathbf{G}_{\text{user}\ (1:K)}$ = MATCHING($\mathbf{U}_1\mathbf{D}_1^{0.5}, \mathbf{U}_2\mathbf{D}_2^{0.5}$)
  $\mathbf{G}_{\text{item}\ (1:K)}$ = MATCHING($\mathbf{V}_1\mathbf{D}_1^{0.5}, \mathbf{V}_2\mathbf{D}_2^{0.5}$)
  for k = 1 to K **do** ▷ dimension selection
    $\text{error}_k = \|\mathbf{W}_1 \odot \left(\mathbf{R}_1 - \mathbf{G}_{\text{user}\ (k)}\mathbf{U}_2\mathbf{D}_2\mathbf{V}_2\mathbf{G}^T_{\text{item}\ (k)}\right)\|^2_{\text{Fro}}$.
  end for
  $(\text{error}_{\text{min}}, k_{\text{min}}) = \min_k \text{error}_{1:K}$
  $\mathbf{G}^*_{\text{user}} = \mathbf{G}_{\text{user}\ (k_{\text{min}})}$
  $\mathbf{G}^*_{\text{item}} = \mathbf{G}_{\text{item}\ (k_{\text{min}})}$

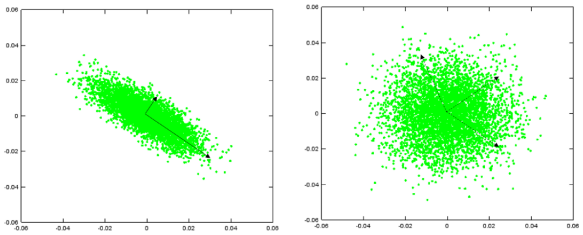
Such deficiency can be observed in the experiment, which leads to our design of a refinement approach which we will describe in the next section.

3.2 Matching Refinement through Optimization

We talked about how the user and item correspondences, $\mathbf{G}_{\text{user}}$ and $\mathbf{G}_{\text{item}}$, can be obtained through latent space matching. Here we propose a method to further refine them using the previously learned $\mathbf{G}_{\text{user}}$ and $\mathbf{G}_{\text{item}}$. Since our ultimate goal is to transfer knowledge from $\mathbf{R}_2$ to help rating prediction of $\mathbf{R}_1$, a reasonable way to evaluate the quality of $\mathbf{G}_{\text{user}}$ and $\mathbf{G}_{\text{item}}$ is to check whether $\mathbf{G}_{\text{user}}\mathbf{P}_2\mathbf{Q}_2^T\mathbf{G}_{\text{item}}^T$ can reproduce the corresponding ratings in $\mathbf{R}_1$, where $\mathbf{P}_2\mathbf{Q}_2^T$ is the low-rank approximation of $\mathbf{R}_2$. Therefore, we want to adjust $\mathbf{G}_{\text{user}}$ and $\mathbf{G}_{\text{item}}$ to satisfy the following equation

$$\min_{\mathbf{G}_{\text{user}}, \mathbf{G}_{\text{item}}} \|\mathbf{W}_1 \odot \left(\mathbf{R}_1 - \mathbf{G}_{\text{user}}\mathbf{P}_2\mathbf{Q}_2^T\mathbf{G}_{\text{item}}^T\right)\|_{\text{Fro}}.$$

Note that in our previous latent space matching, for each user/item we identify its nearest neighbor. Here we propose to record the top nearest candidates for each user and item, and reevaluate their quality based on the above equation. Assume in latent space matching, we have recorded the top C neighboring users/items. Then the $\mathbf{G}_{\text{user}}$ and $\mathbf{G}_{\text{item}}$ we want to solve now become $M_1 \times C$ and $N_1 \times C$ matrices, respectively. M_1 and N_1 are the number of users and the number of items in $\mathbf{R}_1$.



(a) Normal Data (b) A Degenerate Case

Figure 6: If the data distribution is like (a), the principal component is the major axis. In (b) we cannot identify the principal component.

We solve $\mathbf{G}$ in a similar manner as two-block coordinate descent, except here the variables in $\mathbf{G}$ are discrete. When $\mathbf{G}_{\text{item}}$ is fixed, each row in $\mathbf{G}_{\text{user}}$ becomes independent, and finding the best $\mathbf{G}_{\text{user}}$ becomes a straightforward search problem. For each user in $\mathbf{R}_1$, we are finding the candidate (the corresponding row in $\mathbf{P}_2$) that minimizes the rating prediction error in the row of $\mathbf{R}_1$. It takes $\Theta(LCK)$ time to solve $\mathbf{G}_{\text{user}}$, where L is the number of ratings in $\mathbf{R}_1$.

It can be easily seen that this process leads to a local optimal solution as every time when we update $\mathbf{G}$, the objective value is guaranteed to become smaller. Our algorithm solves $\mathbf{G}_{\text{user}}$ and $\mathbf{G}_{\text{item}}$ alternately until both $\mathbf{G}$'s become stable. There is only a finite number of possibilities, so the algorithm will converge. Since the procedure only reaches local optimum, a good initialization is very important to guarantee the quality of the output. Our latent space matching serves such purpose. We set the initial value of $\mathbf{G}$ here to the $\mathbf{G}$ outputted in the original latent space matching.

3.3 Transferring Imperfect Matching to Predict Ratings on the Target Domain

Here we emphasize on transferring the matching information to the target domain to improve rating prediction. We modify matrix factorization [6, 12] for rating prediction. In the original matrix factorization model, the goal is to find $\mathbf{P}$ and $\mathbf{Q}$ to minimize

$$\|\mathbf{W} \odot \left(\mathbf{R} - \mathbf{P}\mathbf{Q}^T\right)\|^2 + \lambda\left(\|\mathbf{P}\|^2 + \|\mathbf{Q}\|^2\right).$$

Once $\mathbf{P}$ and $\mathbf{Q}$ are identified, we can multiply them to obtain the full matrix as the prediction. The norms here are Frobenius norm.

When the user and item correspondences are both given, let g_m be the corresponding user in $\mathbf{R}_2$ of user m in $\mathbf{R}_1$ and g_n be the corresponding item in $\mathbf{R}_2$ of item n in $\mathbf{R}_1$. We modify the original formula of matrix factorization by adding a "matching-based" regularization term:

$$\begin{aligned}\|\mathbf{W}_1 \odot \left(\mathbf{R}_1 - \mathbf{P}_1\mathbf{Q}_1^T\right)\|^2 + \|\mathbf{W}_2 \odot \left(\mathbf{R}_2 - \mathbf{P}_2\mathbf{Q}_2^T\right)\|^2 \\ +\lambda\left(\|\mathbf{P}_1\|^2 + \|\mathbf{Q}_1\|^2 + \|\mathbf{P}_2\|^2 + \|\mathbf{Q}_2\|^2\right) \\ +\beta\sum_m^{M_1} \arctan\left(\|\mathbf{P}_1(m,:) - \mathbf{P}_2(g_m,:)\|^2\right) \\ +\beta\sum_n^{N_1} \arctan\left(\|\mathbf{Q}_1(n,:) - \mathbf{Q}_2(g_n,:)\|^2\right)\end{aligned}$$

The extra regularization terms associated with β restrict the latent factors of matched users/items to be the same. Therefore, if the matching is correct, since the corresponding users or items are forced to align in their latent space, the extra information from the other domain can then be exploited.

Acknowledging the fact that our matching is not perfect, special care needs to be taken to prevent incorrect matching from hampering the prediction performance. Thus, we add an $\arctan$ function to alleviate the influence of outliers. When the matching results become unreliable, our model degenerates into regular matrix factorization model as β would become 0 after parameter selection. The objective is still differentiable and can be solved via standard approaches such as gradient descent.

4. EXPERIMENTS

4.1 Experimental Setup

Table 1: Statistics of $\mathbf{R}$

Dataset	# of Users and Items	Rating Scale	Sparsity
Low Rank	(20000, 10000)	real, [-1,1]	5%
Yahoo! Music	(20000, 10000)	integer, 0-100	5.4%

Our goal is to link the users and items in two homogeneous rating matrices, $\mathbf{R}_1$ and $\mathbf{R}_2$, and then use the corresponding information in $\mathbf{R}_2$ to improve rating prediction in $\mathbf{R}_1$. However, we cannot find two independent datasets that provide the ground truth for user/item mappings to evaluate our model. Fortunately, we can use a real dataset and split it into two rating sets as $\mathbf{R}_1$ and $\mathbf{R}_2$. Another advantage of using such strategy for evaluation is that we can then test different scenarios (i.e. different splitting condition) to evaluate the usefulness of our model under a variety of different assumptions. The user and item ids of $\mathbf{R}_2$ are randomly permuted, while the permutation (unknown to our model) becomes the ground truth of the correspondence.

We conduct experiments on a synthetic dataset and a real dataset, Yahoo! music dataset. The synthetic dataset is a noise-free low-rank matrix for verifying the soundness of different models. It is a rank-50 matrix, generated by the following MATLAB command

randn(20000,50)*diag(1.1.^[1:50])*randn(50,10000).

We sample 5% of it as $\mathbf{R}$ and linearly scale the minimum and maximum values to -1 and 1. Yahoo! music dataset has been used as the benchmark data in KDD Cup 2011 [1]. For both datasets, we take out a subset $\mathbf{R}$ and split it into $\mathbf{R}_1$ and $\mathbf{R}_2$. The statistics of $\mathbf{R}$ are listed in Table 1.

With the capability to control the mapping condition, now we are ready to test the effectiveness of our model under different assumptions. Below lists the conditions (see Figure 7 for details):

1. **Disjoint Split.** We assume a user will not rate the same item twice, so the same rating will not appear in both $\mathbf{R}_1$ and $\mathbf{R}_2$. Real world scenario is in ratings for durable goods. For example, assume $\mathbf{R}_1$ and $\mathbf{R}_2$ are two nearby retailers that sells laptops, and thus have similar customers and products. When a user buys a laptop in $\mathbf{R}_1$, he or she is unlikely to buy the same laptop again in the other store.
2. **Overlap Split.** We assume ratings given in $\mathbf{R}_1$ may or may not appear in $\mathbf{R}_2$. This is a very common situation. For example, given two supermarkets in the same area, a customer can buy identical or different products in both stores.
3. **Contained Split.** We assume $\mathbf{R}_2$ is more frequently visited and all rating information in $\mathbf{R}_1$ is also available in $\mathbf{R}_2$. This is an extreme case of overlap split when the overlap ratio is maximized, and it is presumably the easiest situation for linking users and items between $\mathbf{R}_1$ and $\mathbf{R}_2$.

For the three splits, we vary the overlap ratio for ratings, but we assume that every user and item in $\mathbf{R}_1$ can be found in $\mathbf{R}_2$ and vice versa. This assumption may not be true in some cases so we created two additional scenarios.

4. **Subset Split.** We assume the users and items in $\mathbf{R}_1$ are subsets of that in $\mathbf{R}_2$. This is a common situation when $\mathbf{R}_1$ and $\mathbf{R}_2$ are two stores selling same type of goods in the same area, where $\mathbf{R}_2$ is a much larger and potentially cheaper store that has more products for sale.
5. **Partial Split.** We assume the user/item set in $\mathbf{R}_1$ are partially overlapped with that in $\mathbf{R}_2$. This is also a very common situation that two stores have overlapping but not identical customers and products for sell. We want to first discover the mapping between the overlapped users/items, and then use such information to improve the rating prediction.

For subset split and partial split, we simply assume that their overlap ratio for ratings is similar to that of the overlap split.

Because $\mathbf{R}_1$ is the target domain in which we want to evaluate whether rating prediction accuracy can be boosted by knowledge transfer, we divide its data into training, validation and testing sets. $\mathbf{R}_2$, on the other hand, belongs to the source domain, thus we only need to divide it into training and validation set. Validation sets are used for parameter tuning in factorization models. Each of the two validation sets takes up 2.5% of the original data $\mathbf{R}$, while the testing set takes up 5%. Each of the validation and testing sets does not overlap with any other set so as to ensure the sanity of our experiment. The training sets of $\mathbf{R}_1$ and $\mathbf{R}_2$ are sampled from $\mathbf{R}$ according to the scenarios described above (Figure 7).

4.2 Competing Models

The first experiment is to evaluate the matching quality. To our knowledge, there has not yet been solutions proposed to utilize only incomplete rating information to perform user/item alignment, so we compare our method with other baseline models.

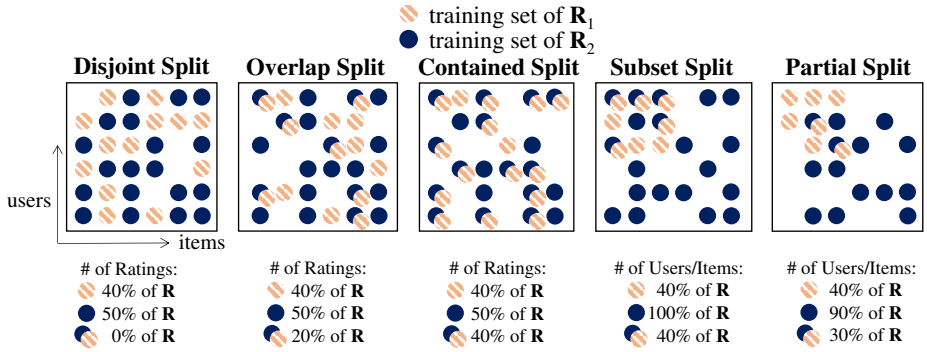
We define the following three baseline models. Here we only describe the user-matching procedure as the item-matching procedure is identical.

1. **Matching based on Regular Matrix Factorization.** We conduct the regular matrix factorization and then perform nearest neighbor matching based on Euclidean distances between the obtained latent factors.
2. **Matching based on User Mean/Item Mean.** Find nearest users for matching based on the mean of their ratings. In other words, users with similar rating averages are matched.
3. **Matching based on Rating Lists.** We first sort the ratings of each user from large to small into a list. For any two users, we align the two rating lists and use the Euclidean distance of these two vectors as the matching criteria. In other words, two users are paired if their sorted rating lists are similar.

As mentioned in the introduction section, latent factor matching from simple matrix factorization does not work. The solution of matrix factorization is not unique since

$$\mathbf{P}\mathbf{Q}^T = \mathbf{P}\mathbf{A}\mathbf{A}^{-1}\mathbf{Q}^T,$$

Figure 7: Illustration of Splits



Even when $\mathbf{R}_1 = \mathbf{R}_2$, we can still obtain very different $\mathbf{P}_1$ and $\mathbf{P}_2$. In fact, experiment results show that the first two baselines perform no better than random guess, so we omit them from the result table to save space. We found the third baseline that works much better than the first two, and therefore its results are included in our table.

Second, we conduct an experiment to evaluate the quality boost of rating prediction in $\mathbf{R}_1$ after transferring information from $\mathbf{R}_2$. There has not yet been much work on information transfer without knowing the underlying correspondence. Hence, for rating prediction we use rating-matrix generative model (RMGM) as the state-of-the-art for comparison.

4.3 Implementation Details

For our factorization models, we use gradient descent and backtracking line search to solve the objective function. The dimension K is fixed to 50, and the parameters λ and β are automatically selected by observing the error rate of the validation set. After selection, λ is determined to be 0 for noise-free low-rank dataset and 5 for Yahoo! music dataset, and β ranges from 0 to 400. Besides, we employ the data scaling and early stopping procedure. We scale the ratings in the training set to zero mean and unit variance, and scale the values back in the prediction phase; the training process stops when validation error starts to increase.

We implement two versions of rating-matrix generative model (RMGM) [8] for comparison. They can be solved by standard expectation-maximization algorithm. [2] The original RMGM uses categorical distribution for $P(r|C_m, C_n)$. However, categorical distribution does not reflect the ordinal relation among the rating values. Therefore, we implement another version of RMGM that uses Gaussian distribution. For both models, the original and Gaussian RMGMs, we set the latent dimension K to 50 (we find that a larger K leads to similar performance), and we conduct the same early stopping procedure. For Gaussian RMGM, we find that the variance of Gaussian is better set to a constant parameter that can be tuned by observing the error rate of the validation set. We have also conducted the data scaling process for Gaussian RMGM.

4.4 Results of User and Item Matching

In the matching process, the matching can output either the most likely candidate or a ranked list of candidates for each user/item in $\mathbf{R}_1$. Therefore, we can use accuracy as well as mean average precision (MAP) as the evaluation criteria. However, in the partial split, some users and items in $\mathbf{R}_1$ do not appear in $\mathbf{R}_2$. Thus, when evaluating the matching result of such scenario, we remove these users and items from consideration.

[2]An example code is provided by the author of RMGM: https://sites.google.com/site/libin82cn/

Table 2: Matching Result on the Low-Rank Dataset

	Rating List	Latent Space	Refinement
Disjoint Split			
Accuracy(user)	0.000	1.000	1.000
MAP(user)	0.003	1.000	1.000
Accuracy(item)	0.001	1.000	1.000
MAP(item)	0.007	1.000	1.000
Overlap Split			
Accuracy(user)	0.025	1.000	1.000
MAP(user)	0.048	1.000	1.000
Accuracy(item)	0.051	1.000	1.000
MAP(item)	0.089	1.000	1.000
Contained Split			
Accuracy(user)	0.538	1.000	1.000
MAP(user)	0.612	1.000	1.000
Accuracy(item)	0.703	1.000	1.000
MAP(item)	0.765	1.000	1.000
Subset Split			
Accuracy(user)	0.013	1.000	1.000
MAP(user)	0.029	1.000	1.000
Accuracy(item)	0.029	1.000	1.000
MAP(item)	0.058	1.000	1.000
Partial Split			
Accuracy(user)	0.007	1.000	1.000
MAP(user)	0.019	1.000	1.000
Accuracy(item)	0.018	1.000	1.000
MAP(item)	0.042	1.000	1.000

The matching results are shown in Table 2 and Table 3. The results on the noise-free low-rank dataset demonstrate the soundness of our approach. When there is no noise in the rating matrix, our latent space matching can precisely match all users and items by comparing the singular vectors regardless of the splits. On the other hand, the performance of baseline model (rating list) improves when more ratings are shared between domains , but the results are still far from perfect.

On Yahoo! music dataset, we see a similar trend: when more ratings are shared, the results are generally better. We observe that the contained split is easier to match than overlap split, while both are easier to match than disjoint split. The matching accuracy for the subset split and partial split are slightly worse (though still comparable) than that of the overlap split. It is because although the

Table 3: Matching Result on Yahoo! Music

	Rating List	Latent Space	Refinement
Disjoint Split			
Accuracy(user)	0.074	0.310	0.633
MAP(user)	0.126	0.419	0.717
Accuracy(item)	0.019	0.204	0.325
MAP(item)	0.047	0.325	0.463
Overlap Split			
Accuracy(user)	0.150	0.547	0.960
MAP(user)	0.226	0.652	0.973
Accuracy(item)	0.056	0.442	0.786
MAP(item)	0.107	0.578	0.859
Contained Split			
Accuracy(user)	0.419	0.851	0.997
MAP(user)	0.526	0.905	0.998
Accuracy(item)	0.323	0.815	0.975
MAP(item)	0.423	0.886	0.986
Subset Split			
Accuracy(user)	0.122	0.392	0.918
MAP(user)	0.190	0.510	0.941
Accuracy(item)	0.047	0.297	0.686
MAP(item)	0.090	0.433	0.780
Partial Split			
Accuracy(user)	0.109	0.350	0.871
MAP(user)	0.175	0.470	0.906
Accuracy(item)	0.043	0.272	0.573
MAP(item)	0.081	0.402	0.684

ratio of shared ratings is similar among these splits, some users and items can never be aligned in the subset split and partial split.

For all cases, the latent space matching we proposed already enjoys a significant boost comparing to the best baseline model, while the matching refinement through optimization further leads to great improvement over latent space matching. It is reasonable because there are some drawbacks of latent space matching, which have been discussed in Section 3.1.4.

4.5 Results for Rating Prediction by Transferring

Table 4: Rating Prediction (RMSE) on Yahoo! Music

	RMGM	Gaussian RMGM	MF	Proposed Approach
Disjoint Split	27.47	26.59	24.24	**23.34** †
Overlap Split	27.54	26.48	24.23	**23.49** †
Contained Split	27.58	26.53	24.29	**23.92** †
Subset Split	27.65	26.56	24.13	**23.40** †
Partial Split	27.57	26.51	24.21	**23.77** †

Those significantly better then the second best results are marked with †

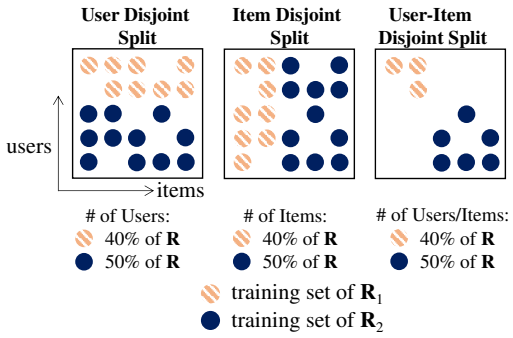
Next we want to evaluate whether our matching can indeed be used to improve the quality of a recommendation system in the target domain. We resort to the experiment on rating prediction and use root mean square error (RMSE) as the evaluation criterion. The hypothesis to be verified with this experiment is that despite the existence of user/item mis-matching, our model can still improve the prediction accuracy of the target domain after information transfer. Here we focus on the Yahoo! music dataset because in the noise-free low-rank dataset the RMSE before transferring is already close to 0.

The results are shown in Table 4. Gaussian RMGM outperforms the original RMGM, likely due to the fact that the actual rating itself does not follow categorical distribution. However, RMGM does not perform well in this experiment. One of the reason is that the likelihood function optimized by RMGM does not directly reflect the objective for evaluation (discussed in Section 2.3), while in our model minimizing the prediction error (i.e. RMSE in this experiment) is one of the direct objectives.

For all cases, our proposed approach leads to significant improvement over the original matrix factorization (MF) model, which is widely considered as one of the most effective single domain model. Interestingly, although the results in Table 3 show that the matching accuracy can be ranked as disjoint < overlap < contained split, results in Table 4 show that in terms of rating prediction, the improvement for the disjoint split is the best, while the improvement for the contained split is the worst. We believe this is because when there is less overlap of ratings, the amount of information $\mathbf{R}_2$ provides to $\mathbf{R}_1$ increases, and our solution is capable of leveraging the extra information to provide better prediction outcome.

4.6 When Users/Items Do Not Overlap

Figure 8: Illustration of Three New Splits



Finally, we want to test what happens when the original assumption of our model is violated: that is, what happens if the user sets or the item sets do not overlap at all. It implies the matching is always wrong and our matching algorithm can at best identify some "similar" users/items. Thus, we create three other splits and they are illustrated in Figure 8.

1. **User Disjoint Split.** The user sets of $\mathbf{R}_1$ and $\mathbf{R}_2$ are disjoint and the item sets are the same though the mapping is still unknown. It is also a common scenario in the real world. For example, there are two DVD-rental stores located in different states. Although they have similar items to be rented, the customer bases are completely disjoint.

2. **Item Disjoint Split.** The user sets are identical (without knowing the mapping), while the item sets are disjoint. For example, two near-by restaurants, one of which serves eastern food and the other serves western food, may share similar set of customers.

3. **User-Item Disjoint Split.** The user sets and item sets are both non-overlapped. It is conceivably the most challenging task of all.

The results are shown in Table 5 and Table 6. For the disjoint dimension, our algorithm can only identify the "similar" entities. (thus we do not report their matching accuracy.) Despite this, it

is very promising to see that the rating prediction can still be improved, most likely due to the fact that on the other overlapped dimension the correspondence can be identified. If both sides are disjoint, our model cannot yield any boost, as β becomes 0 after parameter tuning and our model degenerates into regular matrix factorization model.

In short, when original assumption is violated, our model may still lead to some improvement and it is at least no worse than regular matrix factorization model. Such discovery delivers an encouraging and important message for practical usage because the aforementioned scenarios are all very common in the real world.

Table 5: Matching Result on Yahoo! Music

	Rating List	**Latent Space**	**Refinement**
User Disjoint Split			
Accuracy(item)	0.017	0.148	0.247
MAP(item)	0.044	0.257	0.370
Item Disjoint Split			
Accuracy(user)	0.070	0.159	0.579
MAP(user)	0.121	0.250	0.663

Table 6: Rating Prediction (RMSE) on Yahoo! Music

	MF	**Proposed Approach**
User Disjoint Split	23.24	**23.15** †
Item Disjoint Split	23.88	**23.35** †
User-Item Disjoint Split	24.21	24.21

Those significantly better then the second best results are marked with †

5. CONCLUSION

We present a novel yet intuitive two-step algorithm for a very challenging and seldom tackled task of identifying user/item correspondences between two homogeneous rating matrices. After the correspondences are identified, we introduce a transfer learning approach to boost the rating prediction accuracy in the target domain. Our two-stage matching algorithm not only aims at matching users/items in the latent space, but also refines the matching based on the objective to predict the observed values in the target domain. The refinement not only boosts the quality of matching but also facilitates further transferring process to enhance rating prediction. We test our model on 8 different scenarios, each corresponds to a real-world scenario of transferring. The results are very promising as our model can identify the matching between user/item sets from different domains to some extent. More importantly, except one extremely difficult scenario where the users/items are both completely disjoint, our model significantly boosts the rating prediction performance.

6. ACKNOWLEDGEMENTS

This work was sponsored by AOARD grant number No. FA2386-13-1-4045, Ministry of Science and Technology, National Taiwan University and Intel Corporation under grants NSC102-2911-I-002-001 and NTU103R7501, and grant 102-2923-E-002-007-MY2, 102-2221-E-002-170, 101-2628-E-002-028-MY2.

References

[1] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer. The yahoo! music dataset and kdd-cup'11. *Journal of Machine Learning Research-Proceedings Track*, 2012.

[2] M. Gamon. Linguistic correlates of style: authorship classification with deep linguistic analysis features. In *Proc. COLING*, 2004.

[3] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and C. Zhu. Personalized recommendation via cross-domain triadic factorization. In *Proc. WWW*, 2013.

[4] M. Jamali and L. Lakshmanan. Heteromf: recommendation in heterogeneous information networks using context dependent factor models. In *Proc. WWW*, 2013.

[5] Y. Koren and R. Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145–186. Springer, 2011.

[6] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 2009.

[7] B. Li, Q. Yang, and X. Xue. Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In *Proc. IJCAI*, 2009.

[8] B. Li, Q. Yang, and X. Xue. Transfer learning for collaborative filtering via a rating-matrix generative model. In *Proc. ICML*, 2009.

[9] J. Liu, F. Zhang, X. Song, Y.-I. Song, C.-Y. Lin, and H.-W. Hon. What's in a name?: an unsupervised approach to link users across communities. In *Proc. WSDM*, 2013.

[10] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. Springer, 2011.

[11] H. Ma, H. Yang, M. R. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. In *Proc. CIKM*, 2008.

[12] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *Proc. NIPS*, 2008.

[13] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *Security and Privacy, IEEE Symposium on*, 2008.

[14] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Security and Privacy, IEEE Symposium on*, 2009.

[15] W. Pan, E. W. Xiang, N. N. Liu, and Q. Yang. Transfer learning in collaborative filtering for sparsity reduction. In *Proc. AAAI*, 2010.

[16] W. Pan and Q. Yang. Transfer learning in heterogeneous collaborative filtering domains. *Artificial Intelligence*, 2013.

[17] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *Proc. SIGKDD*, 2008.

[18] L. Xiong, X. Chen, T.-K. Huang, J. G. Schneider, and J. G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*, 2010.

[19] N. J. Yuan, F. Zhang, D. Lian, K. Zheng, S. Yu, and X. Xie. We know how you live: exploring the spectrum of urban lifestyles. In *Proc. COSN*, 2013.

[20] R. Zafarani and H. Liu. Connecting users across social media sites: a behavioral-modeling approach. In *Proc. KDD*, 2013.

[21] Y. Zhang, B. Cao, and D.-Y. Yeung. Multi-domain collaborative filtering. In *Proc. UAI*, 2010.