

國立臺灣大學電機資訊學院資訊工程學系

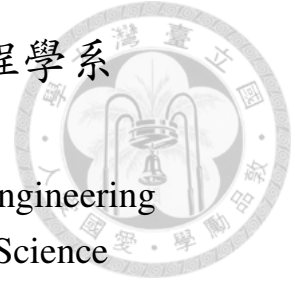
碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis



協同式過濾裡基於使用者匹配及物品匹配的轉移學習
Matching Users and Items for Transfer Learning in Collaborative
Filtering

李重毅

Chung-Yi Li

指導教授：林守德 博士

Advisor: Shou-De Lin, Ph.D.

中華民國 102 年 7 月

July, 2013

國立臺灣大學碩士學位論文
口試委員會審定書

協同式過濾裡基於使用者匹配及物品匹配的轉移學習

Matching Users and Items for Transfer Learning in
Collaborative Filtering

本論文係李重毅君（學號 R00922051）在國立臺灣大學資訊工程學系完成之碩士學位論文，於民國 102 年 7 月 30 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

林子德

（指導教授）

林智仁

王金強

詹育杰

林軒田

系主任

許永英



致謝

感謝守德老師的提攜和指點。也謝謝俊崧、冠緯、恩昶、鈞百，謝謝你們願意抽空和我討論，讓我獲得許多寶貴的想法和意見。最後，謝謝我的家人，好夥伴瑋詩、佳欣、鵬驊，俊宇、至中、育正、上恩、琬瑜、俊良、數學系的薛克民老師、NTUTM，還有所有一路上曾幫助我、支持我的人，謝謝你們。



中文摘要

本篇論文的研究目標是具有類似使用者和類似物品的同質性評分矩陣：即使不知道使用者的匹配關係和物品的匹配關係時，是否仍能在矩陣間進行轉移學習。更精確地說，我們假設有兩個評分矩陣，它們表達了同樣的喜好，而且兩個評分矩陣的使用者集合、物品集合都有很大一部份是重疊的。我們的目標便是找出這些使用者的匹配關係和物品的匹配關係，進而利用這樣的關係把一個矩陣的資訊傳遞到另一個矩陣上並改善其評分預測。

為了解出對應關係，我們會將稀疏的大型評分矩陣用低秩矩陣來近似，並用分解出來的因子來辨認使用者和物品。在解匹配問題的演算法中，一開始會把因子轉為奇異值分解的形式，並執行近鄰演算法。之後我們會指出奇異值分解的缺點，並用另一個目標函數來修正結果，以獲得更準確的匹配關係。最後，我們修改了協同式過濾常用的矩陣分解模型，使其能利用解出的匹配關係連結兩個矩陣，並做評分預測。我們的實驗顯示，在匹配問題中，我們能得到相當準的解。而即便匹配問題得到的解並不完美，我們仍能用其來改善評分預測模型。

關鍵詞：協同式過濾、轉移學習、矩陣分解



Abstract

This paper investigates the possibility of transferring information between homogeneous datasets of similar users and items but both user correspondence and item correspondence are unknown. More specifically, we assume there are two rating matrices that model the same kind of preferences, and there is a significant degree of overlap between the two user sets and between the two item sets. Our goal is to find out the user correspondence and item correspondence between the two rating matrices, and utilize the correspondence for exploiting the information of one matrix to improve the quality of rating prediction in the other matrix.

For finding out the correspondence, we factorize both rating matrices and exploit the latent factors to identify the users and items. The algorithm for solving the correspondence is initially based on singular value decomposition and nearest neighbor search, and then we point out the drawbacks of singular value decomposition and use another formulation to refine its result. Finally, we introduce a simple modification of regular matrix factorization model for transferring information across matrices with the obtained correspondence. In our experiment, we show that it is possible to solve the correspondence with decently high accuracy, and even with non-perfect correspondence obtained from our method, it is still possible to improve the quality of rating prediction.

Keywords: Collaborative Filtering, Transfer Learning, Matrix Factorization



Contents

致謝	ii
中文摘要	iii
Abstract	iv
1 Introduction	1
2 Related Work	4
2.1 Transfer Learning in Collaborative Filtering Given Correspondence	4
2.2 Transfer Learning in Collaborative Filtering When Correspondence is Unknown	6
3 Methodology	8
3.1 Solving the Correspondence Problem	8
3.2 Matching from Singular Value Decomposition	9
3.2.1 The Idea of Singular Value Decomposition	9
3.2.2 From Matrix Factorization to Singular Value Decomposition	9
3.2.3 Objective Function of SVD Matching	11
3.2.4 Remarks on SVD Matching	13
3.3 Refined Search Algorithm	13
3.4 Rating Prediction Given Noisy Matching	14
4 Experiment	16
4.1 Experiment Setup	16
4.2 Implementation Details	17
4.3 Experiment Result of User and Item Matching	18
4.4 Experiment Result for Rating Prediction	19
5 Conclusion and Future Work	21
Reference	22



List of Figures

1.1	A Partially Observed Low-Rank Matrix	1
1.2	User and Item Matching on Low-Rank Matrices	2
3.1	From MF to SVD	10
3.2	Matching as Nearest Neighbor Search	11
3.3	$\arctan(x)$	14
4.1	Illustration of Splits	17



List of Tables

4.1	Statistics of \mathbf{R}	16
4.2	Matching Result	18
4.3	Rating Prediction (RMSE)	19



Chapter 1

Introduction

Collaborative filtering is a popular technique used in recommendation systems [2]. The goal of a recommendation system is to model users' preferences on items. Unlike content-based filtering [6], which exploits information like user profiles and item attributes, collaborative filtering models predict users' future preferences on items based on current available preference records only.

	Item			
User	1	2	3	?
	?	2	3	4
	1	?	3	4
	1	2	?	4

Figure 1.1: A Partially Observed Low-Rank Matrix

More specifically, collaborative filtering algorithms can be viewed as a matrix completion process, as shown in Figure 1.1. The rows and columns represent users and items, and each entry in the matrix represents a rating given by the user to the item. The rating matrix is partially observed and we want to predict the missing entries in the matrix. We know that if the matrix is randomly generated, the missing elements can never be predicted, so we have to make assumptions about the rating matrix. The fundamental assumption of collaborative filtering is: if two users rated some items similarly in the past, they will rate other items similarly in the future; in other words, the two rows are dependent. Similar assumption is made on the item (column) side as well. For example, matrix factorization (MF), one of the most successful approach in collaborative filtering

[3, 1], assumes linear dependency among rows and among columns. Matrix factorization assumes the rating matrix is low-rank, and the goal is to recover a partially observed low-rank matrix. Back to the example in the Figure 1.1, if we assume the matrix is rank-1, then the missing values are easy to guess.

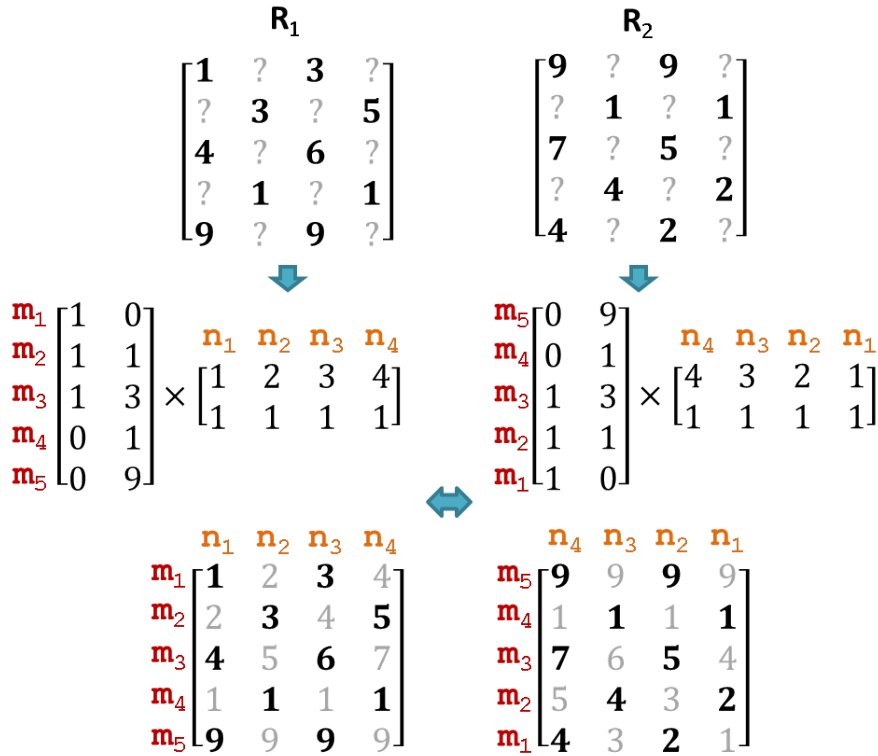


Figure 1.2: User and Item Matching on Low-Rank Matrices

There are many recommendation systems which are modelling the same kind of user preference. For example, both Netflix and IMDb contain American users' preferences toward movies, Similarly, Yahoo! music, Youtube and last.fm predict users' preferences toward music. We call these homogeneous rating datasets. The targeted items are similar among these datasets. The users also overlap since users tend to have multiple accounts across different websites. Homogeneous rating datasets are ideal sources for knowledge transfer. However, the users and items in these datasets are usually anonymous. It might be to protect user privacy, or simply because that the names are different across datasets. The anonymity of users and items becomes the main obstacle for transferring knowledge between homogeneous datasets. The major goal of this paper is to answer a question: when the identities of users and items are both unknown, is it still possible to transfer the

information between the two homogeneous systems?

In this paper, we assume we are given two homogeneous rating matrices, a target matrix \mathbf{R}_1 and another auxiliary data matrix \mathbf{R}_2 , where the two user sets and two item sets both overlap significantly, but we do not know how to link the users and the items between the matrices. We want to answer the following two problems:

1. Is it possible to find out the mapping of users and the mapping of items between \mathbf{R}_1 and \mathbf{R}_2 ?
2. Given the noisy mapping we obtained from our algorithm, is it sufficient to transfer information from \mathbf{R}_2 and help the rating prediction in \mathbf{R}_1 ?

The problems sound hard because the user correspondence and item correspondence are both unknown. If one side is matched, e.g. if the items are matched, then we can use approaches like k-nearest-neighbor to find out similar users and match them, as discussed in [9]. However, the problem is not impossible to solve. Figure 1.2 shows an illustrating example of \mathbf{R}_1 and \mathbf{R}_2 . If we assume the both matrices are low-rank and we factorize them as the way shown in the figure, the correspondence becomes obvious. For example, the first row of \mathbf{R}_1 should be mapped to the last row in \mathbf{R}_2 and they are labelled as user m_1 . This is the fundamental idea of our matching algorithm.

Our paper is organized as follows. We will first summarize the existing transfer learning models in collaborative filtering and discuss their difference from our model. Then we will present our two-step algorithm for user matching and item matching, and a factorization model that exploits the matching result for rating prediction. In the experiment, we will evaluate our approach on a noisy-free dataset and Yahoo! music dataset and discuss how well we can solve the under different scenarios, and how matching the result affects rating prediction.



Chapter 2

Related Work

Many models have been proposed for transfer learning or multi-task learning in collaborative filtering (sometimes called cross-domain collaborative filtering or multi-domain collaborative filtering). The common goal is to transfer information across different data matrices. Most of models assume that there is a one-to-one correspondence between users or between items across matrices. Such correspondence is known beforehand, but because of the heterogeneity, the data matrices must be treated separately. In contrast, our goal is to transfer information between homogeneous matrices when the correspondence is unknown. We will summarize heterogeneous models (given correspondence) in Section 2.1. Only two models assume homogeneous setting and the correspondence is unknown. We will discuss them in Section 2.2.

2.1 Transfer Learning in Collaborative Filtering Given Correspondence

In the following, we summarize models by listing an example of data matrices and the assumption of the model.

Collective Matrix Factorization [13]

- \mathbf{R}_1 : a rating matrix (users by movies), \mathbf{R}_2 : a label matrix (genres by movies).
- $\mathbf{R}_1 \approx \mathbf{P}_1 \mathbf{Q}^T$, $\mathbf{R}_2 \approx \mathbf{P}_2 \mathbf{Q}^T$.



Social Recommendation [7]

- \mathbf{R}_1 : a trust network (users by users), \mathbf{R}_2 : a rating matrix (users by movies).
- $\mathbf{R}_1 \approx \mathbf{P}\mathbf{Q}_1^T$, $\mathbf{R}_2 \approx \mathbf{P}\mathbf{Q}_2^T$.

Bayesian Probabilistic Tensor Factorization [14]

- \mathbf{R}_i : the rating matrix (users by movies) at time i .
- Combine all \mathbf{R}_i 's as a tensor \mathbf{R} and $\mathbf{R}_{imn} \approx \sum_k b_{ik}p_{mk}q_{nk}$. Or equivalently, $\mathbf{R}_i \approx \mathbf{P}\mathbf{B}_i\mathbf{Q}^T$, where \mathbf{B}_i 's are diagonal matrices.

Moreover, a special regularization term is added on \mathbf{B}_i 's to ensure the temporal smoothness of the model.

Multi-Domain Collaborative Filtering [15]

- \mathbf{R}_i : the rating matrix (users by movies) of movies type i .
- $\mathbf{R}_i \approx \mathbf{P}_i\mathbf{Q}_i^T$ and learn covariance matrices in order to fuse \mathbf{P}_i 's.

Coordinate System Transfer [11]

- \mathbf{R} : the target matrix, a numerical rating matrix (users by movies),
 \mathbf{R}_1 : an binary rating matrix (users by other items),
 \mathbf{R}_2 : an binary rating matrix (other users by movies).
- $\mathbf{R}_1 \approx \mathbf{U}_1\mathbf{D}_1\mathbf{V}_1^T$, $\mathbf{R}_2 \approx \mathbf{U}_2\mathbf{D}_2\mathbf{V}_2^T$, $\mathbf{R} \approx \mathbf{U}\mathbf{B}\mathbf{V}^T$.

For all \mathbf{U} 's and \mathbf{V} 's, $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, $\mathbf{V}^T\mathbf{V} = \mathbf{I}$, and \mathbf{D} 's are diagonal.

Use regularization to enforce $\mathbf{U} \approx \mathbf{U}_1$ and $\mathbf{V} \approx \mathbf{V}_2$.

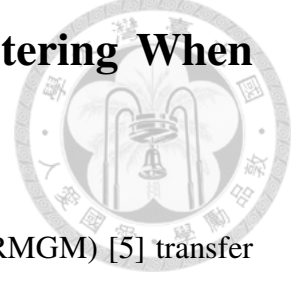
Collective SVD [10] (or journal version [12])

- \mathbf{R}_1 : a numerical rating matrix, \mathbf{R}_2 : a binary rating matrix (both users by movies).
- $\mathbf{R}_1 \approx \mathbf{U}\mathbf{B}_1\mathbf{V}^T$, $\mathbf{R}_2 \approx \mathbf{U}\mathbf{B}_2\mathbf{V}^T$, subject to $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, $\mathbf{V}^T\mathbf{V} = \mathbf{I}$.

In conclusion, all models transfer information across two or more heterogeneous data matrices via constraining the latent factors of the shared user side or item side to be similar.

In heterogeneous datasets, the rating patterns are different in each data matrix, e.g. the \mathbf{B} in tri-factorization models. On the other hand, in our setting the data matrices are homogeneous while both the user and item correspondence are unknown in our problem.

2.2 Transfer Learning in Collaborative Filtering When Correspondence is Unknown



Codebook transfer (CBT) [4] and rating-matrix generative model (RMGM) [5] transfer information between two rating matrices when user correspondence and item correspondence are both unknown. The basic assumption of two models are of the form:

$$\begin{aligned}\mathbf{R}_1 &\approx \mathbf{U}_1 \mathbf{B} \mathbf{V}_1^T, \\ \mathbf{R}_2 &\approx \mathbf{U}_2 \mathbf{B} \mathbf{V}_2^T,\end{aligned}$$

where \mathbf{B} is a shared K by K matrix and it represents the homogeneous rating pattern.

In CBT, it constrains \mathbf{U} and \mathbf{V} to be 0-1 matrix, and there can only be a 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: users and items are both divided into K groups; the rating prediction for a user and item is the value of the correspondence element in \mathbf{B} of the user group and item group. To connect the two domains, CBT assumes the cluster pattern matrix \mathbf{B} is shared between \mathbf{R}_1 and \mathbf{R}_2 . The main advantage of the model is that it does not need to solve user correspondence and item correspondence. Both correspondence matrices are naturally embedded in the clustering matrix \mathbf{U} and \mathbf{V} : for any \mathbf{U}_1 and \mathbf{U}_2 , we can always find a permutation \mathbf{G} representing user correspondence such that $\mathbf{G} \mathbf{U}_1 = \mathbf{U}_2$. However, the model is too restrictive. Take the matrix in Figure 1.2 in the introduction section for example. It is a rank-2 matrix, but it cannot be factorized into rank-2 $\mathbf{U} \mathbf{B} \mathbf{V}^T$ because of the hard clustering constraints. Besides, 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.

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

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

where $m^{(i)}$ and $n^{(i)}$ are a 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

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

The term $\sum_r r P(r|C_m, C_n)$ 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. However, restrictions and problems still exist in the model. First, the elements in each row of \mathbf{U} and \mathbf{V} must be nonnegative and sum to 1. Constraints limit the expressive power of the model and complicate the optimization task. Second, 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:

$$\mathbf{R} = \begin{bmatrix} 1 & 1 \\ 3 & 3 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0.5 & 0.5 \end{bmatrix} \times \begin{bmatrix} 1 & 1 \\ 3 & 3 \end{bmatrix} \times \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} = \mathbf{UBV}^T$$

Even though this \mathbf{UBV}^T perfectly predicts all ratings given by the three users, the data likelihood is much lower because the value 2 is far from the centers of all clusters (1 and 3). Thus, RMGM might try to find other \mathbf{U} , \mathbf{V} , \mathbf{B} to fit the data. In our experiment, we will show that RMGM performs poorly.

In conclusion, past models avoid solving correspondence at the cost of introducing extra restrictions. Our model directly tackles the correspondence problem and removes aforementioned restrictions.



Chapter 3

Methodology

3.1 Solving the Correspondence Problem

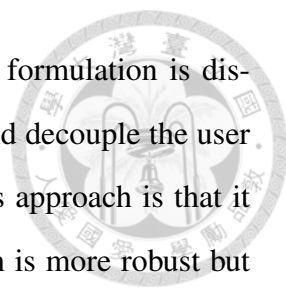
We have two partially observed rating matrices \mathbf{R}_1 and \mathbf{R}_2 and there are more rating records in \mathbf{R}_2 . The two matrices are homogeneous: we assume they can be merged into one homogeneous matrix when we know the correspondence. To find out the correspondence between \mathbf{R}_1 and \mathbf{R}_2 , we want to solve

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

where \mathbf{G}_{user} and \mathbf{G}_{item} are 0-1 matrices that represent user correspondence and item correspondence. However, remember that the rating matrices are partially observed. Figure 1.2 in the introduction section had shown a case where the corresponding entries in \mathbf{R}_1 and \mathbf{R}_2 never show up together. Therefore, we change the above formulation to the following two alternatives:

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

The $\hat{\mathbf{R}}$ stands for the low-rank approximation of \mathbf{R} and it has no missing values. Also, to simplify the optimization process, we do not require \mathbf{G} 's to be permutation matrices. We only impose constraints on each row such that one entry is 1 and others are 0.



In this section, we will discuss the two formulations. The first formulation is discussed in the Section 3.2. We apply singular value decomposition and decouple the user matching problem and item matching problem. The problem of this approach is that it is less accurate when the noise occurs, while the second formulation is more robust but harder to solve. Consequently, we start from solving the first formulation, which offers a good initialization and reduces the search space, then we will move on to the second formulation (Section 3.3) to improve the quality of matching.

3.2 Matching from Singular Value Decomposition

3.2.1 The Idea of Singular Value Decomposition

Given a completely observed matrix \mathbf{R} , the singular value decomposition (SVD) is $\mathbf{R} = \mathbf{U}\mathbf{D}\mathbf{V}^T$, where \mathbf{D} is diagonal and \mathbf{U} and \mathbf{V} are unitary. From the theory of linear algebra, we know that after sorted \mathbf{D} is unique given \mathbf{R} . Moreover, when all the singular values are distinct, \mathbf{U} and \mathbf{V} are unique except sign differences (e.g., multiply a column of \mathbf{U} and the corresponding column of \mathbf{V} by -1 at the same time). Thus, if we permute the rows and columns of \mathbf{R} , we are essentially switching the rows of \mathbf{U} and rows of \mathbf{V} . In other words, by examining the singular vectors in \mathbf{U} and \mathbf{V} , we can solve user and item correspondence.

3.2.2 From Matrix Factorization to Singular Value Decomposition

Conventional SVD solver cannot be applied on an incomplete matrix. Unlike some papers that solve a constrained optimization problem to get the solution, we provide a simple and efficient process to transform the result of regular matrix factorization (MF) to the form of SVD. (Note that regular matrix factorization is called probabilistic matrix factorization [8] in some literature.)

\mathbf{R} is a rating matrix and \mathbf{W} is an indicator matrix where $\mathbf{W}_{ij} = 1$ means user i rated item j , $\mathbf{W}_{ij} = 0$ otherwise. The number of users is M and the number of items is N . K is the parameter that controls the rank. If we omit the regularization terms, the objective



function of matrix factorization is to find $\mathbf{P}_{M \times K}$ and $\mathbf{Q}_{N \times K}$ such that

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

\odot is the Hadamard (entrywise) product and the norm is Frobenius norm. The SVD model we want is to find $\mathbf{U}_{M \times K}$, $\mathbf{D}_{K \times K}$ and $\mathbf{V}_{N \times K}$ such that

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

We can transform \mathbf{P} , \mathbf{Q} to \mathbf{U} , \mathbf{D} , \mathbf{V} efficiently by applying conventional SVD on \mathbf{P} and \mathbf{Q} . The process is illustrated in Figure 3.1. 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,

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

Thus, $\mathbf{U} = \mathbf{U}_P \mathbf{U}_X$, $\mathbf{V} = \mathbf{U}_Q \mathbf{V}_X$ and $\mathbf{D} = \mathbf{D}_X$.¹

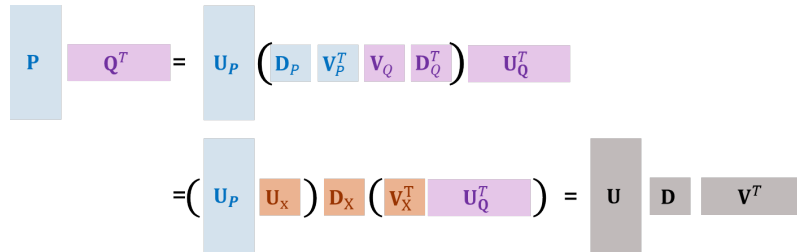


Figure 3.1: From MF to SVD

Finally, 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 a large matrix. To make a fair comparison between matrices of different sizes, when we are doing the matching, we redefine \mathbf{U} , \mathbf{D} and \mathbf{V} as

$$\mathbf{U} = \mathbf{U} \times M^{0.5}, \mathbf{D} = \mathbf{D} \times M^{-0.5} \times N^{-0.5}, \mathbf{V} = \mathbf{V} \times N^{0.5}.$$

¹the matrix is highly asymmetric so thin SVD (economical SVD) should be used in the implementation for efficiency.



3.2.3 Objective Function of SVD Matching

Here we want to solve

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

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

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

Since SVD is unique except the sign difference, we can decouple the above equation into two matching problems:

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

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

\mathbf{G} means the user correspondence matrix or item correspondence matrix, and \mathbf{S} is the sign matrix, a small diagonal matrix (K by K) and each diagonal element is either 1 or -1 . Theoretically, \mathbf{S}_{user} is equal to \mathbf{S}_{item} , but we do not enforce this constraint. Since the formulations of user matching and item matching are identical, from now on we will discuss user side only.

When \mathbf{S} is given, the matching problem becomes a simple nearest neighbor search problem: for each row in $\mathbf{U}_1 \mathbf{D}_1^{0.5}$, the goal is to find the most similar row in $\mathbf{U}_2 \mathbf{D}_2^{0.5}$. We illustrate this in Figure 3.2, where $\mathbf{Z} = \mathbf{U} \mathbf{D}^{0.5}$. It takes $\Theta(M_1 M_2 K)$ time to solve \mathbf{G} , where M_1 and M_2 are the number of users in \mathbf{R}_1 and \mathbf{R}_2 .

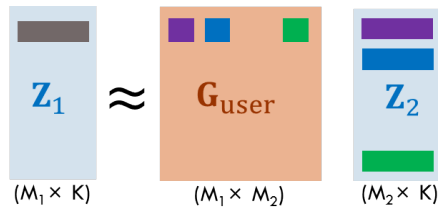


Figure 3.2: Matching as Nearest Neighbor Search

For solving the sign matrix, there are 2^K possible solutions so it is impossible to

try all possibilities. We use a greedy approach to get an approximate solution. Assume $d_1 \geq d_2 \geq \dots \geq d_K$, where d_i 's are singular values. In the k -th iteration, we try both $s_k = 1$ and $s_k = -1$, and solve \mathbf{G} by looking at dimensions from 1 to k . We will choose the sign of s_k that results in a smaller objective value. After determining s_k , we move on to the next iteration and solve s_{k+1} .

On the other hand, since the small singular values and singular vectors are more noisy, we want to use only the top singular values and vectors for matching. This can be combined with the sign solving process. We will output a $\mathbf{G}_{(k)}$ for each latent dimension k . In the end, we will select the latent dimension k and the corresponding \mathbf{G} that leads to better rating prediction. We summarize the complete procedure in algorithm 1.

Algorithm 1 SVD 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 MATCHING($\mathbf{Z}_1, \mathbf{Z}_2$)

initialize all elements in the all-pair distance table T to 0 \triangleright caching for speed-up

for $k = 1$ to K **do**

let $s_k = +1$

let $(\alpha_+, \mathbf{G}_+) = \min_{\mathbf{G}} \|\mathbf{Z}_1(:, 1:k) - \mathbf{G}\mathbf{Z}_2(:, 1:k)\mathbf{S}\|^2$ using T

let $s_k = -1$

let $(\alpha_-, \mathbf{G}_-) = \min_{\mathbf{G}} \|\mathbf{Z}_1(:, 1:k) - \mathbf{G}\mathbf{Z}_2(:, 1:k)\mathbf{S}\|^2$ using 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 $\mathbf{G}_{(1)}, \dots, \mathbf{G}_{(K)}$

end function

$\mathbf{G}_{\text{user}(1:K)} = \text{MATCHING}(\mathbf{U}_1\mathbf{D}_1^{0.5}, \mathbf{U}_2\mathbf{D}_2^{0.5})$

$\mathbf{G}_{\text{item}(1:K)} = \text{MATCHING}(\mathbf{V}_1\mathbf{D}_1^{0.5}, \mathbf{V}_2\mathbf{D}_2^{0.5})$

for $k = 1$ to K **do**

$\text{error}_k = \|\mathbf{W}_1 \odot (\mathbf{R}_1 - \mathbf{G}_{\text{user}(k)}\mathbf{U}_2\mathbf{D}_2\mathbf{V}_2\mathbf{G}_{\text{item}(k)}^T)\|_{\text{Fro}}^2$ \triangleright dimension selection

end for

$(\text{error}_{\min}, k_{\min}) = \min_k \text{error}_k$

$\mathbf{G}_{\text{user}}^* = \mathbf{G}_{\text{user}(k_{\min})}$

$\mathbf{G}_{\text{item}}^* = \mathbf{G}_{\text{item}(k_{\min})}$

In algorithm 1, we maintain a distance table T that records all pair distance between \mathbf{Z}_1 and \mathbf{Z}_2 for speed-up. Thus, the complete time complexity is still $\Theta(M_1M_2K)$ for

user matching and $\Theta(N_1 N_2 K)$ for item matching. The dimension selection part takes $\Theta(K \times R \times K) = \Theta(K^2 R)$, where R is the number of ratings in \mathbf{R}_1 .



3.2.4 Remarks on SVD Matching

The main advantage of SVD matching algorithm is that it decouples user matching and item matching. In the ideal case, the result of SVD matching is perfect (in Section 4.3, our experiment will show this). However, there are some disadvantages of SVD matching when noise occurs. First, SVD matching is based on matrix factorization, but the result of matrix factorization is only an approximation of the original rating matrix. Second, when some singular values are too close and the data is too noisy, the corresponding singular vectors are mixed together (in degenerate cases, SVD is not unique). Therefore, we would like to use another formulation and improve the result of SVD matching.

3.3 Refined Search Algorithm

Assume the low-rank approximation of \mathbf{R}_2 is $\hat{\mathbf{R}}_2 = \mathbf{P}_2 \mathbf{Q}_2^T$, we consider the following objective function to improve the result of SVD matching:

$$\min_{\mathbf{G}_{\text{user}}, \mathbf{G}_{\text{item}}} \|\mathbf{R}_1 - \mathbf{G}_{\text{user}} \mathbf{P}_2 \mathbf{Q}_2^T \mathbf{G}_{\text{item}}^T\|$$

From the SVD matching algorithm, we got a good initialization of \mathbf{G}_{user} and \mathbf{G}_{item} . Also, since SVD matching is similar to nearest neighbor search, it can output a candidate neighbor list for each user (item). In this algorithm, we will only look at the candidates given by SVD matching and reevaluate them based on the above formulation.

Assume the number of candidates we get from SVD matching is C , then the size of \mathbf{G}_{user} and \mathbf{G}_{item} become M_1 by C and N_1 by C , where M_1 and N_1 are the number of users and the number of items in \mathbf{R}_1 . We solve \mathbf{G} in a similar manner to two-block coordinate descent, except here the variables in \mathbf{G} are discrete. When fixing \mathbf{G}_{item} , each row in \mathbf{G}_{user} becomes independent: finding the best \mathbf{G}_{user} becomes a simple search problem. For each user in \mathbf{R}_1 , we are finding the candidate (which will select the corresponding row

in \mathbf{P}_2) that minimizes the rating prediction error in the row of \mathbf{R}_1 . It takes $\Theta(RCK)$ time to solve \mathbf{G}_{user} , where R is the number of ratings in \mathbf{R}_1 . Also, when there are several candidates that yield the same objective value, we will choose the one with the least index. Then, if \mathbf{G}_{user} is changed, the objective value will decrease. Things are the same when fixing \mathbf{G}_{user} . In short, our algorithm solves \mathbf{G}_{user} and \mathbf{G}_{item} alternatively until both \mathbf{G} 's are not changing (which is like a local minimum). Since there is only a finite number of possibilities, the algorithm will converge.

3.4 Rating Prediction Given Noisy Matching

In matrix factorization, our goal is to find \mathbf{P} and \mathbf{Q} to minimize

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

The norms here are all Frobenius norm and we omit the subscript. When the user and item correspondence is given, let g_m be the corresponding user in \mathbf{R}_2 for user m in \mathbf{R}_1 and g_n be the corresponding item in \mathbf{R}_2 for item n in \mathbf{R}_1 . We combine the matrix factorization of \mathbf{R}_1 and \mathbf{R}_2 by adding a regularization term:

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

The extra regularization term implies that the latent factors of matched users (items)

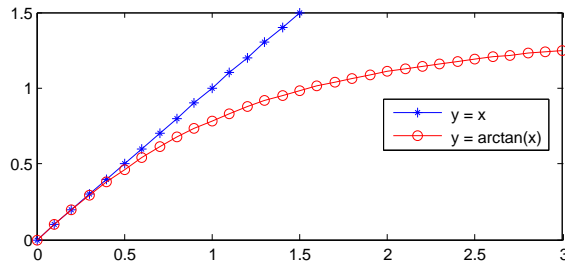
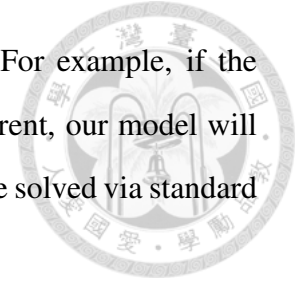


Figure 3.3: $\arctan(x)$

should be similar. The arctan reduces the influence of outliers. For example, if the matching is wrong and the latent factors of two users are very different, our model will not be distorted much. The objective is still differentiable and it can be solved via standard approaches like gradient descent.





Chapter 4

Experiment

4.1 Experiment Setup

Table 4.1: Statistics of \mathbf{R}

Dataset	Number of Users	Number of Items	Rating Scale	Sparsity
Pure Low Rank	20000	10000	$[-1,1]$	5%
Yahoo! Music	20000	10000	integers in 0-100	5.4%

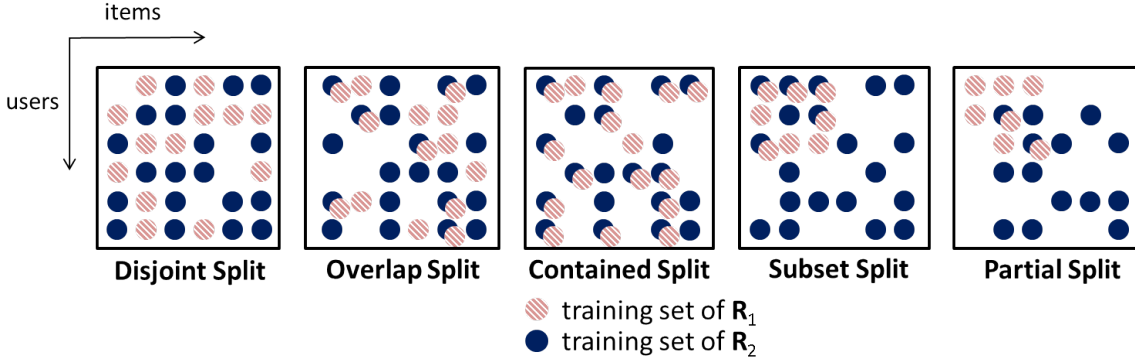
We conduct experiments on Yahoo! music dataset from track 1 of KDD cup 2011 [1]. We take out a dense subset as \mathbf{R} and split it into \mathbf{R}_1 and \mathbf{R}_2 . The user and item ids of \mathbf{R}_2 are randomly permuted, and the permutation result will become the ground truth of correspondence. Our goal is to solve the correspondence as well as use \mathbf{R}_2 to help rating prediction in \mathbf{R}_1 . The correspondence problem is an unsupervised learning problem so the mapping of users and the mapping of items are only used in the evaluation process. For verifying the soundness of our approach, we also try our algorithm on a synthetic dataset, which is a noise-free low-rank matrix. ¹

To be more specific, we split \mathbf{R} into training set, validation set, testing set of \mathbf{R}_1 and training set, validation set of \mathbf{R}_2 . Each of the validation sets or testing set never overlaps with any other set so as to ensure the sanity of our experiment. The training sets of \mathbf{R}_1 and \mathbf{R}_2 may overlap depending on the following scenarios.

¹It is a rank-50 matrix, generated by matlab command “`randn(20000,50)*diag(1.1.^[1:50])*randn(50,10000)`”. We subsample 5% of it as \mathbf{R} and linearly scale the minimum and maximum values to -1 and 1 .

We consider five different scenarios to see how the overlap ratio affects the matching accuracy. The scenarios are illustrated in the Figure 4.1. We design *disjoint split*, *overlap split* and *contained split* to reflect different overlap ratios of rating records. In the three splits, the number of ratings in training set of \mathbf{R}_1 , validation set of \mathbf{R}_1 , testing set of \mathbf{R}_1 , training set of \mathbf{R}_2 and validation set of \mathbf{R}_2 are 40%, 2.5%, 5%, 50% and 2.5% of \mathbf{R} , respectively. The user sets and item sets of \mathbf{R}_1 and \mathbf{R}_2 are the same for the three splits, so we also try *subset split* and *partial split*: in the *subset split*, the user set and the item set of \mathbf{R}_1 are contained in those of \mathbf{R}_2 (40% for \mathbf{R}_1 and 100% for \mathbf{R}_2), while in the *partial split*, the user set and the item set of \mathbf{R}_1 and those of \mathbf{R}_2 only partially overlap (40% for \mathbf{R}_1 , 90% for \mathbf{R}_2 , 30% are in both). Besides, we adjust the number of ratings in \mathbf{R}_1 and \mathbf{R}_2 in the two splits in order to make the number of ratings per user similar to the previous three splits.

Figure 4.1: Illustration of Splits



4.2 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 0 for purely low-rank dataset and 5 for Yahoo! music dataset, and β ranges from 25 to 400. Besides, we have 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 will stop when validation error starts increasing.

We implement two versions of rating-matrix generative model (RMGM) [5]. They can be solved by standard expectation-maximization algorithm.² The original RMGM uses categorical distribution for $P(r|C_m, C_n)$, and we implement the other RMGM that uses Gaussian distribution. For both models, we set the latent dimension K to 50 and we have included the early stopping procedure. For Gaussian RMGM, we found that the variance of Gaussian is better set to a constant and automatically selected by observing the error rate of the validation set. The selected values range from 0.5 to 0.7. We have also included data scaling for Gaussian RMGM.

4.3 Experiment Result of User and Item Matching

Table 4.2: Matching Result

(a) Purely Low-Rank Dataset			(b) Yahoo! Music Dataset		
	SVD Match	Refined Search		SVD Match	Refined Search
Disjoint Split			Disjoint Split		
Accuracy(user)	1.000	1.000	Accuracy(user)	0.310	0.633
MAP(user)	1.000	1.000	MAP(user)	0.419	0.717
Accuracy(item)	1.000	1.000	Accuracy(item)	0.204	0.325
MAP(item)	1.000	1.000	MAP(item)	0.325	0.463
Overlap Split			Overlap Split		
Accuracy(user)	1.000	1.000	Accuracy(user)	0.547	0.960
MAP(user)	1.000	1.000	MAP(user)	0.652	0.973
Accuracy(item)	1.000	1.000	Accuracy(item)	0.442	0.786
MAP(item)	1.000	1.000	MAP(item)	0.578	0.859
Contained Split			Contained Split		
Accuracy(user)	1.000	1.000	Accuracy(user)	0.851	0.997
MAP(user)	1.000	1.000	MAP(user)	0.905	0.998
Accuracy(item)	1.000	1.000	Accuracy(item)	0.815	0.975
MAP(item)	1.000	1.000	MAP(item)	0.886	0.986
Subset Split			Subset Split		
Accuracy(user)	1.000	1.000	Accuracy(user)	0.392	0.918
MAP(user)	1.000	1.000	MAP(user)	0.510	0.941
Accuracy(item)	1.000	1.000	Accuracy(item)	0.297	0.686
MAP(item)	1.000	1.000	MAP(item)	0.433	0.780
Partial Split			Partial Split		
Accuracy(user)	1.000	1.000	Accuracy(user)	0.350	0.871
MAP(user)	1.000	1.000	MAP(user)	0.470	0.906
Accuracy(item)	1.000	1.000	Accuracy(item)	0.272	0.573
MAP(item)	1.000	1.000	MAP(item)	0.402	0.684

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

In the matching process, our algorithm can output either the most likely candidate or a rank list of candidates for each user (item) in \mathbf{R}_1 . Therefore, we use accuracy as well as mean average precision (MAP) as evaluation criteria. On the other hand, 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, we will not consider these users and items.

The matching result is shown in Table 4.2(a) and 4.2(b). The result on purely low-rank dataset proves the soundness of our approach. When there is no noise in the rating matrix, matrix factorization recovers the whole matrix almost perfectly. In this case, SVD matching can successfully distinguish all users and items by comparing the singular vectors, and our algorithm is applicable on all kinds of splits. On Yahoo! music dataset, we can see that in the first three splits the higher the overlap ratio is, the better the result will be. For the subset split and partial split, the overlap condition is similar to the overlap split, so they are a bit harder than the overlap split but still better than the disjoint split. In all cases, refined search leads to great improvement over SVD matching. It is reasonable because there are some drawbacks of SVD matching, which we have discussed in Section 3.2.4.

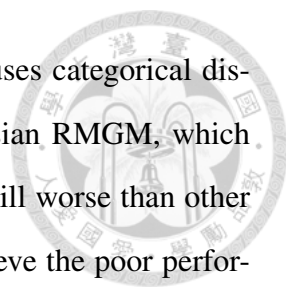
4.4 Experiment Result for Rating Prediction

Table 4.3: Rating Prediction (RMSE)

	RMGM	Gaussian RMGM	MF	Proposed Approach
Disjoint Split	27.5	26.6	24.2	23.3
Overlap Split	27.5	26.5	24.3	23.5
Contained Split	27.6	26.5	24.3	23.9
Subset Split	27.6	26.6	24.2	23.4
Partial Split	27.6	26.5	24.2	23.8

For rating prediction, we use root mean square error (RMSE) as the evaluation criterion. Since matrix factorization recovers the whole matrix almost perfectly on purely low-rank dataset, we focus on Yahoo! music dataset here. The experiment result is shown in the Table 4.3.

We can see that rating-matrix generative model (RMGM) performs poorly in all cases.



Some people might argue that RMGM performs poorly because it uses categorical distribution for $P(r|C_m, C_n)$. Therefore, we have implemented Gaussian RMGM, which changes the term to Gaussian distribution. The result is better, but still worse than other models. As we discussed in the relation work (Section 2.2), we believe the poor performance of RMGM is because of the constraints and the design of the objective function.

In all cases, our proposed approach leads to great improvement over matrix factorization (MF), which is the best single domain model. Interestingly, even though the matching accuracy is lower when the overlap ratio is lower, the improvement is greater. This is because when there is less overlap, the two training sets (\mathbf{R}_1 and \mathbf{R}_2) together contain more rating records, and more information available leads to better prediction models.



Chapter 5

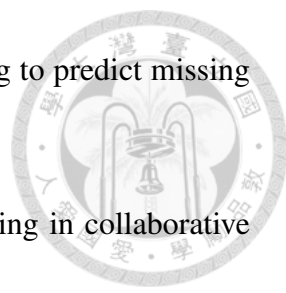
Conclusion and Future Work

We present a novel two-step algorithm for identifying user correspondence and item correspondence between two homogeneous rating matrices. We then introduce a simple modification of regular matrix factorization model for transferring the information given the obtained correspondence. The correspondence is identified by utilizing the latent factors. The initial algorithm, SVD matching, works perfectly on noise-free data, but it has problems dealing with data noise. In this case, refined search can lead to a much better solution. Even though the correspondence problem sounds difficult, the experiment result shows that it is possible to identify the correspondence with decently high accuracy. We note that the matching result suggests a privacy concern on publishing rating datasets. More studies can be done in this direction for further investigation. On the other hand, our transfer learning model is robust to wrong matches. We show that even when our matching result is non-perfect, our algorithm still leads to great improvement in the quality of rating prediction.



Reference

- [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] Y. Koren and R. Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145–186. Springer, 2011.
- [3] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 2009.
- [4] B. Li, Q. Yang, and X. Xue. Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In *Proc. IJCAI*, 2009.
- [5] B. Li, Q. Yang, and X. Xue. Transfer learning for collaborative filtering via a rating-matrix generative model. In *Proc. ICML*, 2009.
- [6] 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.
- [7] H. Ma, H. Yang, M. R. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. In *Proc. CIKM*, 2008.
- [8] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *Proc. NIPS*, 2008.
- [9] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *Security and Privacy, IEEE Symposium on*, 2008.

- 
- [10] W. Pan, N. N. Liu, E. W. Xiang, and Q. Yang. Transfer learning to predict missing ratings via heterogeneous user feedbacks. In *Proc. IJCAI*, 2011.
- [11] W. Pan, E. W. Xiang, N. N. Liu, and Q. Yang. Transfer learning in collaborative filtering for sparsity reduction. In *Proc. AAAI*, 2010.
- [12] W. Pan and Q. Yang. Transfer learning in heterogeneous collaborative filtering domains. *Artificial Intelligence*, 2013.
- [13] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *Proc. SIGKDD*, 2008.
- [14] 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.
- [15] Y. Zhang, B. Cao, and D.-Y. Yeung. Multi-domain collaborative filtering. In *Proc. UAI*, 2010.