TRANSFER LEARNING IN COLLABORATIVE FILTERING

by

WEIKE PAN

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
in Computer Science and Engineering

June 2012, Hong Kong

Copyright © by Weike Pan 2012
Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

WEIKE PAN
TRANSFER LEARNING IN COLLABORATIVE FILTERING

by

WEIKE PAN

This is to certify that I have examined the above Ph.D. thesis and have found that it is complete and satisfactory in all respects, and that any and all revisions required by the thesis examination committee have been made.

PROF. QIANG YANG, THESIS SUPERVISOR

PROF. MOUNIR HAMDI, HEAD OF DEPARTMENT

Department of Computer Science and Engineering

3 June 2012
ACKNOWLEDGMENTS

First of all, I would like to express my sincere thanks to my supervisor Prof. Qiang Yang. I benefit enormously from your support in my difficult times. I learn a lot from your vision and advice in doing my research. I am deeply impressed and changed by your passion in research and self-giving service in various communities. I am also very thankful to my co-supervisor Prof. Vincent Y. Shen for your great help these years.

I would like to thank Prof. Sunghun Kim, Prof. Qiang Yang, Prof. Dit-Yan Yeung and Prof. Nevin Lianwen Zhang for your serving as the examination committee for my thesis proposal defense; and Prof. Lei Chen, Prof. Wilfred Siu-Hung Ng, Prof. Weichuan Yu, Prof. Haifeng Wang and Prof. Qiang Yang for your willingness and effort to serve as the examination committee for my thesis defense.

I would like to thank my classmates and friends at HKUST, Bin Cao, Weizhu Chen, Michelle Dan Hong, Chonghai Hu, Derek Hao Hu, Wu-Jun Li, Yu-Feng Li, Nathan Nan Liu, Zhongqi Lu, Kaixiang Mo, Sinno Jialin Pan, Si Shen, Ben Tan, Ivor Wai-Hung Tsang, Bin Wu, Evan Wei Xiang, Qian Xu, Can Yang, Mingxuan Yuan, Kai Zhang, Yu Zhang, Skye Lili Zhao, Yi Zhen, Vincent Wenchen Zheng, Erheng Zhong, Wenliang Zhong, Yin Zhu, and so on. Thanks for your accompany and sharing.

I would like to thank my mentors and colleagues at Baidu and Tencent during my internships. Thanks for your help and sharing.

I am also very thankful to my pastors, spiritual advisors, brothers and sisters from fellowships and churches in Hong Kong, Hangzhou, Cixi, and so on. Thanks for your prayers.

Last but not least, I would like to say thanks to my wife, my parents and my parents in law. Without your love, for me, nothing is possible.
# TABLE OF CONTENTS

Title Page i

Authorization Page ii

Signature Page iii

Acknowledgments iv

Table of Contents v

List of Figures ix

List of Tables xi

Abstract xiii

Chapter 1 Introduction 1
  1.1 Background 2
    1.1.1 Categorization of Auxiliary Data 3
    1.1.2 Problem Definition 4
    1.1.3 Basic Math Formulas 5
  1.2 Specific Problems and Challenges 7
  1.3 Main Contributions 8
  1.4 Notations 10
  1.5 Thesis Outline 10

Chapter 2 Related Work 13
  2.1 Transfer Learning Methods 13
    2.1.1 Model-based Transfer 15
    2.1.2 Instance-based Transfer 17
    2.1.3 Feature-based Transfer 18
    2.1.4 Summary 23
  2.2 Collaborative Filtering Techniques 24
    2.2.1 Memory-based Methods 24
    2.2.2 Model-based Methods 26
Chapter 3  Transfer Learning in Collaborative Filtering with Two-Sided Implicit Feedbacks 34

3.1 Introduction 34

3.2 Collaborative Filtering with Implicit Feedbacks 36

3.2.1 Problem Definition 36

3.2.2 Challenges 37

3.2.3 Overview of Our Solution 38

3.3 Coordinate System Transfer 39

3.3.1 Step 1: Coordinate System Construction 39

3.3.2 Step 2: Coordinate System Adaptation 40

3.3.3 Learning the CST 43

3.3.4 Extensions of CST 48

3.4 Experimental Results 49

3.4.1 Data Sets and Evaluation Metrics 50

3.4.2 Baselines and Parameter Settings 51

3.4.3 Summary of the Experimental Results 53

3.5 Discussions 57

3.5.1 Transfer Learning in Collaborative Filtering 57

3.5.2 Manifold Learning in Matrix Factorization 59

3.5.3 Tensor Factorization 61

3.6 Summary 62

Chapter 4  Transfer Learning in Collaborative Filtering with Frontal-Side Binary Ratings 63

4.1 Introduction 63

4.2 Collaborative Filtering with Binary Ratings 68

4.2.1 Problem Definition 68

4.2.2 Challenges 69
# LIST OF FIGURES

1.1 Auxiliary data in a real video streaming system of Tencent Video (http://v.qq.com/).  
1.2 The organization of thesis.  
2.1 A summary of works on collaborative filtering using auxiliary data.  
3.1 Illustration of transfer learning from two-sided implicit feedbacks via \textit{coordinate system transfer} (CST).  
3.2 The algorithm of \textit{coordinate system transfer} (CST).  
3.3 Comparison of CST-biased, CST-manifold and OptSpace at different sparsity levels with different latent dimension numbers (auxiliary sparsity: 10\%, 9.5\%).  
3.4 Prediction performance of CST-biased and CST-manifold at different sparsity levels of auxiliary and target data ($d = 10$).  
3.5 Prediction performance of CST with and without the regularization term (auxiliary sparsity: 10\%, 9.5\%, target sparsity: 0.2\%).  
4.1 Illustration of transfer learning from frontal-side binary ratings via \textit{transfer by collective factorization} (TCF).  
4.2 The algorithm of \textit{transfer by collective factorization} (TCF).  
4.3 Logistic link function $\sigma(x) = \frac{1}{1 + e^{-(x - 0.5)}}$.  
4.4 Prediction performance of TCF (CMTF, CSVD) on Netflix at different sparsity levels with different auxiliary data.  
5.1 Illustration of transfer learning in collaborative filtering from auxiliary uncertain ratings (left: target 5-star numerical ratings; right: auxiliary uncertain ratings represented as ranges or rating distributions). Note that there is a one-one mapping between the users and items from two data.  
5.2 Illustration of the expected rating estimated using Eq.(5.4) with $a_{ui} = 4$ and $b_{ui} = 5$.  
5.3 The algorithm of \textit{transfer by integrative factorization} (TIF).  
5.4 Prediction performance of RSVD, TIF(avg.) and TIF with different iteration numbers (the tradeoff parameter $\lambda$ is fixed as 1).  
5.5 Prediction performance of RSVD, TIF(avg.) and TIF on different user groups (using the first fold of the MovieLens10M data). The tradeoff parameter $\lambda$ is fixed as 1, and the number of iterations is fixed as 50.  
6.1 VIP recommendation in the microblogging social network of Tencent Weibo (http://t.qq.com/).  
6.2 Matrix illustration of the problem setting of transfer learning across heterogeneous social networks of instant messenger and microblog.
6.3 Illustration of the recommendation procedure using instant messenger X and microblog R. From the friendship relations in instant messenger, we can find five friends of user A: B, C, D, E, F, and according to the following relations in microblog (B → X, C → X, D → X and E → Y, F → Y), we can recommend VIP X and Y to user A.

6.4 Prediction performance of “Memory”, SORT-friend (Friend), SORT-followee (Followee) and SORT-friend-followee (SORT) on the whole user set, warm-start user set and cold-start user set.
LIST OF TABLES

1.1 A brief summary of auxiliary data in collaborative filtering. 4
1.2 Notations of variables for different data. 11
3.1 Matrix illustration of coordinate system transfer (CST). 37
3.2 Description of a subset of Netflix data \((n = 5,000, m = 5,000)\) and a subset of MovieLens data \((n = 5,000, m = 5,000)\) used in the experiments. 51
3.3 Prediction performance of CST and other methods on the subset of Netflix data. 53
3.4 Summary of CST and other transfer learning methods in collaborative filtering. 59
3.5 A brief comparison of high-order CST with other tensor factorization methods. \(D_\perp\) denotes orthonormal constraints on the factorized matrices. 61
4.1 Matrix illustration of some related work on transfer learning in collaborative filtering. 65
4.2 Matrix illustration of transfer by collective factorization. 68
4.3 Description of a subset of Moviepilot data \((n = m = 2,000)\) and a subset of Netflix data \((n = m = 5,000)\) used in the experiments. 81
4.4 Prediction performance of TCF and other methods on the subset of Moviepilot data. 83
4.5 Prediction performance of TCF and other methods on the subset of Netflix data. 83
4.6 Summary of TCF and other transfer learning methods in collaborative filtering. 89
5.1 Description of MovieLens10M data \((n = 71,567, m = 10,681)\) and Netflix data \((n = 480,189, m = 17,770)\). Sparsity refers to the percentage of observed ratings in the user-item preference matrix, e.g. \(q_{nm}\) and \(\tilde{q}_{nm}\) are sparsities for target data and auxiliary data, respectively. 100
5.2 Prediction performance of RSVD, TIF(avg.) and TIF on MovieLens10M data (ML) and Netflix data (NF). The tradeoff parameter \(\lambda\) is fixed as 1, and the number of iterations is fixed as 50. 103
5.3 Prediction performance of TIF on MovieLens10M data (ML) and Netflix data (NF) with different value of \(\lambda\). The number of iterations is fixed as 50. 103
5.4 Overview of TIF in a big picture of traditional transfer learning and transfer learning in collaborative filtering. 104
5.5 Summary of TIF and other transfer learning methods in collaborative filtering. 104

6.1 Summary of SORT and other transfer learning methods in collaborative filtering. 109

6.2 Prediction performance of matrix factorization and “Memory” in our preliminary study (training: accumulated data till May 31, 2011, test: new following relations between June 1 and June 7, 2011). 124

6.3 Prediction performance on the whole user set. 126

6.4 Prediction performance on the warm-start user set. 127

6.5 Prediction performance on the cold-start user set. 128

7.1 Overview of our work in a big picture of transfer learning in collaborative filtering. 131

7.2 Overview of our work in a big picture of traditional transfer learning and transfer learning in collaborative filtering. 131
TRANSFER LEARNING IN
COLLABORATIVE FILTERING

by

WEIKE PAN

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology

ABSTRACT

Transfer learning and collaborative filtering have been studied in each community separately since early 1990s and were married in late 2000s. Transfer learning is proposed to extract and transfer knowledge from auxiliary data to improve the target learning task and has achieved great success in text mining, mobile computing, bioinformatics, etc. Collaborative filtering is a major intelligent component in various recommender systems, like movie recommendation in Netflix, news recommendation in Google News, people recommendation in Tencent Weibo (microblog), advertisement recommendation in Facebook, etc. However, in many collaborative filtering problems, we may not have enough data of users’ preferences on items, which is known as the data sparsity problem. Transfer learning in collaborative filtering (TLCF) is studied to address the data sparsity problem in the user-item preference data in recommender systems.

In this thesis, we develop this new multidisciplinary area mainly from two aspects. First, we propose a general learning framework, study four new and specific problem settings for movie recommendation and people recommendation, and design four novel TLCF solutions correspondingly. We transfer knowledge from different types of auxiliary data based on a general regularization framework, and design batch algorithms, stochastic algorithms and distributed algorithms to solve the optimization problems.
Second, we survey and categorize traditional transfer learning works into model-based transfer, instance-based transfer and feature-based transfer, and build a relationship between traditional transfer learning algorithms and TLCF solutions from a unified view of model-based transfer, instance-based transfer, and feature-based transfer.
CHAPTER 1

INTRODUCTION

Collaborative filtering serves as a critical intelligent component in various industry recommender systems. For example, in online e-commerce website of Amazon\(^1\), online movie rental company of Netflix\(^2\), online microblog of Tencent Weibo\(^3\), and online social network service of Facebook\(^4\), collaborative filtering techniques are used for book, movie, people and advertisement recommendations, respectively. Collaborative filtering brings in more revenues for companies and more user activities for online community platforms.

However, collaborative filtering suffers from the data sparsity problem, that is, the users’ preference data on items are usually too few to understand the users’ true preferences, which makes the personalized recommendation task difficult.

In our observation, though the target user-item preference matrix of numerical ratings are sparse, there are some related auxiliary data we may explore to reduce the target data sparsity problem. For example, besides the target dyadic data of user-item rating matrix in a typical recommender system like an online video streaming system, there are various auxiliary data, e.g. movie’s description (content), user’s location (context), user’s friends (network), user’s binary ratings (feedback), etc. We use Tencent Video as an example to show the aforementioned various auxiliary data in Figure 1.1. All the aforementioned auxiliary data are different but related to the target preference data, which gives us an opportunity to improve the target recommendation performance.

Transfer learning is proposed to extract and transfer knowledge from auxiliary data to improve the target learning task and has achieved great success in text mining, mobile computing, bio-informatics, etc. In this thesis, we aim to design transfer learning methods to make use of auxiliary data for sparsity reduction in collaborative filtering.

\(^{1}\)http://www.amazon.com/
\(^{2}\)http://www.netflix.com/
\(^{3}\)http://t.qq.com/
\(^{4}\)http://www.facebook.com/
1.1 Background

There are two main approaches widely used in recommender systems [4, 48, 60], content-based methods and collaborative filtering techniques. Content-based methods recommend items based on content relevance of the target user’s history taste on other items, while collaborative filtering techniques exploit the user-item preference data and recommend items from like-minded users. In this thesis, we tackle the recommendation problem from a transfer learning view [157], that is, we consider the user-item preference matrix as our target data, and all other information as our auxiliary data, which includes four dimensions of content, context, network and feedbacks. The marriage of transfer learning [157] and collaborative filtering extends the binary categorization of recommendation approaches to three branches,

1. content-based methods,
2. collaborative filtering techniques, and
3. transfer learning solutions in collaborative filtering.

Basically, content-based methods are more suitable for items like news articles, since the similarities between items can be estimated accurately. Content-based methods
usually have good interpretability of the recommendation results, but they ignore collective intelligence from other users. Collaborative filtering techniques have been proposed to learn from like-minded users and proved to be more effective in various competitions (e.g. Netflix movie recommendation, Yahoo! music recommendation) and various real applications (e.g. book recommendation at Amazon, movie recommendation at YouTube). However, collaborative filtering is sensitive to sparse observation data, since it may result in overfitting when we train a prediction model when the observed ratings are few. Transfer learning solutions in collaborative filtering go one step further, and address the sparsity problem via leveraging auxiliary data. Though some works are not proposed in the background of transfer learning, but are indeed making use of auxiliary data, e.g. item’s content information [193], user-item pair’s context of time [103], user’s social networks [138], and user’s implicit feedbacks [128], etc.

1.1.1 Categorization of Auxiliary Data

We give a brief summary of auxiliary data in Table 1.1 from four dimensions of content, context, network and feedback, and for each dimension, we further categorize the data from three directions of user side, item side and frontal side. Note that we use “other users” or “other items” to refer to users or items that are not from the set of users or items of the target user-item preference matrix.

From Table 1.1, we can see that content information are mainly for static auxiliary data, while context information represents dynamic auxiliary data; network information refers to relationships among users and/or items, and feedback information is for various user-involved activities. This categorization is for easy understanding and comparison of different problem settings of transfer learning in collaborative filtering. There may not be a strict boundary between two categories, for example, a user’s generated content of reviews can be considered as or used to infer the user’s feedback on the corresponding items.

Transferring knowledge from different auxiliary data may have different effects on the target collaborative filtering task. For example, the item-side auxiliary data of content (e.g. item’s description) or frontal-side auxiliary data of UGC (e.g. posted message), may help both developers and users understand why the recommender system generates the result. In our four specific problems, we only consider leveraging auxiliary data of feedbacks and social networks, e.g. two-sided implicit feedback-
s, frontal-side binary ratings, frontal-side uncertain ratings, and user-side large-scale heterogeneous social networks.

Table 1.1: A brief summary of auxiliary data in collaborative filtering.

<table>
<thead>
<tr>
<th>Content</th>
<th>Context</th>
<th>Network</th>
<th>Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>user’s static profile of demographics, affiliations, etc.</td>
<td>user’s dynamic context of location, mood, health, etc.</td>
<td>user-user social network of friendship, following, etc.</td>
<td>user’s feedback of rating, browsing, purchasing, collection on other items, etc.</td>
</tr>
<tr>
<td>item’s static description of price, brand, location, etc.</td>
<td>item’s dynamic context of remaining quantities, coupon, etc.</td>
<td>item-item relevance network of links, taxonomy, etc.</td>
<td>item’s feedback of rating, browsing, purchasing, collection by other users, etc.</td>
</tr>
<tr>
<td>user-item pair’s user generated content (UGC) of tags, reviews, comments, etc.</td>
<td>user-item pair’s dynamic context of time, environment, etc.</td>
<td>user-item-user network of sharing an item by one user to another, etc.</td>
<td>user-item pair’s feedback of rating, browsing, purchasing, collection, etc.</td>
</tr>
</tbody>
</table>

1.1.2 Problem Definition

In this section, we give a formal definition of transfer learning in collaborative filtering. In the target data, we have a user-item preference matrix of \( n \) users and \( m \) items, \( \mathbf{R} \), containing \( q \) observations,  

\[
\mathbf{R} = [r_{ui}]_{n \times m} \in (\mathbb{R} \cup \{?\})^{n \times m},  
\]

where the question mark “?” denotes a missing value (unobserved). Note that the observed values in \( \mathbf{R} \) can be 5-star grades of \( \{1, 2, 3, 4, 5\} \), implicit positive feedback of \( \{1\} \), explicit positive and negative feedbacks of \( \{1, 0\} \), or any real number. We use a mask matrix \( \mathbf{Y} \in \{0, 1\}^{n \times m} \) to denote whether the entry \((u, i)\) is observed,  

\[
y_{ui} = \begin{cases} 
1, & \text{if } r_{ui} \text{ is observed,} \\
0, & \text{otherwise.} 
\end{cases} 
\]

where \( \sum_{u=1}^{n} \sum_{i=1}^{m} y_{ui} = q \).

We also have some related auxiliary data in a real system, e.g. item’s static description of price and brand (content), item’s dynamic information of remaining quan-
tities and coupon (context), item’s taxonomy (network), item’s browsing or purchasing logs by other users (feedback), etc. We denote all those auxiliary data as $A$.

Our goal is to predict the missing values in $R$ or rank the items by transferring knowledge from the auxiliary data $A$ to the target data $R$, in an adaptive, collective or integrative way, either memory-based or model-based.

In the sequel, without loss of generality, we use $\{1, 2, 3, 4, 5, ?\}^{n \times m}$ to represent target data to make notations clear. Typically, when the target data are implicit feedbacks in the form of $\{1, ?\}^{n \times m}$, we can adopt some weighting or sampling strategies for negative feedback [152, 151] or convert the implicit positive feedback to explicit pairwise preferences [169]; and when the target data are explicit like/dislike feedbacks in the form of $\{0, 1, ?\}^{n \times m}$, almost all collaborative filtering techniques for $\{1, 2, 3, 4, 5, ?\}^{n \times m}$ can be applied without revision.

1.1.3 Basic Math Formulas

In this section, we will first introduce some basic concepts of matrix transposition, matrix multiplication, matrix regularization, and constraints on matrix variables, and then we put those concepts together in a matrix factorization case.

**Matrix Transposition** Give a matrix $Y = [y_{ui}]_{n \times m} \in \mathbb{R}^{n \times m}$, its transposition $Y^T$ is defined as follows,

$$Y^T = [y_{iu}]_{m \times n} \in \mathbb{R}^{m \times n}$$

where the entry located at $(i, u)$ in $Y^T$ is $y_{ui}$, which is at $(u, i)$ in $Y$. The transposition of a vector is similar and considered as a special case of matrix transposition when either $n$ or $m$ is 1.

**Matrix Multiplication** Given two matrices, $U \in \mathbb{R}^{n \times d}$ and $V \in \mathbb{R}^{m \times d}$, the multiplication of $U$ and the transposition of $V$, $V^T$, is defined as follows,

$$UV^T = [r_{ui}]_{n \times m} \in \mathbb{R}^{n \times m}$$

where $r_{ui} = U_u V_i^T = \sum_{k=1}^{d} U_{uk} V_{ik}$ is the entry located at $(u, i)$ of the resulting matrix.

**Matrix Element-Wise Multiplication** Given two matrices, $Y \in \mathbb{R}^{n \times m}$ and $R \in \mathbb{R}^{n \times m}$, with same size, their element-wise product is defined as follows,

$$Y \odot R = [y_{ui}r_{ui}]_{n \times m} \quad (1.3)$$
where \( y_{ui} \) is the value at \((u, i)\) of the resulting matrix.

**Regularization on Matrix** Given a matrix variable, we may use some regularization techniques to avoid overfitting when we learn the variable. For example, we can use Frobenius norm on the matrix \( V \in \mathbb{R}^{m \times d} \),

\[
||V||_F^2 = \sum_{i=1}^{m} \sum_{k=1}^{d} v_{ik}^2
\]

which is the summation of square values in the matrix \( V \). \(||V_i|||_2^2 = \sum_{k=1}^{d} v_{ik}^2\) denotes a regularization on the \(i\)th row (vector) of the matrix \( V \), \( V_i \in \mathbb{R}^{1 \times d} \).

**Constraints on Matrix** There are two commonly used types of constraints, non-negative constraints and orthornomal constraints. Given a matrix \( V \in \mathbb{R}^{m \times d} \), the non-negative constraints require that every entry in \( V \) is non-negative, \( v_{ik} \geq 0 \), which usually results in good interpretability. The orthornomal constraints is defined not on each entry but on each whole column,

\[
V_k^T V_\ell = \begin{cases} 1, & \text{if } k = \ell, \\ 0, & \text{otherwise.} \end{cases}
\] (1.4)

where \( V_k \) and \( V_\ell \) denote the \( k \)th and \( \ell \)th columns, respectively. Each column of the matrix satisfying orthornomal constraints usually represents a certain latent topic in applications like document clustering in information retrieval.

**Matrix Factorization** We use the regularized matrix factorization model (RSVD) [102] to illustrate the basic math formulas of matrix factorization, and how loss function, regularization and optimization techniques are used for collaborative filtering problems. Given a target user-item rating matrix \( R = [r_{ui}]_{n \times m} \) and a corresponding indicator matrix \( Y \in \{0, 1\}^{n \times m} \), the objective function of RSVD [102] is as follows,

\[
\min_{U, V, b_u, b_i, \mu} \frac{1}{2} \sum_{u=1}^{n} \sum_{i=1}^{m} y_{ui} [(r_{ui} - U_u V_i^T - \mu - b_u - b_i)^2
\]

\[
+ \frac{\alpha_u}{2} ||U_u||^2 + \frac{\alpha_v}{2} ||V_i||^2 + \frac{\beta_u}{2} ||b_u||^2 + \frac{\beta_v}{2} ||b_i||^2],
\]

from which we can see that the matrix \( R \) is factorized into two matrices of \( U \in \mathbb{R}^{n \times d} \) for the user-specific latent feature matrix and \( V \in \mathbb{R}^{m \times d} \) for the item-specific latent feature matrix, and some latent variables of \( b_u \in \mathbb{R} \) for the user \( u \)'s rating bias, \( b_i \in \mathbb{R} \) for the item \( i \)'s rating bias and \( \mu \in \mathbb{R} \) for the global average rating value. More
specifically, for each observed entry $r_{ui}$ with $y_{ui} = 1$, the objective function contains two parts, a loss function and some regularization terms,

- the loss function: $(r_{ui} - U_u V_i^T - \mu - b_u - b_i)^2$ is a square loss function, which is usually adopted to minimize the root mean square error (RMSE); and

- the regularization terms: $\frac{\alpha_u}{2} ||U_u||^2 + \frac{\alpha_v}{2} ||V_i||^2 + \frac{\beta_u}{2} ||b_u||^2 + \frac{\beta_v}{2} ||b_i||^2$ are regularization terms used to avoid overfitting when learning the parameters.

In order to learn the parameters, $U_u, V_i, b_u, b_i, \mu$, we usually adopt some gradient descent based optimization techniques to minimize the whole objective function, e.g. the stochastic gradient descent (SGD) algorithm is used in [102]. Once we have learned the parameters, we can predict the rating of user $u$ on item $i$ via $\hat{r}_{ui} = U_u V_i^T + \mu + b_u + b_i$. Note that the tradeoff parameters, $\alpha_u, \alpha_v, \beta_u$ and $\beta_v$, are usually set empirically [102].

### 1.2 Specific Problems and Challenges

In order to address the data sparsity problem, there are some challenges we have to overcome when we design transfer learning methods in collaborative filtering. In particular,

1. we have to decide “what to transfer” in transfer learning [157]:

   (a) we have to extract some data-independent knowledge that is consistent for both target data and auxiliary data, e.g. for explicit feedbacks of numerical ratings and two-sided (user and item) implicit feedbacks of clicks, we may find some common latent features that can be shared;

   (b) we have to model data-dependent effect besides sharing the data-independent knowledge, e.g. for numerical ratings and frontal-side binary ratings, we may use two sets of parameters, one for data-independent knowledge, the other for data-dependent effect;

   (c) we have to integrate very different user feedbacks, e.g. for numerical ratings and frontal-side uncertain ratings represented as ranges or rating distributions, we may use uncertain ratings as constraints for parameters to learn in the target data;
(d) we have to combine heterogeneous social relations, e.g. for following relations in a microblog and friendship relations in an instant messenger, we may use the friendship relations as an intermediate step of social chains for people recommendation.

2. we have to decide “how to transfer” in transfer learning [157]:

(a) we may design some adaptive algorithm to transfer knowledge from the auxiliary data to the target data with focus on the target recommendation problem only;

(b) we may design some collective algorithm to achieve knowledge sharing and data-dependent effect learning simultaneously with richer interactions between the target data and auxiliary data;

(c) we may design some integrative algorithm to achieve knowledge transfer from very different user feedbacks with additional constraints for the target learning problem;

(d) we may design some large scale integrative algorithm to fuse social relations from heterogeneous social networks with efficiency as the major concern.

1.3 Main Contributions

We develop a new multidisciplinary area of TLCF mainly from two aspects.

First, we propose a novel learning framework to achieve knowledge transfer from various auxiliary data,

$$
\min_{\Theta} \mathcal{E}(\Theta|\mathbf{R}, \mathbf{K}, \mathbf{A}) + \mathcal{R}(\Theta|\mathbf{K}), \text{ s.t. } \Theta \in \mathcal{C}(\mathbf{A})
$$

(1.5)

which contains three terms of loss $\mathcal{E}(\Theta|\mathbf{R}, \mathbf{K}, \mathbf{A})$, regularization $\mathcal{R}(\Theta|\mathbf{K})$ and constraint $\Theta \in \mathcal{C}(\mathbf{A})$. Specifically, $\mathbf{R}$ is the target user-item rating matrix, $\mathbf{A}$ is the auxiliary data, $\mathbf{K}$ is the extracted knowledge from $\mathbf{A}$, and $\Theta$ is the parameter to learn.

Specifically, we instantiate the framework in Eq.(1.5) for each of four new problems studied in this thesis:

1. For collaborative filtering with two-sided implicit feedback (chapter 3),

$$
\min_{\Theta} \mathcal{E}(\Theta|\mathbf{R}, \mathbf{K}) + \mathcal{R}(\Theta|\mathbf{K}), \text{ s.t. } \Theta \in D_{\perp}
$$

(1.6)
where $\mathcal{K}$ is extracted knowledge of the coordinate systems $U_0$, $V_0$, and $D_\perp$ represents orthonormal constraints on the latent user-specific and item-specific feature matrices included in the parameter $\Theta$. We propose a novel adaptive transfer learning model called coordinate system transfer (CST), and two variants of CST, CST-biased and CST-manifold for biased regularization and manifold regularization, respectively.

2. For collaborative filtering with frontal-side binary ratings (chapter 4),

$$\min_{\Theta} \mathcal{E}(\Theta|\mathbf{R}, \mathbf{A}) + \mathcal{R}(\Theta), \text{ s.t. } \Theta \in D_\mathcal{R} \text{ or } \Theta \in D_\perp$$

(1.7)

where both the target data $\mathbf{R}$ and auxiliary data $\mathbf{A}$ are used to be factorized collectively, the shared knowledge of latent features are included in the parameter $\Theta$, and $D_\mathcal{R}$ and $D_\perp$ represent the constraints on latent feature matrices. We propose a novel collective transfer learning framework named transfer by collective factorization (TCF), and two variants of TCF, CMTF (collective matrix tri-factorization) and CSVD (collective singular value decomposition) for different constraints on latent feature matrices.

3. For collaborative filtering with frontal-side uncertain ratings (chapter 5),

$$\min_{\Theta} \mathcal{E}(\Theta|\mathbf{R}) + \mathcal{R}(\Theta), \text{ s.t. } \Theta \in \mathcal{C}(\mathbf{A})$$

(1.8)

where rating instances in the auxiliary data $\mathbf{A}$ are transferred as constraints in the target matrix factorization problem. We propose a novel integrative transfer learning algorithm, transfer by integrative factorization (TIF), to efficiently and effectively achieve knowledge transfer from auxiliary uncertain data.

4. For collaborative filtering with user-side social networks (chapter 6), the learning framework in Eq.(1.5) is significantly simplified without numerical optimization,

$$\mathbf{R} + \mathbf{A} \Rightarrow \Theta$$

(1.9)

where the target data $\mathbf{R}$ and auxiliary data $\mathbf{A}$ are integrated to generate users’ preferences on items as encoded in parameter $\Theta$. We propose a novel memory-based transfer learning solution, social relation based transfer (SORT), for big and sparse data.
Second, we provide a survey of traditional transfer learning works w.r.t. model-based transfer, instance-based transfer and feature-based transfer (chapter 2); and build a relationship between traditional transfer learning methods and TLCF from a unified view (chapter 7).

### 1.4 Notations

We use boldface uppercase letters, such as $\mathbf{Y}$, to denote matrices, and $Y_u$, $Y_i$, $y_{ui}$ to denote the $u$th row, $i$th column and the entry located at $(u, i)$ of $\mathbf{Y}$, respectively. To differentiate the variables in different data sources, we use $\mathcal{X}$, $\mathcal{X}_1$, $\mathcal{X}_2$, $\mathcal{X}_3$ and $\tilde{\mathcal{X}}$ to denote variables for target data, user-side auxiliary data, item-side auxiliary data, auxiliary data without correspondence and frontal-side auxiliary data, respectively, where $\mathcal{X}$ can be the observation matrix $\mathbf{R}$, the indicator matrix $\mathbf{Y}$, the latent matrices $\mathbf{U}$, $\mathbf{V}$ of appropriate dimension, and other variables, etc. $\mathbf{I}$ denotes the identity matrix of appropriate dimension.

We list the commonly used notations in the thesis in Table 1.2. Note that the variables for auxiliary data can be defined in a similar way as that of the target data.

### 1.5 Thesis Outline

The thesis is organized into seven chapters as shown in Figure 1.2. In chapter 2, we first survey transfer learning works w.r.t model-based transfer, instance-based transfer and feature-based transfer, and collaborative filtering works w.r.t. model-based methods and memory-based methods; and then we summarize some closely related works proposed in the background of both transfer learning and non transfer learning from four dimensions of auxiliary data, content, context, network and feedback. In chapters 3, 4, 5 and 6, we study each of the four different problem settings and instantiate the learning framework correspondingly. Finally, in chapter 7, we conclude our work with an interesting link between traditional transfer learning and transfer learning in collaborative filtering from a unified view, and then we list some future research directions.
### Table 1.2: Notations of variables for different data.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbf{R}$</td>
<td>the target <em>user-item</em> rating matrix, $\mathbf{R} \in \mathbb{R}^{n \times m}$</td>
</tr>
<tr>
<td>$n$</td>
<td>number of users in $\mathbf{R}$</td>
</tr>
<tr>
<td>$m$</td>
<td>number of items in $\mathbf{R}$</td>
</tr>
<tr>
<td>$u$</td>
<td>user’s index, $u = 1, \ldots, n$</td>
</tr>
<tr>
<td>$i$</td>
<td>item’s index, $i = 1, \ldots, m$</td>
</tr>
<tr>
<td>$r_{ui}$</td>
<td>rating assigned by user $u$ on item $i$ in $\mathbf{R}$</td>
</tr>
<tr>
<td>$\mathbf{Y}$</td>
<td>the indicator matrix of $\mathbf{R}$, $\mathbf{Y} \in {0, 1}^{n \times m}$</td>
</tr>
<tr>
<td>$y_{ui}$</td>
<td>indicator in $\mathbf{Y}$: $(u, i)$ is observed ($y_{ui} = 1$) or not ($y_{ui} = 0$)</td>
</tr>
<tr>
<td>$d$</td>
<td>number of latent dimensions from matrix factorization of $\mathbf{R}$</td>
</tr>
<tr>
<td>$\mathbf{B}$</td>
<td>inner matrix from matrix tri-factorization of $\mathbf{R}$, $\mathbf{B} \in \mathbb{R}^{d \times d}$</td>
</tr>
<tr>
<td>$\mathbf{U}$</td>
<td>user-specific latent feature matrix, $\mathbf{U} \in \mathbb{R}^{n \times d}$</td>
</tr>
<tr>
<td>$U_u$</td>
<td>user $u$’s latent feature vector, $U_u \in \mathbb{R}^{1 \times d}$</td>
</tr>
<tr>
<td>$b_u$</td>
<td>user $u$’s rating bias, $b_u \in \mathbb{R}$</td>
</tr>
<tr>
<td>$\mathbf{V}$</td>
<td>item-specific latent feature matrix, $\mathbf{V} \in \mathbb{R}^{m \times d}$</td>
</tr>
<tr>
<td>$V_i$</td>
<td>item $i$’s latent feature vector, $V_i \in \mathbb{R}^{1 \times d}$</td>
</tr>
<tr>
<td>$b_i$</td>
<td>item $i$’s rating bias, $b_i \in \mathbb{R}$</td>
</tr>
<tr>
<td>$\mu$</td>
<td>global average rating, $\mu \in \mathbb{R}$</td>
</tr>
<tr>
<td>$\alpha_u$</td>
<td>tradeoff parameter on regularization for $U_u$</td>
</tr>
<tr>
<td>$\alpha_v$</td>
<td>tradeoff parameter on regularization for $V_i$</td>
</tr>
<tr>
<td>$\beta_u$</td>
<td>tradeoff parameter on regularization for $b_u$</td>
</tr>
<tr>
<td>$\beta_v$</td>
<td>tradeoff parameter on regularization for $b_i$</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>tradeoff parameter between two domains</td>
</tr>
</tbody>
</table>

- $\mathcal{X}_1$: variables for target data
- $\mathcal{X}_2$: variables for user-side auxiliary data
- $\mathcal{X}_3$: variables for item-side auxiliary data
- $\mathcal{X}_4$: variables for auxiliary data without correspondence
- $\mathcal{X}^\prime$: variables for frontal-side auxiliary data
Figure 1.2: The organization of thesis.
CHAPTER 2

RELATED WORK

Over the years, transfer learning has received much attention in machine learning research and practice. Researchers have found that a major bottleneck associated with machine learning and collaborative filtering is the lack of labels or ratings to help train a model. In response, transfer learning offers an attractive solution for this problem. Various transfer learning methods are designed to extract the useful knowledge from different but related auxiliary data. In its connection to collaborative filtering, transfer learning has found novel and useful applications. In this chapter, we will first review some most recent developments in transfer learning and collaborative filtering; and then provide a brief overview of collaborative filtering with auxiliary data.

2.1 Transfer Learning Methods

Transfer learning refers to the machine learning framework in which one extracts knowledge from some auxiliary domains to help boost the learning performance in a target domain. Transfer learning as a new paradigm of machine learning and has achieved great success in various areas over the last two decades [32, 157], e.g., text mining [21, 51, 44], speech recognition [213, 120], computer vision (e.g., image [165] and video [218] analysis), recommender systems [115, 116], and ubiquitous computing [233, 210].

For example, transfer learning has many application scenarios, e.g., from Wikipedia documents (auxiliary) to Twitter text (target) [90], from WWW webpages to Flickr images [239], from book recommendation to movie recommendation [115], etc. One fundamental motivation of transfer learning is the so-called data sparsity problem in the target domain, where the data sparsity can be defined by a lack of useful labels or sufficient data in the training set. For example, Twitter text are generated by users and always are unlabeled. When data sparsity happens, overfitting can easily happen when we train a model. Although many machine learning methods, including semi-supervised learning [238, 35], co-training [22] and active learning [204], have been
proposed for addressing the data sparsity problem, in many situations we have to look elsewhere for additional knowledge for learning. We can take the following two views on knowledge transfer,

(i) *In theory*, transfer learning can be considered as a new *learning paradigm*, where most non-transfer learning methods are considered as a special case when learning happens within a single target domain only, e.g., text classification in Twitter, and

(ii) *In applications*, transfer learning can be considered as a new cross-domain *learning technique*, since it explicitly addresses the various aspects of domain differences, e.g., data distribution, feature and label space, noise in the auxiliary data, relevance of auxiliary and target domains, etc. For example, we have to address most of the above issues when we transfer knowledge from Wikipedia documents to Twitter text.

In the following, we survey the recent transfer learning works, where we divide the approaches in three categories of transfer learning methodology, namely,

1. model-based transfer, which studies on how to reuse a *model* trained on some auxiliary data,

2. instance-based transfer, which studies on how to leverage auxiliary data *instances*,

3. feature-based transfer, which studies on how to bridge two domains via feature transformation or *feature* learning.

Discriminative learning methods, which explicitly model the conditional distribution $P_r(Y|X)$, e.g. support vector machines (SVM) [92], maximum entropy (MaxEnt) [15], logistic regression (LR) [80], conditional random field (CRF) [109], have dominated in various classification and regression problems from data mining and machine learning applications, such as text classification and sentiment analysis. As SVM has been recognized as a state-of-the-art model, below, we will use SVM as a representative base model among various discriminative models to illustrate how the labeled data in auxiliary domains can be used to achieve knowledge transfer from auxiliary domains to the target domain. Most techniques can be used in other discriminative models of MaxEnt, LR, CRF, etc.
**Basic SVM**  Given $\ell$ labeled data points $\{(x_i, y_i)\}_{i=1}^{\ell}$ with $x_i \in \mathbb{R}^{d \times 1}$ and $y_i \in \{\pm 1\}$ in the target domain, we have the following optimization problem for the linear SVM with soft margin [186],

$$\min_{w, \xi} \quad \frac{1}{2} ||w||^2 + \lambda \sum_{i=1}^{\ell} \xi_i$$  \hspace{1cm} (2.1)

s.t.  \hspace{1cm} $y_i w^T x_i \geq 1 - \xi_i, \xi_i \geq 0, \ i = 1, \ldots, \ell$

where $w \in \mathbb{R}^{d \times 1}$ is the model parameter, $\xi \in \mathbb{R}^{\ell \times 1}$ are the slack variables, and $\lambda > 0$ is the tradeoff parameter to balance the model complexity $||w||^2$ and loss function $\sum_{i=1}^{\ell} \xi_i$. Solving the convex optimization problem in Eq.(2.1), we obtain a decision function

$$f(x) = w^T x = \sum_{k=1}^{d} w_k x_k.$$  \hspace{1cm} (2.2)

We will describe how SVM and other discriminative models are extended to transfer knowledge from the auxiliary domains to the target domain from three perspectives, namely model-based transfer, instance-based transfer, and feature-based transfer.

### 2.1.1 Model-based Transfer

Basically, these kind of techniques study knowledge transfer from the perspective of model (or equivalently model parameter). For example, we may reuse the model trained from auxiliary domains as a prior for the target domain, and by adding such a prior, knowledge encoded in the auxiliary domains can be transferred. Taking SVM for example, we can transfer the model parameters of a learned SVM in auxiliary domains to a target domain via *biased regularization* [185, 186], which replaces the regularization term $||w||^2$ in Eq.(2.1) with $||w - w_0||^2$ [120, 97],

$$\min_{w, \xi} \quad \frac{1}{2} ||w - w_0||^2 + \lambda \sum_{i=1}^{\ell} \xi_i$$  \hspace{1cm} (2.3)

s.t.  \hspace{1cm} $y_i (w - w_0)^T x_i \geq 1 - \xi_i, \xi_i \geq 0, \ i = 1, \ldots, \ell$

where $w_0 \in \mathbb{R}^{d \times 1}$ is the learned model parameter from the labeled data in auxiliary domains. We can see that the only difference between the standard SVM in Eq.(2.1) and SVM with model-based transfer in Eq.(2.3) is from the regularization terms of
\[||w||_2^2 \text{ and } ||w - w_0||_2^2.\] The knowledge (or the decision function) of the auxiliary domain has been encoded in the model parameter \(w_0\), and the biased regularization term \(||w - w_0||_2^2\) constrains the model parameters \(w\) and \(w_0\) to be similar.

A similar idea has also been explored as maximum a posteriori (MAP),

\[
\max_{w} P_r(w|w_0)P_r\left(\{(x_i, y_i)\}_{i=1}^{\ell}\right|w)
\] (2.4)

where \(P_r(w|w_0)\) encodes prior information \(w_0\) for \(w\). Interestingly, Li and Bilmes [121] derive a similar term of biased regularization from a novel perspective of Bayesian divergence prior of the distribution of labeled data in the auxiliary domain and the predictions of the target classifier, which also gives some theoretical results of adaptation bounds.

The idea of model-based transfer with biased regularization has also been applied to other discriminative models, e.g. MaxEnt and LR [36, 37, 55], CRF [224, 199], multi-layer perceptron (MLP) [120], etc. For example, Eaton et al. [55] propose a novel model-based transfer learning method that exploits the relationships (or transferability) between the auxiliary and target domains (or tasks) via a graph, and find that transferring the average model parameter of multiple models trained from auxiliary domains using biased regularization works well when all of the auxiliary domains are relevant to the target domain.

The model-based transfer learning framework can be further extended to incorporate unlabeled data in the target domain [54], or transfer parameters of multiple trained models from the auxiliary domains [142, 218, 54], etc.

Evgeniou and Pontil [57] consider a different but very interesting formulation in the context of multi-task learning (MTL) [32]. For notational simplicity, we assume there are two domains (or tasks), and the model parameters are as follows,

\[
\tilde{w} = w_0 + \tilde{w}_\Delta, \quad w = w_0 + w_\Delta.
\] (2.5)

We can see that \(w_0 \in \mathbb{R}^{d \times 1}\) is a shared domain-independent variable, and \(\tilde{w}_\Delta \in \mathbb{R}^{d \times 1}\) and \(w_\Delta \in \mathbb{R}^{d \times 1}\) are domain-specific variables. Thus, knowledge transfer is enabled and made bidirectional when we jointly optimize the variables \(w_0, \tilde{w}_\Delta\) and \(w_\Delta\) in the SVM formulation [57]. Jiang [86] studies a similar formulation in a different discriminative model (logistic regression) with feature learning, and successfully applies it to
Daumé III and Marcu [45] introduce MEGA (maximum entropy genre adaptation model), which includes two domain-dependent distributions $\tilde{P}(X, Y)$, $P(X, Y)$ and one domain-independent distribution $\tilde{P}(X, Y)$,

$$\tilde{P}(X, Y), \ P(X, Y), \ P(X, Y)$$

where the labeled data in the auxiliary domain are assumed to be generated by $\tilde{P}(X, Y)$ and $\tilde{P}(X, Y)$, the labeled data in the target domain are assumed to be generated by $P(X, Y)$ and $P(X, Y)$. Finally, the model parameters in $\tilde{P}(X, Y)$ is used as a bridge to bring these two domains together.

In addition to the above methods, many other model-based transfer methods have also been proposed, such as confidence-weighted methods [52, 53], online learning methods [229, 107], etc.

### 2.1.2 Instance-based Transfer

The basic idea of instance-based transfer is that some instances in auxiliary domains are helpful while others may do harm to the learning task in the target domain, and thus we need to select those useful and kick others. One effective way to achieve this is to perform instance weighting according to their importance. Again, taking SVM for example, suppose that we have $\tilde{\ell}$ labeled data in the auxiliary domain, $\{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^{\tilde{\ell}}$ with $\tilde{x}_i \in \mathbb{R}^{d \times 1}$ and $\tilde{y}_i \in \{\pm 1\}$, which can be incorporated into the standard SVM in Eq.(2.1) as follows [214, 122],

$$\begin{align*}
\min_{\mathbf{w}, \xi, \tilde{\xi}} & \quad \frac{1}{2}||\mathbf{w}||^2 + \lambda \sum_{i=1}^{\ell} \xi_i + \lambda \sum_{i=1}^{\tilde{\ell}} \tilde{\rho}_i \tilde{\xi}_i \\
\text{s.t.} & \quad y_i \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i, \ \xi_i \geq 0, \ i = 1, \ldots, \ell \\
& \quad \tilde{y}_i \mathbf{w}^T \tilde{x}_i \geq 1 - \tilde{\xi}_i, \ \tilde{\xi}_i \geq 0, \ i = 1, \ldots, \tilde{\ell}
\end{align*}$$

(2.7)

where $\tilde{\rho}_i \in \mathbb{R}$ is the weight on the data point $(\tilde{x}_i, \tilde{y}_i)$ in the auxiliary domain, which can be estimated via some heuristics [122, 88] or optimization techniques [123]. We can see that the only difference between the standard SVM in Eq.(2.1) and SVM
with instance-based transfer in Eq.(2.7) is from the loss function $\lambda \sum_{i=1}^{\ell} \tilde{\rho}_i \tilde{\xi}_i$ and its corresponding constraints defined on the labeled data in the auxiliary domain. The auxiliary data $\{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^{\ell}$ can be the support vectors of a trained SVM in the auxiliary domain [122, 88] or the whole auxiliary data set [214, 123]. Note that the approach in [214] uses a slightly different base model of linear programming SVM (LP-SVM) [141] instead of the standard SVM in Eq.(2.1). Similar techniques are also developed in the context of incremental learning [174], where support vectors of a learned SVM in the auxiliary domain are combined with labeled data in the target domain with different weight.

Research works have also been done in sample selection bias [77, 222] with $\tilde{P}_r(X) \neq P_r(X)$, $\tilde{P}_r(Y|X) \neq P_r(Y|X)$, and covariate shift [189] with $\tilde{P}_r(X) \neq P_r(X)$, $\tilde{P}_r(Y|X) = P_r(Y|X)$. For example, Bickel et al. [18] explicitly consider the difference of conditional distributions, $\tilde{P}_r(Y|X) \neq P_r(Y|X)$, and propose an alternating gradient descent algorithm to automatically learn the weight of the instances besides the model parameter of Logistic regression. Jiang and Zhai [87] propose a general instance weighting framework from a distribution view considering differences from both marginal distributions, $\tilde{P}_r(X) \neq P_r(X)$, and conditional distributions, $\tilde{P}_r(Y|X) \neq P_r(Y|X)$.

Xiang et al. propose BIG (bridging information gap) [215], a framework to make use of a worldwide knowledge base (e.g. Wikipedia) as a bridge to achieve knowledge transfer from an auxiliary domain with labeled data to a target domain with test data. Specifically, Xiang et al. [215] study the information gap between the target domain and auxiliary domain, and propose a margin related criteria to sample unlabeled data from Wikipedia to fill the information gap, which enables more effective knowledge transfer. Transductive SVM [91] is then trained using the improved data pool of labeled data in the auxiliary domain, unlabeled data from Wikipedia, and test data in the target domain. The proposed framework is studied in cross-domain text classification, sentiment analysis and query classification [215].

2.1.3 Feature-based Transfer

Feature-based transfer is another main transfer learning approach, where algorithms are designed from the perspective of feature space, e.g. feature replication [81, 106],
feature projection [21, 20, 154], dimensionality reduction [153, 155, 156, 191, 42],
feature correlation [166, 105, 227], feature subsetting [182], feature weighting [8], etc.

**Feature Replication** The feature replication or feature augmentation approach [81] is basically a pre-processing step on the labeled data \( \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^{\ell} \) in the auxiliary domain
and labeled data \( \{(x_i, y_i)\}_{i=1}^{\ell} \) in the target domain,

\[
(\tilde{x}_i, \tilde{y}_i) \rightarrow ([\tilde{x}_i^T \hat{x}_i^T 0^T]^T, \tilde{y}_i), \ i = 1, \ldots, \ell \\
(x_i, y_i) \rightarrow ([x_i^T 0^T x_i^T]^T, y_i), \ i = 1, \ldots, \ell
\]

where the feature dimensionality is expanded from \( \mathbb{R}^{d \times 1} \) to \( \mathbb{R}^{3d \times 1} \), and standard supervised learning methods can then be used, e.g. SVM in Eq.(2.1).

As a follow-up work, Kumar et al. [106] further generalize the idea of feature replication via incorporating unlabeled data \( \{x_i\}_{i=\ell+1}^{n} \) in the target domain,

\[
x_i \rightarrow ([0^T x_i^T - x_i^T]^T, +1), \ i = \ell + 1, \ldots, n \\
x_i \rightarrow ([0^T x_i^T - x_i^T]^T, -1), \ i = \ell + 1, \ldots, n
\]

where the processed data points are all with labels now.

The relationship of the feature replication method and the model-based transfer is discussed in [81] and some theoretical results of generalization bound are given in [106]. Feature replication approach have been successfully applied in cross-domain named entity recognition [81], part-of-speech tagging [81] and sentiment analysis [106].

**Feature Projection** Structured correspondence learning (SCL) [21] introduces the concept of pivot features, which possess high frequency and similar meaning in both auxiliary and target domains. Non-pivot features can be mapped to each other via the pivot features from the unlabeled data of both auxiliary and target domains. Learning in SCL [21] is based on the alternating structure optimization (ASO) algorithm [6]. Typically, SCL [21] goes through the following steps. First, it selects \( n_p \) pivot features. Then, for each pivot feature, SCL trains an SVM model in Eq.(2.1) using unlabeled data instances from both domains with labels indicating whether the pivot feature appears in the data instance. In this step it obtains \( n_p \) models such that \( W = [w_{ij}]_{j=1}^{n_p} \in \mathbb{R}^{d \times n_p} \). Third, SCL applies singular value decomposition (SVD) to the model parameters \( W \).
\[ [U \Sigma V^T] = \text{svd}(W) \], and it takes the top \( k \) columns of \( U \) as the projection matrix \( \theta \in \mathbb{R}^{d \times k} \). Finally, it obtains the following transformation for each labeled data point in the auxiliary domain,

\[
(\tilde{x}_i, \tilde{y}_i) \rightarrow ([\tilde{x}_i^T \lambda (\theta^T \tilde{x}_i)^T] \tilde{y}_i), \quad i = 1, \ldots, \tilde{\ell}
\]  

(2.8)

In the above equation, \( \lambda > 0 \) is a tradeoff parameter. The transformed data points is augmented with \( k \) additional features encoded with structural correspondence information between the features from auxiliary and target domains. With the transformed labeled data in the auxiliary domain, SCL can train a discriminative model, e.g. SVM in Eq.(2.1). For any future data instance \( x \), it is transformed via \( x \rightarrow [x^T \lambda (\theta^T x)^T]^T \) before \( x \) is classified by the learned model according to Eq.(2.2).

Blitzer et al. [20] successfully apply SCL [21] to cross-domain sentiment classification, and Prettenhofer and Stein [162, 163] extend SCL [21] with an additional cross-language translator to achieve knowledge transfer from English to German, French and Japanese for text classification and sentiment analysis. Pan et al. [154] propose a spectral learning algorithm for cross-domain sentiment classification using co-occurrence information from auxiliary-domain-specific, target-domain-specific and domain-independent features. They then align domain-specific features from both domains in a latent space via a learned projection matrix \( \theta \in \mathbb{R}^{k \times d} \). In some practical cases, the cross-domain sentiment and review classification performance of [154] is empirically shown to be superior to SCL [21] and other baselines.

**Dimensionality Reduction** In order to bridge two domains to enable knowledge transfer, Pan et al. [153] introduce maximum mean discrepancy (MMD) [23] as a distribution measurement of unlabeled data from auxiliary and target domains,

\[
\left\| \frac{1}{\tilde{\ell}} \sum_{i=1}^{\tilde{\ell}} \phi(\tilde{x}_i) - \frac{1}{n-\ell} \sum_{i=\ell+1}^{n} \phi(x_i) \right\|_2^2
\]  

(2.9)

which is used to minimize the distribution distance in a latent space. The MMD measurement is formulated as a kernel learning problem [153], which can be solved by semi-definite programming (SDP) by learning a kernel matrix \( K \in \mathbb{R}^{(\tilde{\ell}+n-\ell) \times (\tilde{\ell}+n-\ell)} \). Principle component analysis (PCA) is then applied on the learned kernel matrix \( K \) to obtain a low-dimensional representation,

\[
[U \Sigma U^T] = \text{pca}(K), \quad U \in \mathbb{R}^{(\tilde{\ell}+n-\ell) \times k}
\]  

(2.10)
As a result of the transformation, the original data can now be represented with a reduced dimensionality of $\mathbb{R}^{k \times 1}$ in the corresponding rows of $U$. Standard supervised discriminative method such as SVM in Eq.(2.1) can be used to train a model using the transformed labeled data in the auxiliary domain.

Note that as a transductive learning method, the algorithm in [153] cannot be directly used to classify out-of-sample test data, which problem is addressed in [155, 156] by learning a projection matrix to minimize the MMD [23] criteria. Si et al. [191] introduce the Bregman divergence measurement as an additional regularization term in traditional dimensionality reduction techniques to bring two domains together in the latent space.

The EigenTransfer framework [42] introduces a novel approach to integrate co-occurrence information of instance-feature, instance-label from both auxiliary and target domains in a single graph. Normalized cut [188] is then adopted to learn a low-dimensional representation from the graph to replace original data in both target and auxiliary domains. Finally, standard supervised discriminative model, e.g. SVM in Eq.(2.1) is trained using the transformed labeled data in the auxiliary domain. An advantage of EigenTransfer is its ability to unify almost all available information in auxiliary and target domains, allowing the consideration of heterogenous feature and label spaces.

**Feature Correlation** Transferring feature correlation from auxiliary domains to a target domain is introduced in [166, 105, 227], where a feature-feature covariance matrix $\Sigma_0 \in \mathbb{R}^{d \times d}$ estimated from some auxiliary data is taken as an additional regularization term,

$$\lambda w^T \Sigma_0^{-1} w$$  \hspace{1cm} (2.11)

In this equation, the feature-feature correlation information is encoded in the covariance matrix $\Sigma_0$, which can be estimated from labeled or unlabeled data in auxiliary domains. $\Sigma_0$ will constrain the model parameters $w_i$ and $w_j$ of two high-correlated features $i$ and $j$ to be similar, and constrain the low-correlated features to be dissimilar. Such a regularization term is quite general and can be considered in various regularization based learning frameworks to incorporate the feature-feature correlation knowledge. Feature correlation is quite intuitive, and thus it has attracted several practical
applications. For example, Raina et al. [166] transfer the feature-feature correlation knowledge from a newsgroups domain to a webpage domain for text classification, and Zhang et al. [227] study text classification with different time periods.

**Feature Subsetting** Feature selection via feature subsetting has been proposed for named entity recognition in CRF [182], which makes use of labeled data in auxiliary domains and the unlabeled data in the target domain. To illustrate the idea more clearly, we consider a simplified case of binary classification, where $y \in \{\pm 1\}$, instead of sequence labeling [182]. We re-write the optimization problem as follows,

$$
\begin{align*}
\min_{\tilde{\omega}, \xi} & \quad \frac{1}{2} |\tilde{\omega}|^2_2 + \lambda \sum_{i=1}^{\ell} \xi_i \\
\text{s.t.} & \quad \tilde{\omega}^T \phi(x_i, y_i) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, \ell \\
& \quad \sum_{k=1}^{d} |\tilde{\omega}_k|^\gamma \text{dist}(\tilde{E}_k, E_k) \leq \epsilon
\end{align*}
$$

(2.12)

where $E_k = \frac{1}{n-\ell} \sum_{i=\ell+1}^{n} (\phi_k(x_i, +1) P_r(+1|x_i, \tilde{\omega}) + \phi_k(x_i, -1) P_r(-1|x_i, \tilde{\omega}))$ and $\tilde{E}_k = \frac{1}{\ell} \sum_{i=1}^{\ell} \phi_k(\tilde{x}_i, \tilde{y}_i)$ are expected values of the $k$th feature of the joint feature mapping function $\phi(X, Y)$ in the target and auxiliary data, respectively, and $P_r(+1|x_i, \tilde{\omega})$ and $P_r(-1|x_i, \tilde{\omega})$ are the posterior probabilities of instance $x_i$ belonging to classes $+1$ and $-1$, respectively. The parameter $\gamma$ is used to control the sparsity of the model parameter $\tilde{\omega}$, which produces a subset of non-zeros; this is why it is called feature subsetting. The distance $\text{dist}(\tilde{E}_k, E_k)$ can be square distance $(\tilde{E}_k - E_k)^2$ for optimization simplicity [182], which is used to punish highly distorted features in order to bring two domains closer. The trained model $\tilde{\omega}$ will have better prediction performance in the target domain, especially when some features distort seriously in two domains.

**Feature Weighting** Arnold et al. [8] propose a feature weighting (or rescaling) approach to bridge two domains with labeled data in the auxiliary domain and test data in the target domain. Specifically, the $k$th feature of instance $\tilde{x}_j$ in the auxiliary domain is weighted as follows,

$$
\tilde{x}_{j,k} \rightarrow \tilde{x}_{j,k} \frac{E_k(\tilde{y}_j|X_U, \tilde{\omega})}{E_k(\tilde{y}_j|D_L)}
$$

(2.13)

where $E_k(\tilde{y}_j|X_U, \tilde{\omega}) = \frac{1}{n-\ell} \sum_{i=\ell+1}^{n} x_{i,k} P_r(\tilde{y}_j|x_i, \tilde{\omega})$ is the expected value of $k$th fea-
ture (belonging to class $\tilde{y}_j$) in the target domain using the trained MaxEnt model $\tilde{w}$ from auxiliary domain, $\tilde{E}_k(y_j|\tilde{D}_L) = \frac{1}{T} \sum_{i=1}^{T} \tilde{x}_{i,k} \delta(\tilde{y}_j, \tilde{y}_i)$ is the expected value of $k$th feature (belonging to class $\tilde{y}_j$) in the auxiliary domain. The weighted data (feature) in the auxiliary domain then have the same expected values of joint distribution about $k$th feature and class label $y$, $\tilde{E}_k(y|\tilde{D}_L) = E_k(y|X_U, \tilde{w}), \ y \in \mathcal{Y}$. As a result, the two domains are brought closer together. Note that the learning procedure can be iterated with (a) learning $\tilde{w}$ and (b) weighting the feature, and that is the reason the model is called IFT (iterative feature transformation) [8]. Since $E_k(y_j|X_U, \tilde{w})$ is only an estimated value, [8] adopts a common trick to preserve the original feature, which works quite well in NER problems. In particular,

$$\tilde{x}_{j,k} \rightarrow \lambda \tilde{x}_{j,k} + (1 - \lambda) \frac{\tilde{E}_k(y_j|X_U, \tilde{w})}{\tilde{E}_k(y_j|\tilde{D}_L)}$$

(2.14)

where $0 \leq \lambda \leq 1$ is a tradeoff parameter.

In the same spirit, other feature-based transfer methods have also been proposed, such as distance minimization [14], feature clustering [43, 133], kernel mapping [236], etc.

### 2.1.4 Summary

In transfer learning, a common method to achieve knowledge transfer from auxiliary domains to the target domain is via cross-domain discriminative learning. These methods can be categorized into model-based, instance-based and feature-based approaches, and share the following common properties: they (i) adapt a trained model to fit the data in the target domain, (ii) leverage relevant instances in the auxiliary domains to increase the training data pool in the target domain, and (iii) transform the feature space to well bridge auxiliary and target domains. Most model-based transfer algorithms are quite efficient, since they do not maintain the original auxiliary data, but make use of the trained model only. Instance-based transfer and feature-based transfer can usually be interpreted in a two-stage approach where in the first stage, instance selection/weighting or feature learning is carried out and in the second stage, a model learning step is conducted.
2.2 Collaborative Filtering Techniques

Similar to the discovery of behavior correlation [194] in social networks, the fundamental assumption in collaborative filtering is taste non-independence, where a user $u$’s taste on item $i$, denoted as $r_{ui}$, is dependent on other users’ taste on items. There are two main branches of collaborative filtering methods, memory-based methods and model-based methods [4]. We briefly review some very popular methods in this section, and would like to recommend [4, 48, 60] to readers for more information.

2.2.1 Memory-based Methods

Memory-based methods in collaborative filtering can be categorized into two branches, user-based approaches and item-based approaches.

User-based approaches

Pearson correlation coefficient (PCC) [172] is a similarity measure of two users $u$ and $w$ based on the ratings on their commonly rated items,

$$
PCC(u, w) = \frac{\sum_i y_{ui}y_{wi}(r_{ui} - m_u m_w)}{\sqrt{\sum_i y_{ui}y_{wi}(r_{ui} - m_u)^2} \sqrt{\sum_i y_{ui}y_{wi}(r_{wi} - m_w)^2}},$$

where $m_u = \sum_i y_{ui}r_{ui} / \sum_i y_{ui}$ and $m_w = \sum_i y_{ui}r_{wi} / \sum_i y_{ui}$ are the average rating of user $u$ and $w$ on the commonly rated items, respectively.

The normalized similarity between users $u$ and $w$ can then be calculated as follows,

$$s_{uw} = \frac{PCC(u, w)}{\sum_{u' \in N_u} PCC(u', w)},$$

where $N_u$ is the set of $k$ nearest neighboring users of user $u$ according to PCC.

Finally, we can predict the rating $r_{ui}$ of user $u$ on item $i$ as,

$$\hat{r}_{ui} = \bar{r}_u + \sum_{w \in N_u} y_{wi}s_{uw}(r_{wi} - m_w),$$

where $\bar{r}_u = \sum_i y_{ui}r_{ui} / \sum_i y_{ui}$ is the average rating of user $u$ [172] on all items. Note that in real implementation, practitioners usually replace $m_w$ with $\bar{r}_w$ for simplicity,
since $\bar{r}_w$ is fixed for user $w$,

$$\hat{r}_{ui} = \bar{r}_u + \sum_{w \in N_u} y_{wi}s_{uw}(r_{wi} - \bar{r}_w)$$

$$= \bar{r}_u + \sum_{w \in N_u} \frac{y_{wi}}{\sum_{u' \in N_u} PCC(u, u')} PCC(u, w)(r_{wi} - \bar{r}_w)$$

$$= \bar{r}_u + \frac{\sum_{w \in N_u} y_{wi}PCC(u, w)(r_{wi} - \bar{r}_w)}{\sum_{w \in N_u} PCC(u, w)}$$

(2.15)

which is well known as the Resnick’s formula [172] in collaborative filtering.

Note that parallel to PCC, there is another similarity measure called VS [25] or cosine similarity,

$$VS(u, w) = \frac{\sum_i y_{ui}y_{wi}r_{ui}r_{wi}}{\sqrt{\sum_i y_{ui}y_{wi}r_{ui}^2} \sqrt{\sum_i y_{ui}y_{wi}r_{wi}^2}}$$

which can be used either in user-based approaches or item-based approaches.

**Item-based approaches**

Similar to the user-based approaches, we can predict the rating $r_{ui}$ of user $u$ on item $i$ as,

$$\hat{r}_{ui} = \bar{r}_i + \sum_{j \in N_i} y_{uj}s_{ij}(r_{uj} - m_j),$$

where $N_i$ is the set of $k$ nearest neighboring items of item $i$ according to PCC, and,

$$s_{ij} = \frac{PCC(i, j)}{\sum_{i' \in N_i} PCC(i, i')},$$

$$PCC(i, j) = \frac{\sum_u y_{ui}y_{uj}(r_{ui} - m_i)(r_{uj} - m_j)}{\sqrt{\sum_u y_{ui}y_{uj}(r_{ui} - m_i)^2} \sqrt{\sum_u y_{ui}y_{uj}(r_{uj} - m_j)^2}}$$

where, $m_i = \sum_u y_{ui}y_{uj}r_{ui}/\sum_u y_{ui}y_{uj}$, $m_j = \sum_u y_{ui}y_{uj}r_{uj}/\sum_u y_{ui}y_{uj}$, and $\bar{r}_i = \sum_y y_{ui}r_{ui}/\sum_u y_{ui}$. Similarly, we can replace $m_j$ with $\bar{r}_j$ for simplicity,

$$\hat{r}_{ui} = \bar{r}_i + \sum_{j \in N_i} y_{uj}s_{ij}(r_{uj} - \bar{r}_j)$$

$$= \bar{r}_i + \frac{\sum_{j \in N_i} y_{uj}PCC(i, j)(r_{uj} - \bar{r}_j)}{\sum_{j \in N_i} PCC(i, j)}$$

(2.16)

Memory-based methods usually have good interpretability, but model-based methods are state-of-the-art in various rating or link prediction competitions.
2.2.2 Model-based Methods

There are various model-based methods in collaborative filtering, e.g. MMMF (maximum margin matrix factorization) [196, 171, 211, 212], RBM (restricted Boltzmann machines) [178], RW (random walk) [219], EigenRank [129], MC (matrix completion) [95], etc. In this section, we mainly focus on three algorithms, which will be heavily echoed in later chapters, probabilistic matrix factorization (PMF) [177], non-negative matrix factorization [113], and singular value decomposition (SVD) [181].

Probabilistic matrix factorization (PMF)

Probabilistic matrix factorization (PMF) [177, 176] is a recently proposed method for missing value prediction in a single CF matrix. The main assumption under PMF is the conditional probability of the observed value \( r_{ui} \) over the user-specific and item-specific latent vectors, \( U_u \in \mathbb{R}^{1 \times d} \) and \( V_i \in \mathbb{R}^{1 \times d} \), respectively,

\[
p(r_{ui} | U_u :: V_i) = \mathcal{N}(r_{ui} | U_u V_i^T, \alpha^{-1}),
\]

where \( \mathcal{N}(x | \mu, \alpha^{-1}) = \sqrt{\frac{\alpha}{2\pi}} e^{-\frac{\alpha(x-\mu)^2}{2}} \) is the Gaussian distribution with mean \( \mu \) and precision \( \alpha \). Typically, the observed user-item rating matrix \( R \) is factorized into two latent feature matrices, \( U \in \mathbb{R}^{n \times d} \) and \( V \in \mathbb{R}^{m \times d} \) [177],

\[
R \sim UV^T
\]

where the missing value can then be predicted as \( \hat{r}_{ui} = U_u V_i^T \). For implementation, we usually solve the following optimization problem in vector forms, which regularizes \( U_u \) and \( V_i \) for each observation \( r_{ui} \), with the zero-mean spherical Gaussian priors for the latent feature vectors [177],

\[
\min_{U,V} \frac{1}{2} \sum_{u=1}^{n} \sum_{i=1}^{m} y_{ui} [(r_{ui} - U_u V_i^T)^2 + \frac{\alpha_u}{2} ||U_u||_F^2 + \frac{\alpha_v}{2} ||V_i||_F^2],
\]

which can be solved via alternating least square (ALS) algorithms [13] in closed form alternatively, (a) update each \( U_u \) separately when \( V \) is fixed, and (b) update each \( V_i \) separately when \( U \) is fixed.

Non-negative matrix factorization (NMF)

In NMF [113, 98], the observation matrix \( R \) is factorized into two non-negative latent feature matrices, \( U \in \mathbb{R}^{n \times d}_+ \) and \( V \in \mathbb{R}^{m \times d}_+ \), where \( \mathbb{R}_+ \) is the set of non-negative real
numbers,
\[
R \sim UV^T, \text{ s.t. } U \geq 0, V \geq 0
\]  
(2.19)

where the missing value can be predicted similarly as \( \hat{r}_{ui} = U_u V_i^T \). Note, the loss function can be defined on (a) Euclidean distance same as that in PMF,
\[
\sum_{u=1}^{n} \sum_{i=1}^{m} y_{ui} [(r_{ui} - U_u V_i^T)^2],
\]
or (b) Kullback-Leibler divergence over the observed and recovered values,
\[
\sum_{u=1}^{n} \sum_{i=1}^{m} y_{ui} (r_{ui} \log \frac{r_{ui}}{U_u V_i^T} - r_{ui} + U_u V_i^T).
\]

where the optimization problems can be solved iteratively via multiplicative update rules [113, 98].

The NMF model is closely related to clustering-based methods in collaborative filtering [190, 49, 220], probabilistic latent semantic analysis (PLSA) [79], etc.

**Singular Value Decomposition (SVD)**

We refer SVD as low-rank matrix tri-factorization with orthonormal constraints [16, 66],
\[
R \sim U \Sigma V^T, \text{ s.t. } U^T U = I, V^T V = I,
\]  
(2.20)

where the orthonormal constraints are introduced to make the solution unique [49]. The rating assigned by user \( u \) on item \( i \) can thus be predicted as \( \hat{r}_{ui} = U_u \Sigma V_i^T \). In this section, we introduce works of both SVD and principal component analysis (PCA).

As far as we know, Billsus et al. [19] are the first that apply SVD in collaborative filtering. Specifically, for a target user \( u \), the proposed approach contains four steps. First, it converts the original user-item rating matrix \( R \) (excluding the row of user \( u \)) to a full feature-item matrix. Second, it applies SVD on the obtained full feature-item matrix to reduce the dimensionality of feature. Third, it trains a traditional machine learning model (e.g. neural network in [19]) with user \( u \)'s rating as label. Finally, it predicts the missing rating for the target user \( u \) using the trained model.
Sarwar et al. [181] propose two SVD-based approaches in collaborative filtering, one for 5-star numerical rating prediction, and the other for top-$N$ recommendation of implicit purchase data.

For 5-star numerical rating prediction, the training procedure contains two steps [181]. First, it converts the original rating matrix $R$ to $\tilde{R}$ as follows [181],

$$r_{ui} \rightarrow \tilde{r}_{ui} = \begin{cases} r_{ui} - \bar{r}_u, & \text{if } y_{ui} = 1 \text{ (rated)} \\ \bar{r}_i - \bar{r}_u, & \text{if } y_{ui} = 0 \text{ (not rated)} \end{cases}$$

where $\bar{r}_u = \frac{\sum_{i=1}^{m} y_{ui} r_{ui}}{\sum_{i=1}^{m} y_{ui}}$ is user $u$’s average rating, and $\bar{r}_i = \frac{\sum_{u=1}^{n} y_{ui} r_{ui}}{\sum_{u=1}^{n} y_{ui}}$ is the item $i$’s average rating. Second, it applies SVD on the full obtained matrix $\tilde{R}$ [181],

$$\tilde{R} = U \Sigma V^T.$$

Then, missing ratings can be estimated using the obtained latent variables [181],

$$\hat{r}_{ui} = \bar{r}_u + U_u \Sigma V_i^T,$$

where the average rating $\bar{r}_u$ is added back to the prediction rule.

For top-$N$ recommendation of implicit purchase data, the training procedure contains two steps [181]. First, it converts the original rating matrix $R$ to $\tilde{R}$ as follows [181],

$$r_{ui} \rightarrow \tilde{r}_{ui} = \begin{cases} 1, & \text{if } y_{ui} = 1 \text{ (purchased)} \\ 0, & \text{if } y_{ui} = 0 \text{ (not purchased)} \end{cases}$$

Second, it applies SVD on the full matrix $\tilde{R}$, $\tilde{R} = U \Sigma V^T$. The recommendation procedure contains three steps [181]. First, it uses $U \sqrt{\Sigma} \in \mathbb{R}^{n \times d}$ as users’ new profiles. Second, it finds neighbors for each user $u$ using Cosine similarity. Third, it finds top-$N$ most frequent items for each user $u$ from the neighbors of user $u$.

There are some other work [179, 41] using SVD on the full rating matrix. Sarwar et al. [180] extend their previous work [181, 179], and use folding-in techniques to achieve incremental SVD for new users and items, while this solution is not optimal since the updated latent matrices are not orthonormal. Pryor [164] assumes that the rating matrix $R$ is full, and apply SVD on $R$ to find latent features and singular values for analysis and recommendation for new coming users with few ratings.
The system of Eigentaste by Goldberg et al. [65] assumes that all users rate all items from a gauge set of size $g$, and then PCA is applied on the item-item full correlation matrix. More specifically, the training procedure contains four steps. First, it calculates a full item-item correlation matrix $C \in \mathbb{R}^{g \times g}$. Second, it applies PCA on the full correlation matrix $C = V\Sigma V^T$, where $V \in \mathbb{R}^{g \times d}$ and $d$ is the latent dimension number ($d = 2$ in [65]). Third, it represents each user $u$’s (of $n$ users) feature as $f_u = \sum_{i=1}^{g} y_{ui} r_{ui} V_i \in \mathbb{R}^{1 \times d}$. Fourth, it clusters those $n$ users according to the obtained low-rank ($d = 2$) representation. The prediction procedure contains three steps. First, for a newly coming user $w$, it gets the representation $f_w = \sum_{i=1}^{g} y_{wi} r_{wi} V_i \in \mathbb{R}^{1 \times d}$. Second, it finds the cluster for user $w$. Third, it recommends the items preferred by the users in the same cluster using the average rating of users in the same cluster.

There are some other work using PCA on full item-item correlation matrix [71, 59, 110, 148].

We can see that the commonality of the aforementioned works [19, 181, 59, 164, 65] is full matrix, either assume the matrix is full or the missing ratings are first removed using a certain pre-processing procedure. There are also some works that apply SVD on a full matrix in an iterative way [195, 108], and other works [27, 96] define the objective function on observed ratings only.

### 2.2.3 Summary

In collaborative filtering, various methods have been studied and adopted in real applications. Typically, memory-based methods inherit the advantages of good interpretability and maintenances, while model-based methods usually achieve better prediction accuracy as proved in various competitions. However, in a real system, usually different algorithms from both memory-based methods and model-based methods will be integrated together to obtain best and stable performance.
2.3 Collaborative Filtering with Auxiliary Data

2.3.1 Collaborative Filtering with Auxiliary Content

Various works have been proposed to combine user-side and/or item-side metadata and the target user-item preference matrix of ratings. For example, Basu et al. [11] represent both item’s content and the ratings as features on which classification or regression models can be trained; Claypool et al. [38] and Melville et al. [146] argument the memory-based prediction rule [172] with item’s content information; Gunawardana et al. [69, 70] generalize restricted Boltzman machine (RBM) with item’s content information; Singh et al. [193] propose to collectively factorize the user-item rating matrix and the item-content matrix; Yoo et al. [221] adopt non-negative matrix factorization to collectively factorize three matrices of user-item ratings, user-profile and item-content; Stern et al. [197] extend maximum margin matrix factorization (MMM-F) model to include user-side and item-side metadata; Agarwal et al. [5] and Zhang et al. [223] extend matrix factorization method with regression priors from user-side and item-side content information, etc.

With the fast development of social websites, user generated content (UGC) of tags and reviews are also used to improve the recommendation performance. For example, Sen et al.[187] propose to make use of a user’s preference on tags to help predict the user’s preference on movies; Zhen et al. [231] and Guan et al. [68] extend the matrix factorization method with an additional manifold regularization term calculated from the tags assigned by users; Zhou [237] incorporate tag information via collective matrix factorization; Lippert et al. [126] and Jakob et al. [82] integrate review data via joint matrix factorization; Zhang et al. [225] use review or sentiment to generate virtual ratings for the target user-item rating matrix to reduce the data sparsity problem, etc.

2.3.2 Collaborative Filtering with Auxiliary Context

Karatzoglou et al. [93] propose to use tensor factorization method to address the context-aware problem, where each slice of user-item rating matrix represent a users’ preference data in a typical context, e.g. user’s state of hungry or full. Koren [103] investigate the context information of time via extending the matrix factorization method by learning the bias for each time period. Xiong et al. [217] study the temporal effect via tensor factorization with multiple slices of user-item rating matrices at different time
2.3.3 Collaborative Filtering with Auxiliary Networks

Kautz et al. [94] introduce the idea of social chain to make recommendation, which is similar to that of our daily life. For example, a student would turn to his or her supervisor for advice on which course to take, where the social chain can be represented as student $\sim$ supervisor $\sim$ course. Works on collaborative filtering using social networks can mainly be categorized into two branches, memory-based methods extending the well known Resnick’s formula [172], and model-based methods extending the matrix factorization model or random walk model.

The trust-aware recommender systems [143, 144] and FilmTrust system [63] replaced the nearest neighbors and similarities in the Resnick’s formula [172] with trusted users and trust values calculated via a depth first search algorithm MoleTrust [9], and a breadth first search algorithm TidalTrust [61, 62], respectively. The trust-based weighting (or filtering strategies) [150] combine the similarities and trust values (or the nearest neighbors and trusted users) in order to improve the overall prediction performance. Besides the trust information in social networks, distrust information are also studied in collaborative filtering [209, 208]. The TrustWalker [83] approach extends the random walk model to include both the trust network and the item-item similarities.

With the great success of matrix factorization based method in Netflix competitions, various works are proposed to extend the matrix factorization method with social networks. For example, Ma et al. [138, 140] incorporate the social networks via collective factorization of the user-item rating matrix and the user-user network; Ma et al. [135, 136] integrate the social relations via introducing an improved rating generating function of probabilistic matrix factorization; Liu et al. [127], Jamali et al. [85] and Ma et al. [137, 139] extend the probabilistic matrix factorization with additional regularization terms from the social networks; Vasuki et al. [206, 207] propose to use singular value decomposition on a blended matrix of the user-item rating matrix and the social network matrix, and then use the learned latent variables for rating prediction.

During the KDD-Cup 2011 competition of Yahoo! music recommendation, various works are proposed to incorporate the item-side network, or more specifically, the taxonomy of track, album, artist and genre. For example, Koenigstein et al. [99] pro-
pose to use bias to capture the dependency information between items induced from the taxonomy. The taxonomy information is proved to be useful especially for items with few ratings, e.g. track and album.

2.3.4 Collaborative Filtering with Auxiliary Feedbacks

Li et al. [116, 115] propose to transfer cluster-level rating patterns from a book domain to a movie domain. Cao et al. [29] and Zhang [228] study on leveraging user-side feedbacks of numerical ratings to help the target rating prediction task, where the relationships or relevance between the target data and auxiliary data can be learned to avoid negative transfer [147]. Pan et al. [160] propose to transfer both user-side and item-side implicit feedbacks to the target numerical rating matrix in an adaptive way, and later Pan et al. [159] further study transferring knowledge from frontal-side explicit feedbacks of binary ratings via a collective model.

There are also some works integrating implicit user feedbacks into the matrix factorization methods, e.g. implicit feedbacks of “whether rated” [104, 128], “whether purchased” [226], etc.

Figure 2.1: A summary of works on collaborative filtering using auxiliary data.
2.3.5 Summary

We summarize the aforementioned work in Figure 2.1. We can see that there are much work already on using content and networks, but relatively few on context and feedbacks. In particular, for content, frontal-side UGC in various social networks will be a rich source for knowledge extraction and transfer; for context, real-time knowledge extraction and transfer based on user’s and/or item’s state is a fertile area to explore for performance improvement; for networks, little works making use of frontal-side networks are reported yet, e.g. the network formed by the “sharing” and “forwarding” functionalities in Twitter and microblogs; for feedbacks, there are still various heterogeneous user feedbacks not fully explored yet, e.g. collection, browsing, watching, reading, download, purchasing, etc, especially for user feedbacks in mobile devices.

One major challenge in leveraging auxiliary data for collaborative filtering is the data heterogeneities, content vs. rating, context vs. rating, network vs. rating, and feedbacks vs. rating. More specifically, we have to answer some fundamental questions like “what to transfer”, “how to transfer” and “when to transfer” in transfer learning [157].
CHAPTER 3

TRANSFER LEARNING IN COLLABORATIVE FILTERING WITH TWO-SIDED IMPLICIT FEEDBACKS

Data sparsity is a major problem for collaborative filtering (CF) techniques in recommender systems, especially for new users and items with very few ratings. We observe that, while our target data are sparse for CF systems, related and relatively dense auxiliary data may already exist in some other more mature application domains. In this chapter, we address the data sparsity problem in a target domain by transferring knowledge about both users and items from some auxiliary data sources.

We observe that in different domains the user feedbacks are often heterogeneous such as ratings vs. clicks. Our solution is to integrate both user and item knowledge in auxiliary data sources through a principled matrix-based transfer learning framework that takes into account the data heterogeneity. In particular, we discover the principle coordinates of both users and items in the auxiliary data matrices, and transfer them to the target domain in order to reduce the effect of data sparsity. We describe a method, which is known as coordinate system transfer (CST), and demonstrate its effectiveness in alleviating the data sparsity problem in collaborative filtering. We show that our proposed method can significantly outperform several state-of-the-art solutions for this problem. Furthermore, high-order generalization of CST and detailed discussions of related 1-order, 2-order and high-order methods are also given.

3.1 Introduction

Collaborative Filtering (CF) [64, 172] was proposed to predict the missing values in an incomplete matrix, i.e. user-item rating matrix. A major difficulty in CF is the data sparsity problem, because most users can only access a limited number of items. This is especially true for newly created online services, where overfitting can easily happen when we learn a model, causing significant performance degradation.
For a typical recommendation service provider such as movie rental services, there may not be sufficient user-item rating records of a new customer or a new product. Mathematically, we call such data sparse, where the useful information is scattered and few. Using these data matrices for recommendation may result in low-quality results due to overfitting. To address this problem, some service providers turn to explicitly ask the newly registered customers to rate some selected items, such as some most sparsely rated jokes in a joke recommender system [65, 148]. However, methods like this may degrade the customer’s experience and satisfaction with the system, or even cause the customer churn if the customer is pushed too much.

More recently, researchers have introduced transfer learning methods [157, 165, 44] for solving the data sparsity problem [138, 193, 115, 116]. These methods are aimed at making use of the data from other recommender systems, referred to as the auxiliary domain, and transfer the knowledge that are consistent from different auxiliary domains to the target domain. Works of [161, 149] apply multi-task learning (MTL) [33] to collaborative filtering, but their studied problem is different, as they do not consider any auxiliary information sources. Instead, Phuong et al. [161] formulate a multiple binary classification problem in the same CF matrix, one for each user; and Ning et al. [149] first select some closely related users of the target user and then formulates a multiple regression problem, one for each user in the set of target user and his closely related users. Ma et al. [138] and Singh [193] use common latent features when factorizing two matrices and Li et al. [115, 116] consider the cluster-level rating patterns as potential candidates to be transferred from the auxiliary domain. However, there are two limitations of these methods due to certain assumptions that are often not met in practice. Firstly, they require the user preferences expressed in the auxiliary and target domains to be explicit feedbacks such as numerical ratings of \{1, 2, 3, 4, 5, ?\}, where “?” denotes an unobserved rating or missing value. In practice, the data from the auxiliary domain may not be ratings at all but implicit feedbacks, such as user click records, represented as \{1, ?\}. Secondly, methods like [115, 116] do not make use of any existing correspondences among users or items from the target domain and auxiliary domain.

In this chapter, we propose a principled matrix factorization based framework called coordinate system transfer (CST) for transferring both user and item knowledge from an auxiliary domain.
The organization of the chapter is as follows. We give a formal definition of the problem in Section 3.2 and then describe our solution in detail in Section 3.3. We present experimental results on real-world data sets in Section 3.4, and discuss about some related work in Section 3.5. Finally, we give some concluding remarks and future works in Section 3.6.

3.2 Collaborative Filtering with Implicit Feedbacks

3.2.1 Problem Definition

In our problem setting, we have a target domain where we wish to solve our CF problem. In addition, we also have an auxiliary domain which is similar to the target domain. The auxiliary domain can be partitioned into two parts: a user part and an item part, which share common users and items, respectively, with the target domain. We call them the user side and item side, respectively.

We use \( n \) as the number of users and \( m \) the number of items in the target domain, and we use \( R \in \{1, 2, 3, 4, 5, ?\}^{n \times m} \) as the observed rating matrix, where rating “1, 2, 3, 4, 5” denotes the degree of preference, e.g. 1 for bad, 2 for fair, 3 for good, 4 for excellent, and 5 for perfect. Here, \( r_{ui} \) is the rating given by user \( u \) on item \( i \). \( Y \in \{0, 1\}^{n \times m} \) is the corresponding indicator matrix, with \( y_{ui} = 1 \) if user \( u \) has rated item \( i \), and \( y_{ui} = 0 \) otherwise.

For the auxiliary domain, we use \( R_1, R_2 \) to denote data matrices from auxiliary data sources that share common users and items with \( R \), respectively. Note that there is an one-one mapping between the users in \( R \) and \( R_1 \), and an one-one mapping between items in \( R \) and \( R_2 \). The sets of observed user feedbacks of the two auxiliary data matrices are implicit feedbacks, \( R_1 = \{1, ?\}^{n \times m_1} \), \( R_2 = \{1, ?\}^{n_2 \times m} \), where “1” denotes implicit information only, e.g. “clicked” or “rated”. Hence, we can see that the semantic meaning of numerical ratings in \( R \) and implicit ratings of \( R_1, R_2 \) are different.

Our goal is to make use of auxiliary data \( R_1, R_2 \) to help predict the missing values in \( R \), which is illustrated in Figure 3.1.
Figure 3.1: Illustration of transfer learning from two-sided implicit feedbacks via coordinate system transfer (CST).

Table 3.1: Matrix illustration of coordinate system transfer (CST).

<table>
<thead>
<tr>
<th>Training Data</th>
<th>Auxiliary Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>CST (user + item)</td>
<td>( R \sim UBV^T )</td>
</tr>
<tr>
<td>( R_2 \sim U_2 \Sigma_2 V_2^T ), ( V_0 = V_2 )</td>
<td></td>
</tr>
</tbody>
</table>

Knowledge sharing: \( U \approx U_1, V \approx V_2 \)
Value domain: \( (U, V), (U_1, V_1), (U_2, V_2) \in D_\perp \)
\( D_\perp = \{(U, V)| U \in \mathbb{R}^{n \times d}, U^T U = I, V \in \mathbb{R}^{m \times d}, V^T V = I\} \)

3.2.2 Challenges

We have to address the following challenges for our studied problem as shown in Figure 3.1,

1. we have to extract some knowledge from the auxiliary domain of implicit feed-
backs, which should be domain independent (or consistent) and useful for the
target domain of explicit numerical ratings, though the semantic meanings of
the observed ratings from auxiliary domain ($R_1, R_2$) and target domain ($R$) are
very different;

2. we have to decide how to transfer the extracted knowledge to the target domain,
and thus improve the performance of missing value prediction.

We observe that these two challenges are correlated, and are similar to the two funda-
mental questions in transfer learning; that is, in deciding “what to transfer” and “how
to transfer” in transfer learning [157].

Among existing transfer learning methods in collaborative filtering [138, 193, 115, 116], Ma et al. [138] leverage user-side auxiliary data of explicit social relations, S-ingh [193] make use of item-side auxiliary data of content information, and Li et al. [115, 116] turn to explicit numerical ratings from a different but related domain. Thus, as far as we know, there is no previous work studying a same problem as ours (see Figure 3.1).

### 3.2.3 Overview of Our Solution

Our main idea is to discover the common latent information which is shared in both
auxiliary and target domains. Although the user feedbacks in auxiliary and target do-
 mains are heterogeneous, we find that for many users, their latent tastes which rep-
resent their intrinsic preference structure in some subspace are similar. For example,
for movie renters, their preferences on drama, comedy, action, crime, adventure, doc-
umentary and romance expressed in their explicit rating records in the target domain,
are similar to that in their implicit click records on other movies in an auxiliary do-
main. We assume that there is a finite set of tastes, referred to as principle coordinates,
which characterize the domain independent preference structure of users and thus
can be used to define a common coordinate system for representing users. On the oth-
er hand, we can have another coordinate system for representing items’ main factors,
i.e. director, actors, prices, 3ds Max techniques, etc.

In the proposed transfer learning solution, we first pre-process the auxiliary im-
plicit rating matrices $R_1, R_2$ to obtain two full matrices $\mathbf{\bar{R}}_1, \mathbf{\bar{R}}_2$, and then use low-rank
singular value decomposition (SVD) to discover the principle coordinates for constructing the coordinate systems for users and items, which answers the question of “what knowledge to transfer”. We then use techniques of initialization and regularization in order to transfer the coordinate systems for modeling target domain data, which addresses the problem of “how to transfer the knowledge”. We illustrate the main idea in Figure 3.1 and Table 3.1.

3.3 Coordinate System Transfer

In our solution, known as coordinate system transfer or CST, we first discover an auxiliary domain subspace where we can find some principle coordinates. These principle coordinates can be used to bridge two domains, and ensure knowledge transfer. Our algorithm is shown in Figure 3.2, which is described in two major steps, as described below.

3.3.1 Step 1: Coordinate System Construction

In step 1, we first find the principle coordinates of the auxiliary domain data. The principle coordinates in a CF system can be obtained via low-rank singular value decomposition (SVD) [16] on a full matrix with imputation of zeros for the missing values, which is also known as pure SVD in the community of recommender systems [181, 41]. Specifically, we convert the auxiliary implicit rating matrix (e.g. \( \mathbf{R}_1 \)) to a full imputed rating matrix (\( \tilde{\mathbf{R}}_1 \)) in a similar way as that of [181, 41],

\[
\tilde{r}_{ui} = \begin{cases} 
1, & \text{if } y_{ui} = 1 \text{ (clicked)} \\
0, & \text{if } y_{ui} = 0 \text{ (not clicked)}
\end{cases}
\]  

(3.1)

and then we apply SVD [16] on the imputed matrix \( \tilde{\mathbf{R}}_1 \),

\[
\min_{U_1, V_1, \Sigma_1} ||\tilde{\mathbf{R}}_1 - U_1 \Sigma_1 V_1^T||^2_F, \quad \text{s.t. } U_1^T U_1 = I, \ V_1^T V_1 = I
\]  

(3.2)

where \( \Sigma_1 = \text{diag}(\sigma_1, \ldots, \sigma_j, \ldots, \sigma_d) \) is a diagonal matrix, \( \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_d \geq 0 \) are eigenvalues, and the constraints \( U_1^T U_1 = I, \ V_1^T V_1 = I \) ensure that their columns are othornormal. Similarly, for the auxiliary implicit rating matrix \( \mathbf{R}_2 \), we have the imputed rating matrix \( \tilde{\mathbf{R}}_2 \) and factorized latent variables \( U_2, V_2, \Sigma_2 \). Note that each column of \( U_1, V_1, U_2, V_2 \) represents a semantic concept; i.e. user taste [148] in collaborative filtering or document theme [47] in information retrieval. Those columns
are the principle coordinates in the low-dimensional space, and for this reason, we call our approach the coordinate system transfer.

**Definition** (Coordinate System) A coordinate system is a matrix with columns of orthonormal bases (principle coordinates), where the columns are located in descending order according to their corresponding eigenvalues.

Figure 3.1 shows two coordinate systems in the auxiliary domain, one for users and the other for items. We represent these two coordinate systems using two matrices as follows,

\[ U_0 = U_1, V_0 = V_2 \]  

(3.3)

where the matrices \( U_1, V_2 \) consist of top \( d \) principle coordinates.

### 3.3.2 Step 2: Coordinate System Adaptation

In Step 2 of the CST algorithm (Figure 3.2), we adapt the principle coordinates \( U_0 \) and \( V_0 \) discovered in the previous step to the target domain.

After obtaining the coordinate systems from the auxiliary data \( R_1, R_2 \), the latent user tastes and item factors are captured by the coordinate systems and can be transferred to the target user-item numerical rating matrix \( R \). In the target domain, we denote the two coordinate systems as \( U, V \) for users and items, respectively, which are also required to be orthonormal according to the definition of coordinate system, that is \( U^T U = I, V^T V = I \).

Further, in order to allow more freedom of rotation and scaling, we adopt the trilinear (or tri-factorization) method [49,131], and allow the rating matrix to be factorized into three parts, one for the user-specific coordinate system \( U \), a second part for the item-specific coordinate system \( V \), and the third part \( B \) to allow rotation and scaling between the two coordinate systems. Note that the problems in [49,131] are quite different from ours, as they require that \( U, V \) are non-negative matrices and \( R \) is a full matrix without missing values.

Finally, we obtain a general framework for coordinate system transfer,

\[
\min_{U,V,B} \mathcal{E}_B(U,V) + \mathcal{R}(U,V|U_0,V_0), \quad \text{s.t. } (U,V) \in D_\perp \tag{3.4}
\]

where \( D_\perp = \{(U,V)|U \in \mathbb{R}^{n \times d}, U^T U = I, V \in \mathbb{R}^{m \times d}, V^T V = I\} \) is the value domain, and \( \mathcal{E}_B(U,V) = \frac{1}{2} ||R - UBV^T||_F^2 \) is the \( B \)-regularized square loss function.
The orthonormal constraints $U^T U = I$ and $V^T V = I$ is a must since otherwise the inner matrix $B$ can be absorbed into the user-specific latent feature matrix $U$ or item-specific latent feature matrix $V$ \cite{49}. Note that $B$ in Eq.(3.4) is different from $\Sigma_1$ (or $\Sigma_2$) in Eq.(3.2), as it is not required to be diagonal, but can be full, and the effect of $B$ is not only scaling as that of $\Sigma_1$ (or $\Sigma_2$), but also rotation when fusing two coordinate systems via $UBV^T$, and hence introduce more interactions between the user-specific coordinate system $U$ and item-specific coordinate system $V$. In our preliminary study, we have tried both diagonal matrix used in \cite{27} and full matrix used in \cite{96}, and find that the full matrix produces much better results. Probably full matrix introduces more correlations or interactions \cite{1}. After we learn $U, V, B$, each missing entry located at $(u, i)$ in $R$ can be predicted as follows,

$$\hat{r}_{ui} = U_u \cdot B V_i^T,$$

where $U_u$ and $V_i$ are the user $u$‘s latent tastes and item $i$‘s latent factors, respectively.

We will describe two instantiations of the proposed general framework of coordinate system transfer in Eq.(3.4), one for biased regularization, and the other for manifold regularization.

**Biased Regularization**

Instead of requiring the two coordinate systems from the auxiliary domain and target domain to be exactly the same,

$$U_j = U_{0,j}, V_j = V_{0,j}, \forall j = 1, \ldots, d,$$

or equivalently $U = U_0, V = V_0$, we relax this requirement and only require them to be similar. We believe that though two domains are related, the latent user tastes and item factors encoded in the coordinate systems $(U_0, V_0)$ in two auxiliary domains can still be a bit different due to the domain specific contexture, i.e. advertisements or promotions on the service provider’s website. Hence, we introduce a relaxed auxiliary enhanced regularization term,

$$R_0(U, V) = \frac{\rho_u}{2} \sum_{j=1}^{d} ||U_j - U_{0,j}||_F^2 + \frac{\rho_v}{2} \sum_{j=1}^{d} ||V_j - V_{0,j}||_F^2$$

$$= \frac{\rho_u}{2} ||U - U_0||_F^2 + \frac{\rho_v}{2} ||V - V_0||_F^2$$
where the tradeoff parameters $\rho_u$ and $\rho_v$ represent confidence on the user-side auxiliary data and item-side auxiliary data, respectively.

We obtain the following optimization problem for the adaptation of the coordinate systems,

$$
\min_{U, V, B} \mathcal{E}_B(U, V) + \mathcal{R}_0(U, V), \quad \text{s.t.} \ (U, V) \in D_\perp.
$$

(3.5)

CST with biased regularization can also be considered as a two-sided extension in matrix form of single-sided model-based transfer learning methods in vector form [97], $\mathcal{E}(w, \ldots) + ||w - w_0||^2_F$, where $w$ and $w_0$ are model parameters for the target data and auxiliary data, respectively. The biased SVM model [97] was proposed not in collaborative filtering but classification problems and achieve knowledge transfer via incorporating the model parameters learned from the auxiliary data as prior knowledge.

**Manifold Regularization**

In this section, we design a different approach to make use of the discovered coordinates systems $U_0$ and $V_0$. The main idea is to constrain users with similar implicit rating behaviors in the auxiliary domain to have similar latent tastes in the target domain of explicit numerical ratings.

The manifold regularization on variables $f_i, i = 1, \ldots, m$ is defined as follows [12],

$$
\sum_{i=1}^{m} \sum_{j=1}^{m} s_{ij} ||f_i - f_j||^2 = \sum_{i=1}^{m} \sum_{j=1}^{m} s_{ij} (f_i^2 + f_j^2 - 2f_i f_j) = 2f^T L f
$$

where $f \in \mathbb{R}^{m \times 1}$ is the variable vector to learn, $L = D - S \in \mathbb{R}^{m \times m}$ is the so-called Laplacian matrix, $S = [s_{ij}]_{m \times m}$ is the similarity matrix defined on any two variable $f_i$ and $f_j$, and $D \in \mathbb{R}^{m \times m}$ is a diagonal matrix with $D_{ii} = \sum_{j=1}^{m} s_{ij}$. The Laplacian regularization will constrain variables of similar entities $i$ and $j$ to be similar, and vice versa.

Specifically, we calculate a Laplacian matrix $L_u \in \mathbb{R}^{n \times n}$ from the user-specific coordinate system $U_0$, and introduce a $\frac{\rho_u}{2} \text{tr}(U^T L_u U)$ to resemble the effect of constraints on users’ latent tastes in the target domain. Similarly, we have another regularization term for the items, $\frac{\rho_v}{2} \text{tr}(V^T L_v V)$.

Thus, we reach the following objective function,

$$
\min_{U, V, B} \mathcal{E}_B(U, V) + \mathcal{R}_L(U, V), \quad \text{s.t.} \ (U, V) \in D_\perp
$$

(3.6)
where $R_L(U, V) = \frac{\rho_u}{2} \text{tr}(U^T \mathcal{L}_u U) + \frac{\rho_v}{2} \text{tr}(V^T \mathcal{L}_v V)$ is the Laplacian based regularization term.

We can see that the difference between the objective function of CST with biased regularization as shown in Eq.(3.5) and that of CST with manifold regularization as shown in Eq.(3.6) comes from the biased regularization term $R_0$ and the Laplacian regularization term $R_L$.

### 3.3.3 Learning the CST

**Learning $U, V$ in CST with Biased Regularization**

First, we denote the objective function in Eq.(3.5) as $f = \mathcal{E}_B(U, V) + R_0(U, V) = \frac{1}{2}||Y \odot (R - UBV^T)||_F^2 + \frac{\rho_u}{2}||U - U_0||_F^2 + \frac{\rho_v}{2}||V - V_0||_F^2$, and thus we have the gradient on $U$ as follows,

$$\frac{\partial f}{\partial U} = (Y \odot (UBV^T - R))VB^T + \rho_u(U - U_0). \quad (3.7)$$

In order to project the gradient $\frac{\partial f}{\partial U}$ to the tangent space at point $U$ of the Grassmann manifold (because of the constraint $U^T U = I$) [56, 27, 95, 96]. We can denote the projected gradient as $\nabla U = \frac{\partial f}{\partial U} + UQ_U$, and obtain $Q_U = -U^T \frac{\partial f}{\partial U}$ via setting $\nabla U = 0$. So, the projected gradient is as follows [56, 27, 95, 96],

$$\nabla U = \frac{\partial f}{\partial U} + UQ_U = \frac{\partial f}{\partial U} - UU^T \frac{\partial f}{\partial U} = (I - UU^T) \frac{\partial f}{\partial U}$$

It is easy to verify that $\nabla U^TU + U^T \nabla U = 0$, which means that the new gradient $\nabla U$ is in the tangent space at point $U$. Similarly, we have $Q_V = -V^T \frac{\partial f}{\partial V}$ and the corresponding projected gradient [56, 27, 95, 96],

$$\nabla V = \frac{\partial f}{\partial V} + VQ_V = \frac{\partial f}{\partial V} - VV^T \frac{\partial f}{\partial V} = (I - VV^T) \frac{\partial f}{\partial V}$$

where $\frac{\partial f}{\partial V} = (Y \odot (UBV^T - R))UB + \rho_v(V - V_0)$ is the gradient before projection.

With the projected gradients $\nabla U$ and $\nabla V$, we can update $U$ and $V$ via standard gradient descent algorithms [56, 27, 95, 96],

$$U \leftarrow U - \gamma \nabla U, \quad V \leftarrow V - \gamma \nabla V \quad (3.8)$$

where the step size $\gamma$ can be determined via line search by checking the decline of the objective value as used in [27, 95, 96]. We propose to estimate the step size $\gamma$ in a
closed form, which is empirically more effective in our experiments. We now show that $\gamma$ can be obtained analytically in the following theorem.

**Theorem 1.** The step size $\gamma$ in Eq.(3.8) can be obtained analytically.

**Proof.** Plugging the update rule for the user-specific coordinate system, $U \leftarrow U - \gamma \nabla U$, in Eq.(3.8) into the objective function in Eq.(3.5), we have,

$$g(\gamma) = \frac{1}{2} ||Y \odot [R - (U - \gamma \nabla U)BV^T]||_F^2 + \frac{\rho_u}{2} ||U - U_0||_F^2 + \frac{\rho_v}{2} ||V - V_0||_F^2$$

$$= \frac{1}{2} ||Y \odot (R - UBV^T) + \gamma Y \odot (\nabla UBV^T)||_F^2 + \frac{\rho_u}{2} ||U - U_0||_F^2 + \frac{\rho_v}{2} ||V - V_0||_F^2$$

Denoting $t_1 = Y \odot (R - UBV^T)$, $t_2 = Y \odot (\nabla UBV^T)$, $\tilde{t}_1 = U - U_0$, $\tilde{t}_2 = \nabla U$, and $c = \frac{\rho_v}{2} ||V - V_0||_F^2$, we have $g(\gamma) = \frac{1}{2} ||t_1 + \gamma t_2||_F^2 + \frac{\rho_u}{2} ||\tilde{t}_1 - \gamma \tilde{t}_2||_F^2 + c$, and the gradient,

$$\frac{\partial g(\gamma)}{\partial \gamma} = \text{tr}(t_1^T t_2) + \gamma \text{tr}(t_2^T t_2) + \rho_u [-\text{tr}(\tilde{t}_1^T \tilde{t}_2) - \gamma \text{tr}(\tilde{t}_2^T \tilde{t}_2)]$$

where $\text{tr}(\cdot)$ denotes the trace function. Then, we obtain the following optimal step size (via setting $\frac{\partial g(\gamma)}{\partial \gamma} = 0$),

$$\gamma_u^* = \frac{-\text{tr}(t_1^T t_2) + \rho_u \text{tr}(\tilde{t}_1^T \tilde{t}_2)}{\text{tr}(t_2^T t_2) + \rho_u \text{tr}(\tilde{t}_2^T \tilde{t}_2)}.$$

Similarly, plugging the update rule $V \leftarrow V - \gamma \nabla V$ into the objective function in Eq.(3.5), we have $g(\gamma) = \frac{1}{2} ||Y \odot (R - UBV^T) + \gamma Y \odot (UB \nabla V^T)||_F^2 + \frac{\rho_u}{2} ||U - U_0||_F^2 + \frac{\rho_v}{2} ||V - V_0||_F^2 - \gamma \nabla V||_F^2$, and the optimal step size,

$$\gamma_v^* = \frac{-\text{tr}(t_1^T t_2) + \rho_v \text{tr}(\tilde{t}_1^T \tilde{t}_2)}{\text{tr}(t_2^T t_2) + \rho_v \text{tr}(\tilde{t}_2^T \tilde{t}_2)}$$

via setting $\frac{\partial g(\gamma)}{\partial \gamma} = 0$, where $t_1 = Y \odot (R - UBV^T)$, $t_2 = Y \odot (UB \nabla V^T)$, $\tilde{t}_1 = V - V_0$, and $\tilde{t}_2 = \nabla V$. $\square$
Learning $U, V$ in CST with Manifold Regularization

Let $f = \mathcal{E}_B(U, V) + \mathcal{R}_C(U, V) = \frac{1}{2}\|Y \odot (R - UBV^T)\|_F^2 + \frac{\rho_u}{2}\text{tr}(U^T L_u U) + \frac{\rho_v}{2}\text{tr}(V^T L_v V)$, we have the gradients on $U$ as follows,

$$\frac{\partial f}{\partial U} = (Y \odot (UBV^T - R))VB^T + \rho_u L_u U$$

(3.9)

So, we can update $U$ as follows,

$$U \leftarrow U - \gamma \nabla U, \quad \nabla U = (I - UU^T)\frac{\partial f}{\partial U}$$

(3.10)

Plugging the update rule in Eq.(3.10) into the objective function in Eq.(3.6), we have,

$$g(\gamma) = \frac{1}{2}\|Y \odot [R - (U - \gamma \nabla U)BV^T]\|_F^2 + \frac{\rho_u}{2}\text{tr}[(U - \gamma \nabla U)^T L_u (U - \gamma \nabla U)] + \frac{\rho_v}{2}\text{tr}(V^T L_v V)$$

$$= \frac{1}{2}\|Y \odot (R - UBV^T) + \gamma Y \odot (\nabla UBV^T)\|_F^2 + \frac{\rho_u}{2}[-\gamma\text{tr}(U^T L_u \nabla U) - \gamma\text{tr}(\nabla U^T L_u U) + \gamma^2\text{tr}(\nabla U^T L_u \nabla U)] + C$$

where $C = \frac{\rho_u}{2}\text{tr}(U^T L_u U) + \frac{\rho_v}{2}\text{tr}(V^T L_v V)$ is a constant and independent of the variable $\gamma$. Denoting $t_1 = Y \odot (R - UBV^T)$, $t_2 = Y \odot (\nabla UBV^T)$, $t_3 = \text{tr}(U^T L_u \nabla U)$, $t_4 = \text{tr}(\nabla U^T L_u U)$, $t_5 = \text{tr}(\nabla U^T L_u \nabla U)$, we have $g(\gamma) = \frac{1}{2}\|t_1 + \gamma t_2\|_F^2 + \frac{1}{2}(-\gamma t_3 - \gamma t_4 + \gamma^2 t_5) + C$, and the gradient,

$$\frac{\partial g(\gamma)}{\partial \gamma} = \text{tr}(t_1^T t_2) + \gamma\text{tr}(t_2^T t_2) + \frac{\rho_u}{2}[-t_3 - t_4 + 2\gamma t_5],$$

from which we obtain $\gamma_u^* = -\frac{\text{tr}(t_1^T t_2) + \rho_u(t_3 + t_4)/2}{\text{tr}(t_2^T t_2) + \rho_u t_5}$ via setting $\frac{\partial g(\gamma)}{\partial \gamma} = 0$.

Similarly, we can have an update rule for $V$ as follows,

$$V \leftarrow V - \gamma \nabla V$$

(3.11)

where $\nabla V = (I - VV^T) \left[ (Y \odot (UBV^T - R))UB + \rho_v L_v V \right]$ and the optimal step size $\gamma_v^* = -\frac{\text{tr}(t_1^T t_2) + \rho_v(t_3 + t_4)/2}{\text{tr}(t_2^T t_2) + \rho_v t_5}$, with $t_1 = Y \odot (R - UBV^T)$, $t_2 = Y \odot (UBV^T)$, $t_3 = \text{tr}(V^T L_v \nabla V)$, $t_4 = \text{tr}(\nabla V^T L_v V)$, $t_5 = \text{tr}(\nabla V^T L_v \nabla V)$.

The difference between our approach of learning $U, V$ and that of [56, 27, 95, 96] can be identified from two aspects. First, the gradients as shown in Eq.(3.7) and in
Eq. (3.9) are different from that of [56, 27, 95, 96], since we have additional terms of \( \rho_u(U - U_0) \) and \( \rho_uC_uU \) representing the knowledge extracted from the auxiliary data. Second, the way to determine a suitable step size in Eq. (3.8), Eq. (3.10) and Eq. (3.11) is different since we estimate the \( \gamma \) analytically instead of line search as used in [27, 95, 96].

**Learning B in CST**

Given \( U, V \), the optimization problem in Eq. (3.5) reduces to the following simplified problem,

\[
\min_B \frac{1}{2} ||Y \odot (R - UBV^T)||_F^2 + \frac{\beta}{2} ||B||_F^2,
\]

which is the same as that of [95, 96, 1], and we use the same closed-form solution [95, 96]. Specifically, letting \( f = \frac{1}{2} ||Y \odot (R - UBV^T)||_F^2 + \frac{\beta}{2} ||B||_F^2 \), we have the gradient of \( B \) as follows [27],

\[
\frac{\partial f}{\partial B} = -U^T (Y \odot (R - UBV^T)) V + \beta B
\]

Setting \( \frac{\partial f}{\partial B} = 0 \), we have

\[
U_j^T (Y \odot R)V_k = U_j^T [Y \odot (UBV^T)] V_k + \beta B_{jk}, \ \forall j, k
\]

(3.12)

with,

\[
U_j^T [Y \odot (UBV^T)] V_k = \langle Y \odot (UBV^T), U_j V_k^T \rangle
\]

\[
= \langle UBV^T, Y \odot (U_j V_k^T) \rangle
\]

\[
= \langle B, U^T [Y \odot (U_j V_k^T)] V \rangle
\]

\[
= vec \left(U^T [Y \odot (U_j V_k^T)] V \right)^T vec(B)
\]

(3.13)

where \( \langle X, Y \rangle = \sum_{jk} x_{jk} y_{jk} \) is the inner product of two matrices, and \( vec(X) = [\cdots X^T_k \cdots]^T \) is a big vector concatenated from columns \( X_k \). Combining Eq.(3.12) and Eq.(3.13), we have,

\[
U_j^T (Y \odot R)V_k = vec \left(U^T [Y \odot (U_j V_k^T)] V \right)^T vec(B) + \beta B_{jk}.
\]

Finally, the inner matrix \( B \) can be obtained in a closed form [95, 96],

\[
vec(B) = (A^T + \beta I)^{-1} vec(Z),
\]

(3.14)
from the following linear system \([95, 96]\),

\[
vec(Z) = A^T vec(B) + \beta vec(B),
\]

where \(A \in \mathbb{R}^{d^2 \times d^2}\) with \(A_{k,j} = vec\left(U^T (Y \odot (U_j V_k^T)) V\right), \ell = (k - 1) \times d + j, k, j = 1, \ldots, d,\) and \(Z \in \mathbb{R}^{d \times d}\) with \(z_{jk} = U^T_j (Y \odot R)V_k\).

Note that we have tried both the method used in \([27]\) to estimate a diagonal inner matrix and the method used in \([95, 96]\) to estimate a full inner matrix, and find that the method from \([95, 96]\) produces much better results. Probably the full matrix in \([95, 96]\) introduces more correlations or interactions between user-specific coordinate system \(U\) and item-specific coordinate system \(V\), since \(B\) can be considered as a linear operator \([1]\) when we consider the fixed \(U\) and \(V\) as content information.

Finally, the optimization problem can be solved efficiently via an alternating approach, by (a) fixing \(U, V\), where the inner matrix \(B\) can then be estimated analytically, and (b) fixing \(B\), where \(U, V\) can be alternatively solved on the Grassman manifold through a projected gradient descent method. Each of the above sub-steps of updating \(B, U\) and \(V\) will monotonically decreases the objective function in Eq. (3.4), and hence ensures convergence to local minimum. The complete transfer learning solution is given in Figure 3.2.

The time complexity of CST with biased regularization is \(O(Kpd^5 + Kd^6)\), and the time complexity of CST with manifold regularization is \(O(Kpd^3 + Kd^6 + K\tilde{p}d)\), where \(K\) is the iteration number, \(p(p > n, m)\) is the number of non-zero entries in the rating matrix \(R\), \(\tilde{p}\) is the number of non-zero entries in the Laplacian matrices \(L_u\) and \(L_v\), and \(d\) is the number of latent dimensions. Note that \(d, K\) are usually quite small, i.e. \(d < 20, K < 100\) in our experiments. \(O(d^6)\) comes from the time complexity of a matrix inverse with size of \(d^2 \times d^2\), which is used to estimate the variable \(B \in \mathbb{R}^{d \times d}\). Note that \(O(d^6)\) for matrix inverse (with size of \(d^2 \times d^2\)) is the worst time complexity, which can be improved via various techniques, e.g. stochastic sampling or distributed computing. We only report the worst case without any optimization, since our focus is on knowledge transfer in collaborative filtering. We will study the efficiency issue as our future work.
**Input:** The target user-item numerical rating matrix $R$, the user-side auxiliary user-item implicit rating matrix $R_1$, the item-side auxiliary user-item implicit rating matrix $R_2$.

**Output:** The user-specific coordinate system $U$, the item-specific coordinate system $V$, the inner matrix $B$.

**Step 1. Coordinate system construction**

**Step 1.1. Impute zeros** for missing ratings in $R_1$, $R_2$ to obtain two full matrices $\hat{R}_1$, $\hat{R}_2$ as shown in Eq.(3.1);

**Step 1.2. Apply low-rank SVD** [16] on $\hat{R}_1$ and $\hat{R}_2$ to obtain two principle coordinate systems $U_0 = U_1$, $V_0 = V_2$ as shown in Eq.(3.2);

**Step 1.3. Initialize the target coordinate systems** with $U = U_0$, $V = V_0$ as shown in Eq.(3.3).

**Step 2. Coordinate system adaptation**

**repeat**

**Step 2.1.** Learn $U$, $V$

**repeat**

**Step 2.1.1.** Fix $B$ and $V$, update $U$ as shown in Eq.(3.8) or Eq.(3.10);

**Step 2.1.2.** Fix $B$ and $U$, update $V$ as shown in Eq.(3.8) or Eq.(3.11);

**until** Convergence

**Step 2.2.** Fix $U$ and $V$, estimate $B$ as shown in Eq.(3.14).

**until** Convergence

Figure 3.2: The algorithm of coordinate system transfer (CST).

### 3.3.4 Extensions of CST

In this section, we further generalize CST from 2-order to $N$-order, which has various potential applications for volume data processing using high-order SVD [111, 101].

We first take high-order CST with biased regularization for example, and then illustrate high-order CST with manifold regularization. Consider a high-order ($N$-order) tensor $R$, the corresponding weighting tensor $Y$, the $N$-order core tensor $B$, and the factorized orthonormal matrices $A^{(n)}$, $n = 1, 2, \ldots, N$. Given the coordinate systems, $A_0^{(n)}$, $n = 1, 2, \ldots, N$, obtained from auxiliary data sources, we reach the following optimization problem, which is a high-order generalization of Eq.(3.5),

$$
\min_{A^{(1)}, \ldots, A^{(N)}, B} \|Y \odot (R - [B; A^{(1)} \ldots A^{(N)}])\|_F^2 + \sum_{n=1}^N \frac{\rho_n}{2} \|A^{(n)} - A_0^{(n)}\|_F^2
$$

s.t. $A^{(n)T} A^{(n)} = I$, $n = 1, \ldots, N$

where $[B; A^{(1)} \ldots A^{(N)}]$ is the Tucker operator [100]. We can estimate $A^{(n)}$ and $B$ in a similar way as that of 2-order CST via alternative gradient descent approach:
(a) given $A^{(i)}$, $i = 1, 2, \ldots, N, i \neq n$ and $B$, we can estimate $A^{(n)}$ as follows,

$$\min_{A^{(n)}} ||Y^{(n)} \odot (R^{(n)} - A^{(n)}B^{(n)}A^{(\neq n)T})||^2_F + \frac{\rho_n}{2} ||A^{(n)} - A_0^{(n)}||^2_F$$

s.t. $A^{(n)T}A^{(n)} = I$

where $A^{(\neq n)} = A^{(N)} \odot A^{(n+1)} \odot A^{(n-1)} \cdots \odot A^{(1)}, R^{(n)}, Y^{(n)}$ and $B^{(n)}$ are mode $n$ unfolding [111] of $R$, $Y$ and $B$, respectively. It’s obvious that $A^{(n)}$ can be estimated exactly the same as that of $U$ in Eq.(3.5), as now we have reduced the $N$-dimension to 2-dimension via tensor unfolding or matricization [100].

(b) given $A^{(i)}$, $i = 1, 2, \ldots, N$, we can estimate $B$ as follows,

$$\min_{B^{(1)}} ||Y^{(1)} \odot (R^{(1)} - A^{(1)}B^{(1)}A^{(\neq 1)T})||^2_F$$

where we can estimate $B^{(1)}$ exactly the same as that of $B$ in Eq.(3.5), and finally obtain $B$ via inverse unfolding from $B^{(1)}$.

For high-order CST with manifold regularization, we only need to replace the biased regularization term $\frac{\rho_n}{2} ||A^{(n)} - A_0^{(n)}||^2_F$ with the Laplacian based regularization term $\text{tr}(A^{(n)T}L^{(n)}A^{(n)})$.

### 3.4 Experimental Results

Our experiments are designed to verify the following hypotheses,

1. we believe that the proposed transfer learning methods, CST with biased regularization and CST with manifold regularization, are better than other baseline algorithms, since CST is designed to achieve knowledge transfer from auxiliary implicit feedbacks to alleviate the sparsity problem in the target numerical rating matrix;

2. we believe that the coordinate systems can be transferred, since they contain the knowledge of users’ latent taste and items’ latent factors;

3. we believe that two-sided transfer of both $U_0$ and $V_0$ is more effective than one-sided transfer of either $U_0$ or $V_0$.  

49
3.4.1 Data Sets and Evaluation Metrics

We evaluate the proposed method using two movie rating data sets Netflix\(^1\) and MovieLens\(^2\). The Netflix rating data contains more than \(10^8\) ratings with values in \(\{1, 2, 3, 4, 5\}\), which are given by more than \(4.8 \times 10^5\) users on around \(1.8 \times 10^4\) movies. The MovieLens rating data contains more than \(10^7\) ratings with values in \(\{1, 2, 3, 4, 5\}\), which are given by more than \(7.1 \times 10^4\) users on around \(1.1 \times 10^4\) movies. The data set used in the experiments is constructed as follows,

1. we first identify 5,000 movies appearing both in MovieLens and Netflix via movie title;

2. we then extract a dense item side auxiliary data \(R_2\) of size \(5,000 \times 5,000\) from the MovieLens;

3. we then extract a \(10,000 \times 10,000\) dense rating matrix \(R\) from the Netflix data (10,000 most frequent users and another 5,000 most popular items), and take the sub-matrices \(R = R_{1:5,000;1:5,000}\) as the target rating matrix (\(R\) and \(R_2\) share only common items but no users), and \(R_1 = R_{1:5,000;5001:10,000}\) as the user side auxiliary data (\(R\) and \(R_1\) share only common users but not common items);

4. finally, to simulate implicit feedbacks of auxiliary data, we relabel “1, 2, 3, 4, 5” ratings in \(R_1, R_2\) as positive feedbacks “1”.

In all of our experiments, the target domain rating set from \(R\) is randomly split into training and test sets, \(T_R, T_E\), with 50% ratings, respectively. \(T_R, T_E \subset \{(u, i, r_{ui}) \in \mathbb{N} \times \mathbb{N} \times \{1, 2, 3, 4, 5\}|1 \leq u \leq n, 1 \leq i \leq m\}\). \(T_E\) is kept unchanged, while different (average) number of observed ratings for each user, 10, 20, 30, 40, are randomly picked from \(T_R\) for training, with different sparsity levels of 0.2%, 0.4%, 0.6%, 0.8% correspondingly. The final data set used in the experiments is summarized in Table 3.2.

\(^1\) http://www.netflix.com

\(^2\) http://www.grouplens.org/node/73
Table 3.2: Description of a subset of Netflix data \((n = 5,000, m = 5,000)\) and a subset of MovieLens data \((n = 5,000, m = 5,000)\) used in the experiments.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Form</th>
<th>Size</th>
<th>Sparsity</th>
</tr>
</thead>
<tbody>
<tr>
<td>target (training)</td>
<td>{1, 2, 3, 4, 5, ?}</td>
<td>5,000×5,000</td>
<td>&lt;1.0%</td>
</tr>
<tr>
<td>target (test)</td>
<td>{1, 2, 3, 4, 5, ?}</td>
<td>5,000×5,000</td>
<td>11.3%</td>
</tr>
<tr>
<td>auxiliary (user side)</td>
<td>{1, ?}</td>
<td>5,000×5,000</td>
<td>10%</td>
</tr>
<tr>
<td>auxiliary (item side)</td>
<td>{1, ?}</td>
<td>5,000×5,000</td>
<td>9.5%</td>
</tr>
</tbody>
</table>

We adopt two evaluation metrics: Mean Absolute Error (MAE) and Root Mean Square Error (RMSE),

\[
MAE = \frac{1}{|T_E|} \sum_{(u,i,r_{ui})\in T_E} |r_{ui} - \hat{r}_{ui}| \\
RMSE = \sqrt{\frac{1}{|T_E|} \sum_{(u,i,r_{ui})\in T_E} (r_{ui} - \hat{r}_{ui})^2}
\]

where \(r_{ui}\) and \(\hat{r}_{ui}\) are the true and predicted ratings, respectively, and \(|T_E|\) is the number of test ratings. In all experiments, we run 3 random trials when generating the required number of observed ratings for each user from the target training rating set \(T_R\), and averaged results are reported.

### 3.4.2 Baselines and Parameter Settings

We compare our CST method with five non-transfer learning methods: the average filling method (AF), Pearson correlation coefficient (PCC) [172], PMF [177], singular value decomposition (SVD) [181], and OptSpace [96]. Note, the codebook in CBT [115] and RMGM [116] constructed from a matrix of implicit feedbacks, \(\{1, ?\}\), is always a full matrix of 1s only, and does not reflect any cluster-level rating patterns, hence both CBT and RMGM are not applicable to our problem. SoRec [138] and CMF [193] are designed for transferring knowledge from explicit data instead of implicit feedbacks, and thus are not applicable directly in our problem. CST reduces to the spectral matrix completion method, OptSpace [96], when no auxiliary data exist, and thus we also report the performance of OptSpace [96]. Since the algorithm of learning \(U\) and \(V\) in CST is different from that of [27, 96], we also report the performance of “CST (null)”, which does not make use of any auxiliary data but factorizes the target rating matrix only.
To study the effect of single-sided transfer and two-sided transfer, we also report results of CST for user-side transfer and item-side transfer, separately.

We study the following six average filling (AF) methods,

\[
\hat{r}_{ui} = \bar{r}_u, \\
\hat{r}_{ui} = \bar{r}_i, \\
\hat{r}_{ui} = (\bar{r}_u + \bar{r}_i)/2, \\
\hat{r}_{ui} = b_u + \bar{r}_i, \\
\hat{r}_{ui} = \bar{r}_u + b_i, \\
\hat{r}_{ui} = \bar{r} + b_u + b_i,
\]

where \(\bar{r}_u = \frac{\sum y_{ui} r_{ui}}{\sum y_{ui}}\) is the average rating of user \(u\), \(\bar{r}_i = \frac{\sum y_{ui} r_{ui}}{\sum y_{ui}}\) is the average rating of item \(i\), \(b_u = \frac{\sum y_{ui} (r_{ui} - \bar{r}_i)}{\sum y_{ui}}\) is the bias of user \(u\), \(b_i = \frac{\sum y_{ui} (r_{ui} - \bar{r}_u)}{\sum y_{ui}}\) is the bias of item \(i\), and \(\bar{r} = \frac{\sum y_{ui} r_{ui}}{\sum y_{ui}}\) is the global average rating. We use \(\hat{r}_{ui} = \bar{r} + b_u + b_i\) as it performs best in our experiments. To compare with commonly used average filling methods, we also report the results of \(\hat{r}_{ui} = \bar{r}_u\) and \(\hat{r}_{ui} = \bar{r}_i\).

For SVD [181], we adopt the approach of 5-star numerical rating predictions, which are reported as the best one in [181]. Specifically, we convert the original rating matrix \(R\) to \(\tilde{R}\) as follows [181],

\[
r_{ui} \rightarrow \tilde{r}_{ui} = \begin{cases} 
  r_{ui} - \bar{r}_u, & \text{if } y_{ui} = 1 \text{ (rated)} \\
  \bar{r}_i - \bar{r}_u, & \text{if } y_{ui} = 0 \text{ (not rated)}
\end{cases}
\]

where \(\bar{r}_u\) is the user \(u\)’s average rating and \(\bar{r}_i\) is the item \(i\)’s average rating, same as that used in the aforementioned average filling methods; and then we apply SVD [16, 181] on the matrix \(\tilde{R}\), \(\tilde{R} = U \Sigma V^T\); and (iii) finally, the rating of user \(u\) on item \(i\) can be predicted as follows [181],

\[
\hat{r}_{ui} = \bar{r}_u + U_u \Sigma V_i^T
\]

where the average rating \(\bar{r}_u\) is added to the prediction rule.

For PCC, since the data matrices are sparse, we use the whole set of neighboring users in the prediction rule. For PMF [177], singular value decomposition (SVD) [181], OptSpace [96], CST, we first fix \(d = 10\) for easy comparison (see Table 3.3) and then study the effect of different latent dimensions \(d \in \{5, 10, 15\}\) (see Figure 3.3); for PMF,
different tradeoff parameters \( \{0.01, 0.1, 1\} \) are tried; for CST, the tradeoff parameters are set as \( \rho_u/n = \rho_v/m = 1 \). For CST with manifold regularization, we use heat kernel \( \exp(-||U_{0u} - U_{0wu}||^2/2) \) as the similarity measurement, where \( U_{0u} \) and \( U_{0wu} \) denote user \( u \) and user \( w \)’s latent representation from the auxiliary coordinate system \( U_0 \), and we set the number nearest neighbors as 10 when we construct the Laplacian matrix.

### 3.4.3 Summary of the Experimental Results

We randomly sample \( n \) ratings (one rating per user on average) from the training data \( \mathbf{R} \) and use them as the validation set to determine the tradeoff parameters and convergence condition (the number of iterations to convergence) for PMF, OptSpace and CST. For
AF, PCC and SVD, both the training set and validation set are combined as one set of training data.

**Overall Results**

The results on test data (unavailable during training) are reported in Table 3.3 ($d = 10$). We can make the following observations:

1. The proposed *transfer learning* methods, CST-biased and CST-manifold, perform significantly better than all other baselines in almost all sparsity levels. For sparse data, the smoothing method AF always performs better than PMF, SVD and OptSpace, while another two variants of AF (user) and AF (item) are much worse; however, the proposed transfer learning methods, CST-biased and CST-manifold, beat AF in almost all cases.

2. The *coordinates systems* from the auxiliary data can be transferred. CST is always better than CST (null), which clearly shows the usefulness of the coordinate systems; We also note that CST (null) is slightly better than OptSpace, which shows the advantages of our closed-form solution for searching the step size $\gamma$ in learning $U$ and $V$.

3. For the single-sided transfer learning methods, CST-biased (item), CST-biased (user), CST-manifold (item), CST-manifold (user), they beat the non-transfer learning baselines, PMF, SVD and OptSpace, in all sparsity levels, which demonstrates the usefulness of transfer learning methods for sparse data; however, we can see that the *two-sided transfer* learning methods, CST-biased and CST-manifold, are always better than those single-sided transfer learning approaches.

**Results with and without Auxiliary Data**

Note that when learning is conducted in the target *user-item* rating matrix only, the CST method becomes equivalent to the OptSpace model [95, 96], which considers no auxiliary domain information, neither initialization nor regularization. To gain a deeper understanding of CST, and more carefully assess the benefit from the auxiliary domain, we compared the performance of CST-biased, CST-manifold and OptSpace at different data sparsity levels when the parameter $d$, the number of latent dimensions, is increased from 5 to 15. The results are shown in Figure 3.3. We can see that:
1. the performance of OptSpace consistently deteriorates as $d$ increases, which is due to that more flexible models (trilinear vs. bilinear) are more likely to suffer from overfitting given sparse data;

2. in contrast to OptSpace, CST consistently improve as $d$ increases which demonstrates how the auxiliary domain knowledge based initialization and regularization techniques can help avoid overfitting even for highly flexible models. We also note that the performance of CST with manifold regularization is much better than CST with biased regularization when the target data is very sparse, and similar when the target data becomes denser, which comes from the different regularization effects.

**Results with Different Sparsity Levels of Auxiliary Data**

We also study the effect of varying sparsity of auxiliary data. The previous results use sparsity (10%, 9.5%) of the auxiliary data of $R_1$ and $R_2$, respectively, which is shown in Table 3.2. We set the sparsity to (5.0%, 5.0%) and (2.0%, 2.0%), and study the effect of how different sparsity of auxiliary data will affect the prediction performance in the target task. Similarly, for the target data, different number of observed ratings for each user, 10, 20, 30, 40, are randomly picked from $T_R$ for training, with different sparsity levels of 0.2%, 0.4%, 0.6%, 0.8% correspondingly. The results are shown in Figure 3.4. We can see that the prediction performance of CST degrade when low density of auxiliary data is used, which intuitively makes sense, since the quality and quantity of the knowledge transferred from auxiliary data is reduced.

**Results with Different Regularization from the Auxiliary Data**

We also study the effect of different regularization from the auxiliary data in more detail. The prediction performance on the target test data with 0.2% sparsity (10 observed ratings for each user) and $d = 10$ is shown in Figure 3.5. We can see that the performance of CST is significantly better than that of CST without regularization in all cases, which demonstrates the effect of regularization from the auxiliary data. Comparing to the performance of CST-biased, CST-manifold is much better for sparse target data, from which we can see that the Laplacian based regularization term in CST-manifold is more effective for sparse data.
Figure 3.3: Comparison of CST-biased, CST-manifold and OptSpace at different sparsity levels with different latent dimension numbers (auxiliary sparsity: 10%, 9.5%).
Figure 3.4: Prediction performance of CST-biased and CST-manifold at different sparsity levels of auxiliary and target data ($d = 10$).

Figure 3.5: Prediction performance of CST with and without the regularization term (auxiliary sparsity: 10%, 9.5%, target sparsity: 0.2%).

### 3.5 Discussions

#### 3.5.1 Transfer Learning in Collaborative Filtering

Probabilistic matrix factorization (PMF) [177] or latent factorization model (LFM) [13] is a widely used method in collaborative filtering, which seeks an appropriate low-rank approximation of the rating matrix $R$ with two latent feature matrices, one for users
and one for items. For any missing entry in $R$, it can be predicted by the production of the corresponding two latent feature vectors.

Social recommendation (SoRec) [138] and collective matrix factorization (CMF) [193] are multi-task learning (MTL) [33] versions of PMF [177] or LFM [13], which jointly factorize two matrices with correspondences between rows or columns while sharing same latent features of matched rows or columns in different matrices. Note, there are at least three differences compared to the CST method. First, SoRec [138] or CMF [193] is an MTL style algorithm, which does not distinguish auxiliary domain from target domain, whereas CST is an adaptation style algorithm, which focuses on improving performance in the target domain by transferring knowledge from but not to the auxiliary domain. Hence CST is more efficient especially when the auxiliary data is dense, and more secure for privacy considerations. Second, SoRec [138] or CMF [193] is a bi-factorization (or bilinear) method (i.e. $R = UV^T$), while CST is a tri-factorization (or trilinear) method [49] (i.e., $R = UBV^T$). Third, there is no constraints on the latent feature matrix in SoRec [138] or CMF [193], while the orthonormal constraints in CST, $U^TU = I$ and $V^TV = I$, represent some semantic meanings of users’ latent taste and items’ latent factors.

Codebook transfer (CBT) [115] is a recently developed transfer learning method for collaborative filtering, which contains two steps of codebook construction and codebook expansion, and achieves knowledge transfer with the assumption that both auxiliary and target data share the cluster-level rating patterns (codebook). Rating-matrix generative model (RMGM) [116] is derived and extended from the FMM generative model [190], and we can consider RMGM [116] as an MTL version of CBT [115] with the same assumption. Note, both CBT [115] and RMGM [116] are limited to explicit rating matrices only, and can not achieve knowledge transfer from an implicit rating matrix with values of \{1, ?\} to an explicit one with values of \{1, 2, 3, 4, 5, ?\}, as it requires two rating matrices to share the same cluster-level rating patterns. Also, CBT [115] and RMGM [116] can neither make use of user side nor item side existing correspondences, and only take a general explicit rating matrix as its auxiliary input. Hence, both CBT [115] and RMGM [116] are not applicable to the problem studied in this chapter.

We summarize all methods in Table 3.4 from the perspective of two fundamental questions in transfer learning [157], what to transfer and how to transfer, respectively.
Table 3.4: Summary of CST and other transfer learning methods in collaborative filtering.

<table>
<thead>
<tr>
<th>Perspectives</th>
<th>Methods</th>
<th>CST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge (What to transfer)</td>
<td>CBT [115]</td>
<td>Latent tastes/factors (Two wings)</td>
</tr>
<tr>
<td></td>
<td>RMGM [116]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CMF [193], SoRec [138]</td>
<td></td>
</tr>
<tr>
<td>Algorithm style (How to transfer)</td>
<td>Adaptation</td>
<td>Adaptation</td>
</tr>
<tr>
<td></td>
<td>MTL</td>
<td></td>
</tr>
<tr>
<td>Auxiliary data/feedback</td>
<td>Explicit</td>
<td>Implicit</td>
</tr>
<tr>
<td></td>
<td>Explicit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Implicit</td>
<td></td>
</tr>
</tbody>
</table>

3.5.2 Manifold Learning in Matrix Factorization

Matrix factorization with additional Laplacian enhanced regularization has recently been proposed [145, 137, 231, 134].

The MACF (manifold alignment collaborative filtering) algorithm [145] studies the problem when users in two CF systems are only partially aligned, which is achieved in the following four steps. First, it requires the aligned users from two CF systems to share the same latent feature vectors. Second, it constrains similar users in the same system to be similar in latent feature space via manifold embedding [89]. Third, it maps every user in the target CF system to a weighted combination of $k$ users in the auxiliary CF system in the latent space. Finally, it obtains a better similarity measure for users in the target CF system. Note that the first two steps are actually constrained LLE (locally linear embedding) [183] for low-rank embedding, and the last two steps are for memory-based rating prediction in the target CF system.

The RES (recommendation with trust) model [137] generalizes the PMF model [177] by introducing two additional regularization terms from trusted and distrusted users,

$$
\lambda^+ \sum_{u=1}^{n} \sum_{w \in T_u^+} s_{1uw} \left\| U_u - U_w \right\|_F^2 - \lambda^- \sum_{u=1}^{n} \sum_{w \in T_u^-} s_{1uw} \left\| U_u - U_w \right\|_F^2
$$

(3.15)

where $T_u^+$ and $T_u^-$ are trusted and distrusted users of user $u$ (not including user $u$ himself), and thus embedding both trust and distrust connections via constraining trusted users and distrusted users to be similar and dissimilar in latent space, respectively. Note that distrust relationship may cause non-PSD Laplacian matrix. We thus have two Laplacian regularization terms, one for trusted users and one for distrusted users.
There are some tagging data associated with the user-item rating matrix, e.g. the Movei lens (http://movielens.umn.edu/) data used in [231]. The tagiCoFi (tag informed collaborative filtering) model [231] first obtains a user-user similarity matrix from social tagging data and then introduces an additional regularization term to the PMF model [177],

\[
\sum_{u=1}^{n} \sum_{w=1}^{n} s_{1uw} \| U_u - U_w \|_F^2,
\]

where \( s_{1uw} \) is the similarity between users \( u \) and \( w \), thus transfer the nearest neighbors’ taste via constraining the user-specific features to be similar in the latent space. The tagiCoFi model [231] is a weighted version of the RRMF (relation regularized matrix factorization) model [119] by considering missing values, which is very important in recommender systems. We thus have one Laplacian regularization for user-specific latent variables, with user-user similarities calculated from social tagging information.

The SptMF (spatially regularized matrix factorization) model [134] introduces two additional regularization terms to the PMF model [177],

\[
\sum_{u=1}^{n} \sum_{w=1}^{n} s_{1uw} \| U_u - U_w \|_F^2 + \sum_{i=1}^{m} \sum_{j=1}^{m} s_{2ij} \| V_i - V_j \|_F^2.
\]

where we have two Laplacian regularization terms, one for user-specific latent features, and one for item-specific latent features.

The MIMO (multi-type interrelated objects embedding) model [68] combines the user-item rating matrix \( R \), user-tag preference matrix \( R_1 \), item-tag link matrix \( R_2 \) and item-item similarity matrices \( S_2 \) in a unified regularization framework,

\[
\sum_{u=1}^{n} \sum_{i=1}^{m} r_{ui} \| U_u - V_i \|_F^2 + \sum_{u=1}^{n} \sum_{t=1}^{r} r_{1u} \| U_u - T_j \|_F^2 + \sum_{i=1}^{m} \sum_{t=1}^{r} r_{2ij} \| V_i - T_j \|_F^2 + \sum_{i=1}^{m} \sum_{j=1}^{m} s_{2ij} \| V_i - V_j \|_F^2
\]

where \( R_1 \in \mathbb{R}^{n \times t} \) and \( R_2 \in \mathbb{R}^{m \times t} \) are obtained via user-item-tag tensor data aggregation on the item and user dimension, respectively, \( R \in \{0, 1\}^{n \times m} \) is constructed as whether user \( u \) has assigned a tag to item \( i \), and \( S_2 \) is constructed from cosine similarities of the items’ content information. The MIMO model can be considered as an
extension of manifold learning without considering missing value, since each of the
above four terms can be considered as a manifold regularization term.

The difference of our proposed method CST-manifold and other methods using
Laplacian based regularization term can be identified from several aspects. First, we
transfer coordinate systems with both initialization and regularization instead of regu-
larization only, since we believe that coordinate systems contain domain-independent
knowledge. Second, we introduce orthonormal constraints on latent variables, U and
V, which represent users’ latent taste and items’ latent factor, respectively.

3.5.3 Tensor Factorization

The generalized high-order CST can also be studied from the perspective of vari-
ous tensor factorization methods, and we give a brief comparison in Table 3.5. Inter-
estingly, we can see two families of tensor factorization methods: CANDECOMP-
PARAFAC (CP) [31, 76], RTF [168], CP-WOPT [2], PITF [170], UCLA [232] and
LOTD [28] are in the family without orthonormal constraints on the factorized ma-
trices, while HOSVD [111], HOOI [112], SP/MP [205] and high-order CST is in the
other family. As far as we know, high-order CST is the first tensor factorization method
considering both missing value and auxiliary data sources, which are very important
in various applications [2, 157].

Table 3.5: A brief comparison of high-order CST with other tensor factorization meth-
ods. \( D_{\perp} \) denotes orthonormal constraints on the factorized matrices.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Perspectives</th>
<th>Missing value</th>
<th>Auxiliary data</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o ( D_{\perp} )</td>
<td>CANDECOMP/PARAFAC (CP) [31, 76]</td>
<td>( \times )</td>
<td>( \times )</td>
</tr>
<tr>
<td></td>
<td>RTF [168]</td>
<td>( \checkmark )</td>
<td>( \times )</td>
</tr>
<tr>
<td></td>
<td>CP-WOPT [2]</td>
<td>( \checkmark )</td>
<td>( \times )</td>
</tr>
<tr>
<td></td>
<td>PITF [170]</td>
<td>( \checkmark )</td>
<td>( \times )</td>
</tr>
<tr>
<td></td>
<td>UCLA [232]</td>
<td>( \times )</td>
<td>( \checkmark )</td>
</tr>
<tr>
<td></td>
<td>LOTD [28]</td>
<td>( \checkmark )</td>
<td>( \times )</td>
</tr>
<tr>
<td>w/ ( D_{\perp} )</td>
<td>HOSVD [111]</td>
<td>( \times )</td>
<td>( \times )</td>
</tr>
<tr>
<td></td>
<td>HOOI [112]</td>
<td>( \times )</td>
<td>( \times )</td>
</tr>
<tr>
<td></td>
<td>SP/MP [205]</td>
<td>( \times )</td>
<td>( \times )</td>
</tr>
<tr>
<td></td>
<td><strong>High-order CST</strong></td>
<td>( \checkmark )</td>
<td>( \checkmark )</td>
</tr>
</tbody>
</table>
3.6 Summary

In this chapter, we presented a novel transfer learning framework called coordinate system transfer (CST) for alleviating the data sparsity problem in collaborative filtering. Our method first finds a subspace where coordinate systems are used for knowledge transfer, then uses the knowledge to adapt to the target domain data. The novelty of our algorithm includes using both the user-side and item-side information in an adaptive way. Experimental results show that CST performs significantly better than several state-of-the-art methods at various sparsity levels. Our experimental study clearly demonstrates the usefulness of transferring two coordinate systems from the auxiliary data (what to transfer), and the effectiveness of incorporating two-sided auxiliary knowledge via a regularized tri-factorization method, thus addressing the how to transfer question.

We also generalize CST from 2-order to $N$-order and obtain high-order CST, which are the first tensor factorization method considering both missing value and auxiliary data sources.
CHAPTER 4

TRANSFER LEARNING IN COLLABORATIVE FILTERING WITH FRONTAL-SIDE BINARY RATINGS

A major challenge for collaborative filtering (CF) techniques in recommender systems is the data sparsity that is caused by missing and noisy ratings. This problem is even more serious for CF domains where the ratings are expressed numerically, e.g. as 5-star grades. We observe that, while we may lack the information in numerical ratings, we sometimes have additional auxiliary data in the form of binary ratings. This is especially true given that users can easily express themselves with their preferences expressed as likes or dislikes for items. In this chapter, we explore how to use these binary auxiliary preference data to help reduce the impact of data sparsity for CF domains expressed in numerical ratings. We solve this problem by transferring the rating knowledge from some auxiliary data source in binary form (that is, likes or dislikes), to a target numerical-rating matrix.

In particular, our solution is to model both the numerical ratings and ratings expressed as like or dislike in a principled way. We present a novel framework of transfer by collective factorization (TCF), in which we construct a shared latent space collectively and learn the data-dependent effect separately. A major advantage of the TCF approach over the previous bilinear method of collective matrix factorization is that we are able to capture the data-dependent effect when sharing the data-independent knowledge. This allows us to increase the overall quality of knowledge transfer. We present extensive experimental results to demonstrate the effectiveness of TCF at various sparsity levels, and show improvements of our approach as compared to several state-of-the-art methods.

4.1 Introduction

Data sparsity is a major challenge in collaborative filtering methods [64, 26, 160]. Sparsity refers to the fact that some observed ratings, e.g. 5-star grades, in a user-item rating matrix are too few, such that overfitting can easily happen when we use a
prediction model for missing values in the test data. However, we observe that, some auxiliary data of the form “like or dislike” may be more easily obtained; examples are the favored/disfavored data in Moviepilot\textsuperscript{1} and Qiyi\textsuperscript{2}, the dig/bury data in Tudou\textsuperscript{3}, the love/ban data in Last.fm\textsuperscript{4}, and the “Want to see”/“Not Interested” data in Flixster\textsuperscript{5}. It is often more convenient for users to express such preferences instead of numerical ratings. The question we ask in this chapter is: how do we take advantage of auxiliary knowledge in the form of binary ratings to alleviate the sparsity problem in numerical ratings when we build a rating-prediction model?

To the best of our knowledge, no previous work answered the question of how to jointly model a target data of numerical ratings and an auxiliary data of binary ratings. There are some prior works on using both the numerical ratings and \textit{implicit} data of “whether rated” \cite{104, 128} or “whether purchased” \cite{226} to help boost the prediction performance. Among the previous works, Koren \cite{104} uses implicit data of “rated” as offsets in a factorization model, and Liu et al. \cite{128} adapt the collective matrix factorization (CMF) approach \cite{193} to integrate the implicit data of “rated”. Zhang et al. \cite{226} convert the implicit data of simulated purchases to a \textit{user-brand} matrix as a user-side meta data representing brand loyalty and a \textit{user-item} matrix of “purchased”. However, none of these previous works consider how to use auxiliary data in the form of like and dislike type of binary ratings in collaborative filtering in a transfer learning framework.

Most existing transfer learning methods in collaborative filtering consider auxiliary data from several perspectives, including user-side transfer \cite{138, 29, 228, 136, 207}, item-side transfer \cite{193}, or knowledge-transfer using related but not aligned data \cite{115, 116}. We illustrate the ideas of knowledge sharing from a matrix factorization view as shown in Table 4.1. We show four representative methods \cite{138, 193, 115, 116} in Table 4.1 and describe the details starting from a non-transfer learning method of probabilistic matrix factorization (PMF) \cite{177}.

**Probabilistic Matrix Factorization** The PMF \cite{177} or latent factorization model

\textsuperscript{1}http://www.moviepilot.de
\textsuperscript{2}http://www.qiyi.com
\textsuperscript{3}http://www.tudou.com
\textsuperscript{4}http://www.last.fm
\textsuperscript{5}http://www.flixster.com
Table 4.1: Matrix illustration of some related work on transfer learning in collaborative filtering.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Training Data</th>
<th>Auxiliary Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>SoRec (user side)</td>
<td>$R \sim UV^T$</td>
<td>$R_1 \sim U_1 V_1^T$</td>
</tr>
<tr>
<td>[138]</td>
<td>Knowledge sharing: $U = U_1$</td>
<td>Value domain: $(U, V), (U_1, V_1) \in D_R$</td>
</tr>
<tr>
<td></td>
<td>$D_R = { (U, V)</td>
<td>U \in \mathbb{R}^{n \times d}, V \in \mathbb{R}^{m \times d} }$</td>
</tr>
<tr>
<td>CMF (item side)</td>
<td>$R \sim UV^T$</td>
<td>$R_2 \sim U_2 V_2^T$</td>
</tr>
<tr>
<td>[193]</td>
<td>Knowledge sharing: $V = V_2$</td>
<td>Value domain: $(U, V), (U_2, V_2) \in D_R$</td>
</tr>
<tr>
<td></td>
<td>$D_R = { (U, V)</td>
<td>U \in \mathbb{R}^{n \times d}, V \in \mathbb{R}^{m \times d} }$</td>
</tr>
<tr>
<td>CBT (not aligned)</td>
<td>$R \sim UBV^T$</td>
<td>$R_3 \sim U_3 B_3 V_3^T$</td>
</tr>
<tr>
<td>[115]</td>
<td>Knowledge sharing: $B = B_3$</td>
<td>Value domain: $(U, V), (U_3, V_3) \in D_{[0,1]}$</td>
</tr>
<tr>
<td></td>
<td>$D_{[0,1]} = { (U, V)</td>
<td>U \in [0,1]^{n \times d}, U 1 \in 1, V \in [0,1]^{m \times d}, V 1 = 1 }$</td>
</tr>
<tr>
<td>RMGM (not aligned)</td>
<td>$R \sim UBV^T$</td>
<td>$R_3 \sim U_3 B_3 V_3^T$</td>
</tr>
<tr>
<td>[116]</td>
<td>Knowledge sharing: $B = B_3$</td>
<td>Value domain: $(U, V), (U_3, V_3) \in D_{[0,1]}$</td>
</tr>
<tr>
<td></td>
<td>$D_{[0,1]} = { (U, V)</td>
<td>U \in [0,1]^{n \times d}, U 1 \in 1, V \in [0,1]^{m \times d}, V 1 = 1 }$</td>
</tr>
</tbody>
</table>

(LFM) [13] seeks an appropriate low-rank approximation, $R = UV^T$, for which any missing value can be predicted by $\hat{r}_{ui} = U_u V_i^T$, where $U \in \mathbb{R}^{n \times d}$, $V \in \mathbb{R}^{m \times d}$ are user-specific and item-specific latent feature matrices, respectively. The optimization problem of PMF is as follows [177, 13],

$$\min_{U, V} \mathcal{E}_1(U, V) + \alpha \mathcal{R}(U, V)$$  \hspace{1cm} (4.1)

where $\mathcal{E}_1(U, V) = \frac{1}{2} \sum_{u=1}^n \sum_{i=1}^m y_{ui} (\hat{r}_{ui} - U_u V_i^T)^2 = \frac{1}{2} ||Y \odot (R - UV^T)||_F^2$ is the loss function, and $\mathcal{R} = \frac{1}{2} (\sum_{u=1}^n ||U_u||_F^2 + \sum_{i=1}^m ||V_i||_F^2) = \frac{1}{2} (||U||_F^2 + ||V||_F^2)$ is the regularization term used to avoid overfitting.
**Social Recommendation**  SoRec [138] is proposed to alternatively factorize the target rating matrix $R$ and a user-side social network matrix $R_1$ with the constraint of sharing the same user-specific latent feature matrix (see $\mathbf{U} = \mathbf{U}_1$ in Table 4.1). The objective function is formalized as follows [138],

$$\min_{\mathbf{U}, \mathbf{V}, \mathbf{U}_1} \text{obj}(\mathbf{U}, \mathbf{V}) + \text{obj}(\mathbf{U}, \mathbf{V}_1).$$  \hspace{1cm} (4.2)$$

where $(\mathbf{U}, \mathbf{V}) \in D_R$, and $\text{obj}(\mathbf{U}, \mathbf{V}) = \mathcal{E}_I(\mathbf{U}, \mathbf{V}) + \alpha \mathcal{R}(\mathbf{U}, \mathbf{V})$ is the same objective function as in Eq.(4.1).

**Collective Matrix Factorization**  CMF [193] is proposed to alternatively factorize the target rating matrix $R$ and an item-side content matrix $R_2$ with the constraint of sharing the same item-specific latent feature matrix (see $\mathbf{V} = \mathbf{V}_2$ in Table 4.1). This approach is similar to that in SoRec [138], but with different auxiliary data. The optimization problem of CMF is stated as follows [193],

$$\min_{\mathbf{U}, \mathbf{V}, \mathbf{U}_2} \text{obj}(\mathbf{U}, \mathbf{V}) + \text{obj}(\mathbf{U}_2, \mathbf{V}) \hspace{1cm} (4.3)$$

where $(\mathbf{U}, \mathbf{V}) \in D_R$, and $\text{obj}(\mathbf{U}, \mathbf{V}) = \mathcal{E}_I(\mathbf{U}, \mathbf{V}) + \alpha \mathcal{R}(\mathbf{U}, \mathbf{V})$ is the same objective function as in Eq.(4.1).

**Codebook Transfer**  The CBT [115] method consists of codebook construction and expansion steps. It achieves knowledge transfer with the assumption that both auxiliary and target data share a common cluster-level rating pattern (see $\mathbf{B} = \mathbf{B}_3$ in Table 4.1).

1. **Codebook Construction.**  Assume that $(\mathbf{U}_3, \mathbf{V}_3) \in D_{\{0,1\}}$ are user-specific and item-specific membership indicator matrices of the auxiliary rating matrix $R_3$, which are obtained using co-clustering algorithms such as NMF [49]. The constructed codebook is represented as $\mathbf{B}_3 = [\mathbf{U}_3^T \mathbf{R}_3 \mathbf{V}_3] \odot [\mathbf{U}_3^T (\mathbf{R}_3 > 0) \mathbf{V}_3]$ [115], where $[\mathbf{U}_3^T \mathbf{R}_3 \mathbf{V}_3]_{k\ell}$ denotes the summation of ratings by users in a user cluster $k$ on items in an item cluster $\ell$. $[\mathbf{U}_3^T (\mathbf{R}_3 > 0) \mathbf{V}_3]_{k\ell}$ denotes the number of ratings from users in a user cluster $k$ on items in an item cluster $\ell$, hence, the element-wise division $\odot$ resembles the idea of normalization, and $[\mathbf{B}_3]_{k\ell}$ is the average rating of users in a user cluster $k$ on items in an item cluster $\ell$.

2. **Codebook Expansion.**  The codebook expansion problem is formalized as follows [115],

$$\min_{\mathbf{U}, \mathbf{V}} \mathcal{E}_B(\mathbf{U}, \mathbf{V}) \hspace{0.5cm} \text{s.t.} \hspace{0.5cm} (\mathbf{U}, \mathbf{V}) \in D_{\{0,1\}}$$  \hspace{1cm} (4.4)
where \( \mathcal{E}_B(U, V) = \frac{1}{2} \| Y \odot (R - UBV^T) \|_F^2 \) is a B-regularized square loss function, and \( B = B_3 \) is the codebook constructed from the auxiliary data \( R_3 \). In [115]. An alternating greedy-search algorithm is proposed to solve the combinatorial optimization problem in Eq.(4.4), and the choice of \( U_{uk} = 1 \), \( V_{i\ell} = 1 \) are used to select the entry located at \( (k, \ell) \) of \( B \) via \( [UBV^T]_{ui} = U_u B V_i^T \). Thus, the predicted rating \( \hat{r}_{ui} = [UBV^T]_{ui} = [B]_{k\ell} \) is the average rating of users in the user cluster \( k \) on items in an item cluster \( \ell \) of the auxiliary data.

**Rating-Matrix Generative Model**  RMGM [116] is derived and extended from the FMM generative model [190], and we re-write it in a matrix factorization manner,

\[
\min_{U, V, B, U_3, V_3} \mathcal{E}_B(U, V) + \mathcal{E}_B(U_3, V_3) \quad \text{s.t.} \quad (U, V), (U_3, V_3) \in D_{[0,1]} \tag{4.5}
\]

where \( \mathcal{E}_B(U, V) \) is again a B-regularized loss function, the same as given in Eq.(4.4). We can see that RMGM is different from CBT since it learns \( (U, V) \) and \( (U_3, V_3) \) alternatively and relaxes the hard membership requirement as imposed by the indicator matrix, e.g. \( U \in \{0,1\}^{n \times d} \). A soft indicator matrix is used in RMGM [116], e.g., \( U \in [0,1]^{n \times d} \).

In this chapter, we consider the situation where the auxiliary data is such that the following information are aligned: users and items of the target rating matrix and the auxiliary binary rating matrix. This assumption gives us precise information on the mapping between auxiliary and target data, which can lead to higher performance than not having this knowledge. We illustrate the idea of these assumptions using matrices in Table 4.2, where we can see that our problem setting and proposed solution are both novel and different from the previous ones as shown in Table 4.1. We will discuss these novelty in the sequel.

The organization of this chapter is as follows. We give a formal definition of the problem in Section 4.2 and then describe our solution in detail in Section 4.3. We present experimental results on real-world data sets in Section 4.4, and discuss about some related work in Section 4.5. Finally, we give some concluding remarks and future works in Section 4.6.
### 4.2 Collaborative Filtering with Binary Ratings

#### 4.2.1 Problem Definition

In the target data, we have a user-item numerical rating matrix \( R = [r_{ui}]_{n \times m} \in \{1, 2, 3, 4, 5, ?\}^{n \times m} \) with \( q \) observed ratings, where the question mark “?” denotes a missing value, which can be an unobserved value. Note that the observed rating values in \( R \) are not limited to 5-star grades; instead, they can be any real numbers. We use an indicator matrix \( Y = [y_{ui}]_{n \times m} \in \{0, 1\}^{n \times m} \) to denote whether the entry \((u, i)\) is observed \((y_{ui} = 1)\) or not \((y_{ui} = 0)\), and \( \sum_{u,i} y_{ui} = q \). Similarly, in the auxiliary data, we have a user-item binary rating matrix \( \tilde{R} = [\tilde{r}_{ui}]_{n \times m} \in \{0, 1, ?\}^{n \times m} \) with \( \tilde{q} \) observations, where a value of one denotes the observed ‘like’ value, and zero denotes the observed ‘dislike’ value. The question mark denotes the missing value. Similar to the target data, we have a corresponding indicator matrix \( \tilde{Y} = [\tilde{y}_{ui}]_{n \times m} \in \{0, 1\}^{n \times m} \), and \( \sum_{u,i} \tilde{y}_{ui} = \tilde{q} \). Note that there is an one-one mapping between the users and items of \( R \) and \( \tilde{R} \). Our goal is to predict the missing values in \( R \) by transferring the rating knowledge from \( \tilde{R} \). Note that the missing values here are different from the implicit data used in [104, 128, 226], which can be represented as \( \{1, ?\}^{n \times m} \), since implicit data corresponds to positive observations only.

---

**Table 4.2: Matrix illustration of transfer by collective factorization.**

<table>
<thead>
<tr>
<th>Variants of TCF</th>
<th>Training Data</th>
<th>Auxiliary Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMTF (frontal side)</td>
<td>[ R \sim UV^T ]</td>
<td>[ \tilde{R} \sim \tilde{U} \tilde{V}^T ]</td>
</tr>
<tr>
<td>Knowledge sharing: ( U = \tilde{U}, V = \tilde{V} )</td>
<td>[ \text{Value domain: } (U, V), (\tilde{U}, \tilde{V}) \in D_R ]</td>
<td>[ \text{Value domain: } (U, V), (\tilde{U}, \tilde{V}) \in D_{\perp} ]</td>
</tr>
<tr>
<td>( D_R = {(U, V)</td>
<td>U \in \mathbb{R}^{n \times d}, V \in \mathbb{R}^{m \times d}} )</td>
<td></td>
</tr>
</tbody>
</table>

CSVD (frontal side)

| \[ R \sim UV^T \] | \[ \tilde{R} \sim \tilde{U} \tilde{V}^T \] |
| Knowledge sharing: \( U = \tilde{U}, V = \tilde{V} \) | \[ \text{Value domain: } (U, V), (\tilde{U}, \tilde{V}) \in D_{\perp} \] |
| \[ D_{\perp} = \{(U, V) | U \in \mathbb{R}^{n \times d}, U^T U = I, V \in \mathbb{R}^{m \times d}, V^T V = I\} \] |
4.2.2 Challenges

Our problem setting is novel and challenging. In particular, we enumerate the following challenges for the problem setting (see Figure 4.1).

1. **How to make use of the existing correspondences** among users and items from two domains, given that such relationships are important and can serve as a bridge across two domains. Some previous solutions were proposed without such correspondences [115, 116], and are thus imprecise. Other works have used correspondence information as additional constraints on the user-specific or item-specific latent feature matrices [138, 193].

2. **What to transfer and how to transfer**, as raised in [157]. Previous works that address this question include approaches that transfer the knowledge of latent features in an adaptive way [160] or in a collective way [138, 193]. Some works in this direction include those that transfer cluster-level rating patterns [115] in an adaptive manner or in a collective manner [116].

3. **How to model the data-dependent effect** of numerical ratings and binary ratings when sharing the data-independent knowledge? This question is important since clearly the auxiliary and target data may have different distributions and quite different semantic meanings.
From Table 4.1, we can see that the solutions of [138, 193, 115, 116] were proposed for different problem settings as compared to ours, which is in Table 4.2 and Figure 4.1. More specifically, for the aforementioned three challenges from our problem setting, the approaches of [138, 193] cannot capture the data-dependent information, and the methods of [115, 116] cannot make use of the existing correspondence information.

### 4.2.3 Overview of Our Solution

We propose a principled matrix-based transfer-learning framework referred to as *transfer by collective factorization*, which jointly factorizes the data matrices in three parts: a user-specific latent feature matrix, an item-specific latent feature matrix, and two data-dependent inner matrices. Specifically, the main idea of our solution has two major steps. First, we factorize both the target numerical rating matrix, $\mathbf{R} \sim \mathbf{UBV}^T$, and the auxiliary binary rating matrix, $\tilde{\mathbf{R}} \sim \tilde{\mathbf{UBV}}^T$, with constraints of sharing user-specific latent feature matrix $\mathbf{U} = \tilde{\mathbf{U}}$ and item-specific latent feature matrix $\mathbf{V} = \tilde{\mathbf{V}}$ (see Table 4.2 for matrix illustration). Second, we learn the inner matrices $\mathbf{B}$ and $\tilde{\mathbf{B}}$ separately in each domain to capture the domain-dependent information, since the semantic meaning and distributions of numerical ratings and binary ratings may be different. Those two major steps are iterated to have richer interactions for knowledge sharing [33, 200] until we reach convergence to a locally optimal state. The intuition of our approach is that same users and items in two domains are likely to have the same latent feature matrices, e.g. $\mathbf{U} = \tilde{\mathbf{U}}$ and $\mathbf{V} = \tilde{\mathbf{V}}$, while the domain differences, the data-dependent information, are left for the inner matrices, $\mathbf{B}$ and $\tilde{\mathbf{B}}$. In summary, our major contributions are:

1. We make full use of the correspondences among users and items, from a source domain and a target domain. We allow the aligned users and items to share the same user-specific latent feature matrix and item-specific latent feature matrix, respectively.

2. We construct a shared latent space to address the what to transfer question, via a matrix tri-factorization, or trilinear, method in a collective way to address the how to transfer question.

3. We model the data-dependent effect of binary ratings and numerical ratings by learning the inner matrices of trilinear method separately.
4.3 Transfer by Collective Factorization

4.3.1 Model Formulation

We assume that a user $u$’s rating on an item $i$ in the target data, $r_{ui}$, is generated from the user-specific latent feature vector $U_u \in \mathbb{R}^{1 \times d_u}$, item-specific latent feature vector $V_i \in \mathbb{R}^{1 \times d_v}$, and some data-dependent effect denoted as $B \in \mathbb{R}^{d_u \times d_v}$. Note that this formulation is different from the PMF model [177], which only contains $U_u$ and $V_i$. Our graphical model is shown in Figure 4.1, where $U_u, u = 1, \ldots, n$ and $V_i, i = 1, \ldots, m$ are shared to bridge two data, while $B, \tilde{B}$ are designed to capture the data-dependent effect. We fix $d = d_u = d_v$ for notation simplicity in the sequel. We define a conditional distribution as

$$p(r_{ui}|U_u, B, V_i, \alpha_r) = \mathcal{N}(r_{ui}|U_u TV_i^T, \alpha_r^{-1}),$$

where $\mathcal{N}(x|\mu, \alpha^{-1}) = \sqrt{\frac{\alpha}{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\alpha}\right)$ is the Gaussian distribution with mean $\mu$ and precision $\alpha$. We further define the prior distributions over $U_u, V_i$ and $B$ as $p(U_u|\alpha_u) = \mathcal{N}(U_u|0, \alpha_u^{-1}I)$, $p(V_i|\alpha_v) = \mathcal{N}(V_i|0, \alpha_v^{-1}I)$, and $p(B|\beta) = \mathcal{N}(B|0, (\beta/q)^{-1}I)$. We then have the log-posterior function over the latent variables $U \in \mathbb{R}^{n \times d}, B \in \mathbb{R}^{d \times d}$ and $V \in \mathbb{R}^{m \times d}$ via Bayesian inference,

$$\log p(U, B, V | R, \alpha_r, \alpha_u, \alpha_v, \beta)$$

$$= \log \prod_{u=1}^{n} \prod_{i=1}^{m} \prod_{u=1}^{n} \prod_{i=1}^{m} [p(r_{ui}|U_u, B, V_i, \alpha_r)p(U_u|\alpha_u)p(V_i|\alpha_v)p(B|\beta)]^{y_{ui}}$$

$$= \log \prod_{u=1}^{n} \prod_{i=1}^{m} \prod_{u=1}^{n} \prod_{i=1}^{m} [\mathcal{N}(r_{ui}|U_u TV_i^T, \alpha_r^{-1})\mathcal{N}(U_u|0, \alpha_u^{-1}I)\mathcal{N}(V_i|0, \alpha_v^{-1}I)\mathcal{N}(B|0, (\beta/q)^{-1}I)]^{y_{ui}}$$

$$= -\sum_{u=1}^{n} \sum_{i=1}^{m} y_{ui} \left[ \frac{\alpha_r}{2} (r_{ui} - U_u TV_i^T)^2 + \frac{\alpha_u}{2} ||U_u||_F^2 + \frac{\alpha_v}{2} ||V_i||_F^2 + \frac{\beta}{2q} ||B||_F^2 + C \right]$$

where $C = \ln \sqrt{\frac{\alpha_r}{2\pi}} + \ln \sqrt{\frac{\alpha_u}{2\pi}} + \ln \sqrt{\frac{\alpha_v}{2\pi}} + \ln \sqrt{\frac{\beta}{2q\pi}}$ is a constant. Setting $\alpha_r = 1$, we have

$$-\sum_{u=1}^{n} \sum_{i=1}^{m} y_{ui} \left[ \frac{1}{2} (r_{ui} - U_u TV_i^T)^2 + \frac{\alpha_u}{2} ||U_u||_F^2 + \frac{\alpha_v}{2} ||V_i||_F^2 - \frac{\beta}{2} ||B||_F^2. \right]$$

Similarly, in the auxiliary data, we have a log-posterior function for the matrix trifactorization, or trilinear, model, $\log p(U, \tilde{B}, \tilde{V} | \tilde{R}, \alpha_r, \alpha_u, \alpha_v, \beta)$. To jointly maximize
these two log-posterior functions, we have

$$\max_{U, V, B, \tilde{B}} \log p(U, B, V | R, \alpha_r, \alpha_u, \alpha_v, \beta) + \lambda \log p(U, \tilde{B}, V | \tilde{R}, \alpha_r, \alpha_u, \alpha_v, \beta)$$

s.t. \(U, V \in \mathcal{D}\)

where \(\lambda > 0\) is a tradeoff parameter to balance the target and auxiliary data and \(\mathcal{D}\) is the value domain of the latent variables. \(\mathcal{D}\) can be \(\mathcal{D}_R = \{U \in \mathbb{R}^{n \times d}, V \in \mathbb{R}^{m \times d}\}\) or \(\mathcal{D}_\perp = \mathcal{D}_R \cap \{U^T U = I, V^T V = I\}\) to get the effect of finding latent topics [47, 160] and noise reduction [17, 96] in SVD. Thus we have two variants of TCF, CMTF (collective matrix tri-factorization) for \(\mathcal{D}_R\) and CSVD (collective SVD) for \(\mathcal{D}_\perp\). Although 2DSVD or Tucker2 [50] can factorize a sequence of full matrices, it does not achieve the goal of missing-value prediction in sparse observation matrices, which is accomplished in our proposed approach.

Finally, we obtain the following equivalent minimization problem for TCF,

$$\min_{U, V, B, \tilde{B}} \mathcal{F}(R \sim UBV^T) + \lambda \mathcal{F}(\tilde{R} \sim U\tilde{B}V^T) \quad \text{s.t. } U, V \in \mathcal{D} \quad (4.6)$$

where,

$$\mathcal{F}(R \sim UBV^T) = \sum_{u=1}^{n} \sum_{i=1}^{m} y_{ui} \left[ \frac{1}{2} (r_{ui} - U_u B V_i^T)^2 \right] + \frac{\alpha_u}{2} ||U_u||^2 + \frac{\alpha_v}{2} ||V_i||^2 + \frac{\beta}{2} ||B||_F^2,$$

$$\mathcal{F}(\tilde{R} \sim U\tilde{B}V^T) = \sum_{u=1}^{n} \sum_{i=1}^{m} y_{ui} \left[ \frac{1}{2} (\tilde{r}_{ui} - U_u \tilde{B} V_i^T)^2 \right] + \frac{\alpha_u}{2} ||U_u||^2 + \frac{\alpha_v}{2} ||V_i||^2 + \frac{\beta}{2} ||\tilde{B}||_F^2.$$

To solve the optimization problem in Eq.(4.6), we first collectively factorize two data matrices of \(R\) and \(\tilde{R}\) to learn \(U\) and \(V\). We then estimate \(B\) and \(\tilde{B}\) separately. We transfer the knowledge of latent feature matrices, \(U\) and \(V\) via collective factorization of the rating matrices \(R\) and \(\tilde{R}\). For this reason, we call our approach transfer by collective factorization.

### 4.3.2 Learning the TCF

**Learning \(U\) and \(V\) in CMTF** Given \(B\) and \(V\), we show that the user-specific latent feature matrix \(U\) in Eq.(4.6) can be obtained analytically.
**Theorem 2.** Given $\mathbf{B}$ and $\mathbf{V}$, we can obtain the user-specific latent feature matrix $\mathbf{U}$ in a closed form.

**Proof.** Let $f_u = \sum_{i=1}^{m} y_{ui} \left[ \frac{1}{2} (r_{ui} - U_u B_i V_i^T)^2 + \frac{\alpha_u}{2} \|U_u\|^2 + \frac{\alpha_v}{2} \|V_i\|^2 + \frac{\beta}{2} \|\mathbf{B}\|^2_F + \lambda \left\{ \sum_{i=1}^{m} \tilde{y}_{ui} \left[ \frac{1}{2} (\tilde{r}_{ui} - U_u \tilde{B}_i V_i^T)^2 + \frac{\alpha_v}{2} \|U_u\|^2 + \frac{\alpha_u}{2} \|V_i\|^2 + \frac{\beta}{2} \|\tilde{\mathbf{B}}\|^2_F \right] \right\} \right]$.

\[
\frac{\partial f_u}{\partial U_u} = \sum_{i=1}^{m} y_{ui} \left[ (-r_{ui} + U_u B_i V_i^T) V_i B_i^T + \alpha_u U_u \right] + \lambda \sum_{i=1}^{m} \tilde{y}_{ui} \left[ (-\tilde{r}_{ui} + U_u \tilde{B}_i V_i^T) V_i \tilde{B}_i^T + \alpha_u U_u \right]
\]

\[
= - \sum_{i=1}^{m} (y_{ui} r_{ui} V_i B_i^T + \lambda \tilde{y}_{ui} \tilde{r}_{ui} V_i \tilde{B}_i^T) + \alpha_u U_u \sum_{i=1}^{m} (y_{ui} B_i^T V_i B_i^T + \lambda \tilde{y}_{ui} \tilde{B}_i^T V_i \tilde{B}_i^T).
\]

Setting $\frac{\partial f_u}{\partial U_u} = 0$, we have the update rule for each $U_u$,

\[
U_u = b_u C_u^{-1}, \quad (4.7)
\]

where $C_u = \sum_{i=1}^{m} (y_{ui} B_i^T V_i B_i^T + \lambda \tilde{y}_{ui} \tilde{B}_i^T V_i \tilde{B}_i^T) + \alpha_u \sum_{i=1}^{m} (y_{ui} + \lambda \tilde{y}_{ui}) I$ and $b_u = \sum_{i=1}^{m} (y_{ui} r_{ui} V_i B_i^T + \lambda \tilde{y}_{ui} \tilde{r}_{ui} V_i \tilde{B}_i^T)$.

We can see that $U_u$ in Eq.(4.7) is independent of all other users’ latent features given $\mathbf{B}$ and $\mathbf{V}$, thus we can obtain the user-specific latent feature matrix $\mathbf{U}$ analytically.

Similarly, given $\mathbf{B}$ and $\mathbf{U}$, the latent feature vector $V_i$ of each item $i$ can be estimated in a closed form, and thus the whole item-specific latent feature matrix $\mathbf{V}$ can be obtained analytically.

\[
V_i = b_i C_i^{-1}, \quad (4.8)
\]

where $C_i = \sum_{u=1}^{n} (y_{ui} B_i^T U_u^T U_u B + \lambda \tilde{y}_{ui} \tilde{B}_i^T U_u \tilde{B}) + \alpha_v \sum_{u=1}^{n} (y_{ui} + \lambda \tilde{y}_{ui}) I$ and $b_i = \sum_{u=1}^{n} (y_{ui} r_{ui} U_u B + \lambda \tilde{y}_{ui} \tilde{r}_{ui} U_u \tilde{B})$.

The closed-form update rule in Eq.(4.7) or Eq.(4.8) can be considered as a generalization of the alternating least square (ALS) approach in [13]. Note that Bell et al. [13]
consider bilinear model in a single matrix, which is different from our trilinear models of two matrices.

**Learning U and V in CSVD** Since the constraints \(D_\perp\) have similar effect of regularization, we remove the regularization terms in Eq.(4.6) and reach a simplified optimization problem,

\[
\begin{align*}
\min_{U,V} & \quad \frac{1}{2}||Y \odot (R - UBV^T)||^2_F + \frac{\lambda}{2}||\tilde{Y} \odot (\tilde{R} - U\tilde{B}V^T)||^2_F \\
\text{s.t.} & \quad U^T U = I, \quad V^T V = I
\end{align*}
\]  

(4.9)

Let \(f = \frac{1}{2}||Y \odot (R - UBV^T)||^2_F + \frac{\lambda}{2}||\tilde{Y} \odot (\tilde{R} - U\tilde{B}V^T)||^2_F\). We have the gradients on \(U\) as follows,

\[
\frac{\partial f}{\partial U} = (Y \odot (UBV^T - R))VB^T + \lambda(\tilde{Y} \odot (U\tilde{B}V^T - \tilde{R}))\tilde{V}\tilde{B}^T.
\]

Then, the variable \(U\) can be learned via a gradient descent algorithm on the Grassmann manifold [56, 27, 96],

\[
U \leftarrow U - \gamma(I - UU^T)\frac{\partial f}{\partial U} = U - \gamma\nabla U.
\]  

(4.10)

We now show that \(\gamma\) can be obtained analytically in the following theorem.

**Theorem 3.** The step size \(\gamma\) in Eq.(4.10) can be obtained analytically.

**Proof.** Plugging in the update rule in Eq.(4.10) into the objective function in Eq.(4.9), we have,

\[
g(\gamma) = \frac{1}{2}||Y \odot (R - UBV^T)||^2_F + \frac{\lambda}{2}||\tilde{Y} \odot (\tilde{R} - U\tilde{B}V^T)||^2_F
\]

Denoting \(t_1 = Y \odot (R - UBV^T), t_2 = Y \odot (U\nabla B V^T)\), we have \(g(\gamma) = \frac{1}{2}||t_1 + \gamma t_2||^2_F + \frac{\lambda}{2}||\tilde{t}_1 + \gamma \tilde{t}_2||^2_F\), and the gradient,

\[
\frac{\partial g(\gamma)}{\partial \gamma} = \text{tr}(t_1^T t_2) + \gamma \text{tr}(t_2^T t_2) + \lambda[\text{tr}(\tilde{t}_1^T \tilde{t}_2) + \gamma \text{tr}(\tilde{t}_2^T \tilde{t}_2)].
\]

74
from which we obtain \( \gamma = \frac{-\text{tr}(t_1^T t_2) - \lambda \text{tr}(\tilde{t}_1^T \tilde{t}_2)}{\text{tr}(t_1^T t_2) + \lambda \text{tr}(\tilde{t}_1^T \tilde{t}_2)} \) via setting \( \frac{\partial g(\gamma)}{\partial \gamma} = 0. \)

Similarly, we have the update rule for the item-specific late \( nt \) feature matrix \( V \),

\[
V \leftarrow V - \gamma \nabla V.
\] (4.11)

where \( \nabla V = (I - VV^T) \frac{\partial f}{\partial V} \), and \( \frac{\partial f}{\partial V} = (Y \odot (UBV^T - R))UB + \lambda(\tilde{Y} \odot (UBV^T - \tilde{R}))UB \).

Note that the previous works of [27, 96] use the gradient descent approach also on a Grassmann manifold. But, they study a single-matrix factorization problem and adopt a different learning algorithm on the Grassmann manifold for searching the step size \( \gamma \).

**Learning \( B \) and \( \tilde{B} \)** Given \( U, V \), we can estimate \( B \) and \( \tilde{B} \) separately in each data, e.g. for the target data,

\[
F(R \sim UBV^T) \propto \sum_{u=1}^{n} \sum_{i=1}^{m} y_{ui} \left[ \frac{1}{2}(r_{ui} - U_uBV_i^T)^2 \right] + \frac{\beta}{2} \| B \|^2_F
\]

\[
= \frac{1}{2} \| Y \odot (R - UBV^T) \|^2_F + \frac{\beta}{2} \| B \|^2_F.
\]

Thus, we obtain the following equivalent minimization problem,

\[
\min_{B} \frac{1}{2} \| Y \odot (R - UBV^T) \|^2_F + \frac{\beta}{2} \| B \|^2_F
\] (4.12)

where the data-dependent parameter \( B \) can be estimated exactly the same as that of estimating \( w \) in a corresponding least square SVM problem, where \( w = \text{vec}(B) = [B_1^T \ldots B_d^T] \in \mathbb{R}^{d^2 \times 1} \) is a large vector that is concatenated from columns of matrix \( B \). The instances can be constructed as \( \{ x_{ui}, r_{ui} \} \) with \( y_{ui} = 1 \), where \( x_{ui} = \text{vec}(U_{u}^T V_i) \in \mathbb{R}^{d^2 \times 1} \). Hence, we obtain the following least-square SVM problem,

\[
\min_{w} \frac{1}{2} \| r - Xw \|^2_F + \frac{\beta}{2} \| w \|^2_F
\] (4.13)

where \( X = [\ldots x_{ui} \ldots]^T \in \mathbb{R}^{p \times d^2} \) (with \( y_{ui} = 1 \)) is the data matrix, and \( r \in \{1, 2, 3, 4, 5\}^{p \times 1} \) is the corresponding observed ratings from \( R \). Setting \( \nabla w = -X^T(r - Xw) + \beta w = 0 \), we have,

\[
w = (X^TX + \beta I)^{-1}X^Tr.
\] (4.14)
Input: The target *user-item* numerical rating matrix $\mathbf{R}$, the auxiliary *user-item* binary rating matrix $\tilde{\mathbf{R}}$, the target *user-item* indicator matrix $\mathbf{Y}$, the auxiliary *user-item* indicator matrix $\tilde{\mathbf{Y}}$.

Output: The shared user-specific latent feature matrix $\mathbf{U}$, the shared item-specific latent feature matrix $\mathbf{V}$, the inner matrix to model the target data-dependent information $\mathbf{B}$, the inner matrix to model the auxiliary data-dependent information $\tilde{\mathbf{B}}$.

Step 1. Scale ratings in $\mathbf{R}$ ($r_{ui} = \frac{r_{ui} - 1}{4}, y_{ui} = 1, u = 1 \ldots n, i = 1 \ldots m$).

Step 2. Initialize $\mathbf{U}, \mathbf{V}$: randomly initialize $\mathbf{U}$ and $\mathbf{V}$ for CMTF; initialize $\mathbf{U}$ and $\mathbf{V}$ in CSVD using the SVD [41] results of $\tilde{\mathbf{R}}$.

Step 3. Estimate $\mathbf{B}$ and $\tilde{\mathbf{B}}$ as shown in Eq.(4.14).

Step 4. Update $\mathbf{U}, \mathbf{V}, \mathbf{B}, \tilde{\mathbf{B}}$.

repeat
  repeat
    Step 4.1.1 Fix $\mathbf{B}$ and $\mathbf{V}$, update $\mathbf{U}$ in CMTF as shown in Eq.(4.7) or CSVD as shown in Eq.(4.10).
    Step 4.1.2 Fix $\mathbf{B}$ and $\mathbf{U}$, update $\mathbf{V}$ in CMTF as shown in Eq.(4.8) or CSVD as shown in Eq.(4.11).
  until Convergence

Step 4.2. Fix $\mathbf{U}$ and $\mathbf{V}$, update $\mathbf{B}$ and $\tilde{\mathbf{B}}$ as shown in Eq.(4.14).

until Convergence

Figure 4.2: The algorithm of transfer by collective factorization (TCF).

Note that $\mathbf{B}$ or $\mathbf{w}$ can be considered as a linear compact operator [1] and solved efficiently using various existing off-the-shelf tools.

Finally, we can solve the optimization problem in Eq.(4.6) by alternatively estimating $\mathbf{B}$, $\tilde{\mathbf{B}}$, $\mathbf{U}$ and $\mathbf{V}$. The complete algorithm is given in Figure 4.2. Note that we can scale the target matrix $\mathbf{R}$ with $r_{ui} = \frac{r_{ui} - 1}{4}, y_{ui} = 1, u = 1 \ldots n, i = 1 \ldots m$, in order to remove the value range difference of two data sources. We adopt random initialization for $\mathbf{U}, \mathbf{V}$ in CMTF and SVD results [41] of $\tilde{\mathbf{R}}$ for that in CSVD.

4.3.3 Analysis

Each sub-step of updating $\mathbf{B}$, $\tilde{\mathbf{B}}$, $\mathbf{U}$ and $\mathbf{V}$ in Figure 4.2 will monotonically decrease the objective function in Eq.(4.6), and hence ensure the convergence to a local minimum. We use a validation dataset to determine the convergence condition and tune the parameters (see Section 4.4.3). The time complexity of TCF is $O(K \max(q, \tilde{q})d^3 + Kd^6)$, where $K$ is the number of iterations to convergence, $q, \tilde{q}$ ($q, \tilde{q} > n, m$) is the number of non-missing entries in the matrix $\mathbf{R}$ and $\tilde{\mathbf{R}}$, respectively, and $d$ is the num-
ber of latent features.

Note that the TCF algorithm can be sped up via a stochastic sampling (or stochastic gradient descent) algorithm or distributed computing. More specifically, the step for estimating $B$ or $\tilde{B}$ in both CMTF and CSVD is equivalent to that of least square SVM, thus various existing off-the-shelf tools can be used, e.g. we can use the stochastic sampling (or stochastic gradient descent) method [24] and distributed algorithms [34]. Second, the step for estimating $U$, $V$ in CMTF can be distributed same as that of PMF and CMF. For example, once $B$ and $V$ are given, each user $u$’s latent feature vector $U_u$ is independent of that of other users, which fits the Map/Reduce framework well [216].

### 4.3.4 Extensions of TCF

In this section, we will show that the TCF framework is quite flexible and can be extended from two-matrix and two-mode (user and item) setting to various settings in a straightforward manner, namely multi-matrix (or multi-slice) setting, single-mode setting, and multi-mode setting.

**TCF for Multi-Matrix Setting**

TCF is proposed for the problem of two-matrix factorization as shown in Figure 4.1, but it can be extended to multi-matrix (or multi-slice) setting as follows,

$$R^t \sim UB^tV^T, \quad t = 1, \ldots, \tau.$$ 

where we have $\tau$ user-item matrices, and the index $t$ can be denoted as time, location or other context dimension. The optimization problem can then be formulated as,

$$\min_{U, V, B^t, t = 1, \ldots, \tau} \sum_{t=1}^{\tau} \lambda^t F(R^t \sim UB^tV^T), \quad \text{s.t. } U, V \in \mathcal{D}$$

(4.15)

where all data sources share the data-independent user-specific and item-specific latent feature matrices, $U$ and $V$, respectively. The data-dependent variable, $B^t$, $t = 1, \ldots, \tau$ is used to capture the matrix-dependent information. Such an extension can be considered as a new weighted tensor factorization [101] approach, and for this reason, we denote this extension as TCF$^{TF}$.

Comparing to other tensor factorization methods of PARAFAC [76, 31] and HOSVD [111], TCF$^{TF}$ is very suitable for the setting of evolutionary matrix factorization with missing
values, where the matrix (or slice) can come on the fly. The newly arrived matrix (slice) can be incorporated to the TCF framework or algorithm (Figure 4.2) seamlessly, while it is difficult for HOSVD or PARAFAC.

**TCF for Single-Mode Setting**

The TCF framework can also be used for data with single type of entity (single mode), e.g. users,

\[ R^t \sim UB^tU^T, \quad t = 1, \ldots, \tau. \]

where \( R^t \in \mathbb{R}^{n \times n} \) is a square matrix, and \( U \) is shared among multiple data matrices. The optimization problem can then be formulated as,

\[
\min_{U, B^t, t = 1, \ldots, \tau} \sum_{t=1}^{\tau} \lambda_t \mathcal{F}(R^t \sim UB^tU^T), \quad \text{s.t. } U \in \mathcal{D} \tag{4.16}
\]

The single-mode setting is related to the multi-dimensional networks [202], which contains different types of interactions (or activities) among the user-user social networks. We can use TCF to detect communities among users, e.g. we can first obtain the latent feature matrix \( U \) and then apply \( k \)-means clustering on \( U \) to get community structure. The difference between traditional multi-view clustering and TCF is that we allow missing values (unknown entries) in the observed data matrices \( R^t, t = 1, \ldots, \tau \).

**TCF for Multi-Mode Setting**

The proposed TCF framework can also be extended to high-order data with more than two types of entities (multi-mode), e.g. collective factorization of two 3-D tensors of (user, advertisement, location) triples with values of click rate,

\[
\min_{U, V, T, B, \hat{B}} \mathcal{P} \sim (U, V, T, B) + \lambda \hat{\mathcal{P}} \sim (U, V, T, \hat{B}), \quad \text{s.t. } U, V, T \in \mathcal{D} \tag{4.17}
\]

where the latent feature matrices \( U, V, T \) are shared between the target tensor \( \mathcal{P} \) and auxiliary tensor \( \hat{\mathcal{P}} \), while the core tensors \( B \) and \( \hat{B} \) are used to capture the data-dependent effect. This problem can be referred as weighted collective tensor factorization. The optimization problem in Eq.(4.17) can also be extended to (i) hybrid data with different orders, e.g. transfer from a 3-D tensor to a 2-D matrix via sharing the corresponding latent feature matrices, or (ii) multiple high-order data sources.
The multi-mode setting can also be considered as a collective or multi-task extension of the so-called multi-mode networks [202], which contains more than two types of entities in the information networks, e.g. the social tagging system, academic information networks, etc.

### 4.4 Experimental Results

Our experiments are designed to verify the following hypotheses. We believe that transfer learning is effective in addressing the data sparsity problem in collaborative filtering, although the smoothing methods are very competitive baselines for the task of missing value prediction in a sparse rating matrix. In particular,

(a) We believe that the proposed transfer learning methods, CMTF and CSVD, perform better than baseline algorithms;

(b) We believe that the transfer learning method CMTF-link is better than the non-transfer learning methods of PMF [177], SVD [181] and OptSpace [96];

(c) We believe that the transfer learning method CMTF is better than CMF-link, since the inner matrices $B$ and $\tilde{B}$ in CMTF are used to capture data-dependent information;

(d) We believe that the transfer learning method CSVD is better than CMTF, since the orthonormal constraints in CSVD can selectively transfer the most useful knowledge via noise reduction.

We verify each of the the above four hypotheses in Section 4.4.3.

#### 4.4.1 Data Sets and Evaluation Metrics

We evaluate the proposed method using two movie rating data sets, Moviepilot and Netflix\(^6\), and compare to some state-of-the-art baseline algorithms.

**Subset of Moviepilot Data** The Moviepilot rating data contains more than $4.5 \times 10^6$ ratings with values in $[0, 100]$, which are given by more than $1.0 \times 10^5$ users on around $2.5 \times 10^4$ movies [175]. The data set used in the experiments is constructed as follows,

\(^6\)http://www.netflix.com
1. We first randomly extract a 2,000 × 2,000 dense rating matrix \( R \) from the Moviepilot data. We then normalize the ratings by \( \frac{r_{ui}}{25} + 1 \), and the new rating range is [1, 5];

2. We randomly split \( R \) into training and test sets, \( T_R, T_E \), with 50% ratings, respectively. \( T_R, T_E \subset \{(u, i, r_{ui}) \in \mathbb{N} \times \mathbb{N} \times [1, 5]|1 \leq u \leq n, 1 \leq i \leq m\} \). \( T_E \) is kept unchanged, while different (average) number of observed ratings for each user, 4, 8, 12, 16, are randomly sampled from \( T_R \) for training, with different sparsity (\( \sum_{u,i} y_{ui}/n/m \)) levels of 0.2%, 0.4%, 0.6% and 0.8% correspondingly;

3. We randomly pick 40 observed ratings on average from \( T_R \) for each user to construct the auxiliary data matrix \( \tilde{R} \). To simulate heterogenous auxiliary and target data, we adopt a pre-processing approach on \( \tilde{R} \), by relabeling ratings with value \( r_{ui} \leq 3 \) in \( \tilde{R} \) as 0 (dislike), and then ratings with value \( r_{ui} > 3 \) as 1 (like). The overlap between \( \tilde{R} \) and \( R \) (\( \sum_{u,i} y_{ui}/n/m \)) is 0.026%, 0.062%, 0.096% and 0.13% correspondingly.

**Subset of Netflix Data** The Netflix rating data contains more than \( 10^8 \) ratings with values in \( \{1, 2, 3, 4, 5\} \), which are given by more than \( 4.8 \times 10^5 \) users on around \( 1.8 \times 10^4 \) movies. The data set used in the experiments is constructed as follows,

1. We use the target data in our previous work [160], which is a dense 5,000 × 5,000 rating matrix \( R \) from the Netflix data; more specifically, in [160], we first identify 5,000 movies appearing both in MovieLens\(^7\) and Netflix via the movie title, and then select 10,000 most frequent users and another 5,000 most popular items from Netflix, and the 5,000 items used in this chapter are the movies appearing both in MovieLens and Netflix and the 5,000 users used in this chapter are the most frequent 5,000 users.

2. We randomly split \( R \) into training and test sets, \( T_R, T_E \), with 50% ratings, respectively. \( T_E \) is kept unchanged, while different (average) number of observed ratings for each user, 10, 20, 30, 40, are randomly sampled from \( T_R \) for training, with different sparsity levels of 0.2%, 0.4%, 0.6% and 0.8% correspondingly;

3. We randomly pick 100 observed ratings on average from \( T_R \) for each user to construct the auxiliary data matrix \( \tilde{R} \). To simulate heterogenous auxiliary and target

---

\(^7\)http://www.grouplens.org/node/73
data, we adopt the pre-processing approach [192] on \( \tilde{R} \), by relabeling 1, 2, 3 ratings in \( \tilde{R} \) as 0 (dislike), and then 4, 5 ratings as 1 (like). The overlap between \( \tilde{R} \) and \( R \left( \sum_{u,i} y_{ui} \tilde{y}_{ui} / n / m \right) \) is 0.035\%, 0.071\%, 0.11\% and 0.14\% correspondingly.

The final data sets are summarized in Table 4.3.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Form</th>
<th>Sparsity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moviepilot (subset)</td>
<td>target (training)</td>
<td>( [1, 5] \cup {?} )</td>
</tr>
<tr>
<td></td>
<td>target (test)</td>
<td>( [1, 5] \cup {?} )</td>
</tr>
<tr>
<td></td>
<td>auxiliary</td>
<td>( {0, 1, ?} )</td>
</tr>
<tr>
<td>Netflix (subset)</td>
<td>target (training)</td>
<td>{1, 2, 3, 4, 5, ?}</td>
</tr>
<tr>
<td></td>
<td>target (test)</td>
<td>{1, 2, 3, 4, 5, ?}</td>
</tr>
<tr>
<td></td>
<td>auxiliary</td>
<td>{0, 1, ?}</td>
</tr>
</tbody>
</table>

**Evaluation Metrics** We adopt the evaluation metric of Mean Absolute Error (MAE) and Root Mean Square Error (RMSE),

\[
MAE = \sum_{(u, i, r_{ui}) \in T_E} |r_{ui} - \hat{r}_{ui}| / |T_E|
\]

\[
RMSE = \sqrt{\sum_{(u, i, r_{ui}) \in T_E} (r_{ui} - \hat{r}_{ui})^2 / |T_E|}
\]

where \( r_{ui} \) and \( \hat{r}_{ui} \) are the true and predicted ratings, respectively, and \( |T_E| \) is the number of test ratings. In all experiments, we run 3 random trials when generating the required number of observed ratings from \( T_R \), and averaged results are reported.

### 4.4.2 Baselines and Parameter Settings

We compare our TCF method with five non-transfer learning methods: the average filling method (AF), Pearson correlation coefficient (PCC) [172], PMF [177], SVD [181], OptSpace [96], as well as one transfer learning method: CMF [193] with logistic link function (CMF-link).
We study the following six average-filling (AF) methods,

\[
\hat{r}_{ui} = \bar{r}_u,
\]

\[
\hat{r}_{ui} = \bar{r}_i,
\]

\[
\hat{r}_{ui} = (\bar{r}_u + \bar{r}_i)/2,
\]

\[
\hat{r}_{ui} = b_u + \bar{r}_i,
\]

\[
\hat{r}_{ui} = \bar{r}_u + b_i,
\]

\[
\hat{r}_{ui} = \bar{r} + b_u + b_i,
\]

where \(\bar{r}_u = \sum_i y_{ui} r_{ui} / \sum_i y_{ui}\) is the average rating of user \(u\), \(\bar{r}_i = \sum_u y_{ui} r_{ui} / \sum_u y_{ui}\) is the average rating of item \(i\), \(b_u = \sum_i y_{ui} (r_{ui} - \bar{r}_i) / \sum y_{ui}\) is the bias of user \(u\), \(b_i = \sum_u y_{ui} (r_{ui} - \bar{r}_u) / \sum y_{ui}\) is the bias of item \(i\), and \(\bar{r} = \sum_{u,i} y_{ui} r_{ui} / \sum_{u,i} y_{ui}\) is the global average rating. We use \(\hat{r}_{ui} = \bar{r}_u\) and \(\hat{r}_{ui} = \bar{r}_i\).

For PCC, since the data matrices are sparse, we use the whole set of neighboring users in the prediction rule. For PMF, SVD, OptSpace, CMF-link and TCF, we fix the latent feature number \(d = 10\). For PMF, different tradeoff parameters of \(\alpha_u = \alpha_v \in \{0.01, 0.1, 1\}\) are tried; for CMF-link, different tradeoff parameters \(\alpha_u = \alpha_v \in \{0.01, 0.1, 1\}, \lambda \in \{0.01, 0.1, 1\}\) are tried; for CMTF, \(\beta\) is fixed as 1, and different tradeoff parameters \(\alpha_u = \alpha_v \in \{0.01, 0.1, 1\}, \lambda \in \{0.01, 0.1, 1\}\) are tried; for CSVD, different tradeoff parameters \(\lambda \in \{0.01, 0.1, 1\}\) are tried.

To alleviate the data heterogeneity of \(\{0, 1\}\) and \(\{1, 2, 3, 4, 5\} - 1\) or \(\{1, 2, 3, 4, 5\} \otimes 1\), a logistic link function \(\sigma(U_u V_i^T)\) is embedded in the auxiliary data matrix factorization of CMF,

\[
\min_{U,V} \sum_{u=1}^{N} \sum_{i=1}^{M} y_{ui} \left[ \frac{1}{2} (r_{ui} - U_u V_i^T)^2 + \frac{\alpha_u}{2} ||U_u||^2 + \frac{\alpha_v}{2} ||V_i||^2 \right] + \lambda \sum_{u=1}^{N} \sum_{i=1}^{M} \tilde{y}_{ui} \left[ \frac{1}{2} (\tilde{r}_{ui} - \sigma(U_u V_i^T))^2 + \frac{\alpha_u}{2} ||U_u||^2 + \frac{\alpha_v}{2} ||V_i||^2 \right]
\]

where \(\sigma(x) = \frac{1}{1+e^{-\gamma(x-\theta)}}\) (see Figure 4.3) and different parameters \(\gamma \in \{1, 10, 20\}\) are tried.
Table 4.4: Prediction performance of TCF and other methods on the subset of Moviepilot data.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Methods</th>
<th>Sparsity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0.2% (tr. 9, val. 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAE</td>
</tr>
<tr>
<td>MAE</td>
<td>AF</td>
<td>0.7942 ± 0.0047</td>
</tr>
<tr>
<td></td>
<td>AF (user)</td>
<td>0.8269 ± 0.0035</td>
</tr>
<tr>
<td></td>
<td>AF (item)</td>
<td>0.8126 ± 0.0035</td>
</tr>
<tr>
<td></td>
<td>PCC</td>
<td>0.7956 ± 0.0037</td>
</tr>
<tr>
<td></td>
<td>PMF</td>
<td>0.8118 ± 0.0014</td>
</tr>
<tr>
<td></td>
<td>SVD</td>
<td>0.8262 ± 0.0081</td>
</tr>
<tr>
<td></td>
<td>OptSpace</td>
<td>1.3465 ± 0.0352</td>
</tr>
<tr>
<td></td>
<td>CMF-link</td>
<td>0.9956 ± 0.0049</td>
</tr>
<tr>
<td></td>
<td>TCF (CMTF)</td>
<td>0.7415 ± 0.0018</td>
</tr>
<tr>
<td></td>
<td>TCF (CSVD)</td>
<td>0.7087 ± 0.0035</td>
</tr>
</tbody>
</table>

Table 4.5: Prediction performance of TCF and other methods on the subset of Netflix data.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Methods</th>
<th>Sparsity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0.2% (tr. 9, val. 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAE</td>
</tr>
<tr>
<td>MAE</td>
<td>AF</td>
<td>0.7765 ± 0.0006</td>
</tr>
<tr>
<td></td>
<td>AF (user)</td>
<td>0.8060 ± 0.0021</td>
</tr>
<tr>
<td></td>
<td>AF (item)</td>
<td>0.8535 ± 0.0007</td>
</tr>
<tr>
<td></td>
<td>PCC</td>
<td>0.8233 ± 0.0028</td>
</tr>
<tr>
<td></td>
<td>PMF</td>
<td>0.8879 ± 0.0006</td>
</tr>
<tr>
<td></td>
<td>SVD</td>
<td>0.8057 ± 0.0021</td>
</tr>
<tr>
<td></td>
<td>OptSpace</td>
<td>0.8276 ± 0.0004</td>
</tr>
<tr>
<td></td>
<td>CMF-link</td>
<td>0.7994 ± 0.0017</td>
</tr>
<tr>
<td></td>
<td>TCF (CMTF)</td>
<td>0.7589 ± 0.0017</td>
</tr>
<tr>
<td></td>
<td>TCF (CSVD)</td>
<td>0.7405 ± 0.0007</td>
</tr>
</tbody>
</table>

| RMSE   | AF               | 0.9855 ± 0.0004 | 0.9427 ± 0.0007 | 0.9277 ± 0.0006 | 0.9200 ± 0.0002 |
|        | AF (user)        | 1.0208 ± 0.0015 | 0.9921 ± 0.0012 | 0.9834 ± 0.0004 | 0.9791 ± 0.0002 |
|        | AF (item)        | 1.0708 ± 0.0011 | 1.0477 ± 0.0005 | 1.0386 ± 0.0004 | 1.0339 ± 0.0001 |
|        | PCC              | 1.0462 ± 0.0032 | 1.0041 ± 0.0018 | 0.9841 ± 0.0048 | 0.9934 ± 0.0062 |
|        | PMF              | 1.0779 ± 0.0001 | 1.0473 ± 0.0004 | 1.0205 ± 0.0012 | 0.9691 ± 0.0007 |
|        | SVD              | 1.0202 ± 0.0014 | 0.9906 ± 0.0012 | 0.9798 ± 0.0005 | 0.9741 ± 0.0004 |
|        | OptSpace         | 1.0676 ± 0.0020 | 1.0089 ± 0.0024 | 0.9750 ± 0.0010 | 0.9543 ± 0.0037 |
|        | CMF-link         | 1.0204 ± 0.0013 | 0.9552 ± 0.0009 | 0.9369 ± 0.0004 | 0.9277 ± 0.0004 |
|        | TCF (CMTF)       | 0.9653 ± 0.0018 | 0.9171 ± 0.0063 | 0.8971 ± 0.0005 | 0.8884 ± 0.0007 |
|        | TCF (CSVD)       | 0.9502 ± 0.0005 | 0.9074 ± 0.0004 | 0.8903 ± 0.0006 | 0.8809 ± 0.0005 |
4.4.3 Summary of the Experimental Results

We randomly sample \( n \) ratings (one rating per user on average) from the training data \( R \) and use them as the validation set to determine the tradeoff parameters \( (\alpha_u, \alpha_v, \beta, \lambda) \) and the number of iterations to convergence for PMF, OptSpace, CMF-link and TCF. For AF, PCC and SVD, both the training set and validation set are combined as one set of training data. The results on test data (unavailable during training) are reported in Table 4.4 and Table 4.5. We can make the following observations:

1. For the smoothing method of average filling (AF), we can see that the best variant, \( \hat{r}_{ui} = \bar{r}_u + b_u + b_i \), is very competitive for sparse rating data, while the commonly used average filling methods of \( \hat{r}_{ui} = \bar{r}_u \) and \( \hat{r}_{ui} = \bar{r}_i \) is much worse. There are two reasons for the advantages of AF. First, for sparse data, average filling is a very strong baseline, which is also observed in the Netflix competition. Second, PMF shows its advantages especially when the user-item rating matrix is large, e.g. the whole data set used in the Netflix competition, and can be improved if we tune the parameters in finer granularity.

2. For matrix factorization methods with orthonormal constraints including SVD and OptSpace, we can see that SVD is better than OptSpace when the sparsity is lower (e.g. \( \leq 0.6\% \) for MoviePilot and \( \leq 0.4\% \) for Netflix), while OptSpace beats SVD when the rating matrix becomes denser, which can be explained by the different strategies adopted by SVD and OptSpace for missing ratings. SVD fills the missing ratings with average values, which may help for an extreme-
ly sparse rating matrix, but will hurt the performance when the rating matrix becomes denser.

3. For the sparsity problem in collaborative filtering, transfer learning is a very attractive technique:

(a) The proposed transfer learning methods of CMTF and CSVD perform significantly better than all other baselines at all sparsity levels;

(b) For the transfer learning method of CMF-link, we can see that it is significantly better than the non-transfer learning methods of PMF, SVD and OptSpace at almost all sparsity levels (except the extremely sparse case of 0.2% on Moviepilot), but is still worse than AF, which can be explained by the heterogeneity of the auxiliary binary rating data and target numerical rating data, and the usefulness of smoothing (AF) for sparse data;

(c) For the transfer learning methods of CMTF and CMF-link, we can see that CMTF performs better than CMF-link in all cases, which shows the advantages of modeling the data-dependent effect using inner matrices $B$ and $\tilde{B}$ in CMTF.

(d) For the two variants of TCF, we can see that the transfer learning method CSVD further improves the performance over CMTF in all cases, which shows the effect of noise reduction from the orthonormal constraints, $U^T U = I$ and $V^T V = I$.

To further study the effectiveness of selective transfer via noise reduction in TCF, we compare the performance of CMTF and CSVD at different sparsity levels with different auxiliary data of sparsity 1%, 2% and 3% on the subset Netflix data. The results are shown in Figure 4.4. We can see that CSVD performs better than CMTF in all cases, which again shows the advantage of CSVD in transferring the most useful knowledge.

There is a very fundamental question in transfer learning [157], namely when to transfer, which is related to negative transfer [147]. For our problem setting (see Figure 4.1), negative transfer [147] may happen when the density of auxiliary binary ratings is lower than that of target numerical ratings, or the semantic meaning of auxiliary binary ratings are completely different from that of target numerical ratings.
Figure 4.4: Prediction performance of TCF (CMTF, CSVD) on Netflix at different sparsity levels with different auxiliary data.

However, in our work, we assume that the auxiliary binary ratings is denser than the target numerical ratings, and both ratings are related though there are some differences. Thus, under our assumption, negative transfer is not likely to happen. In fact negative transfer is not observed in our empirical studies.

4.5 Discussions

SVD  Low-rank singular value decomposition (SVD) or principal component analysis (PCA) [16, 66] is widely used in information retrieval and data mining to find latent topics [47] and to reduce noise [17]. These solutions have also been applied in collab-
orative filtering [65, 59, 19, 164, 181, 195, 108, 27, 96]. Among them, some works apply non-iterative SVD or PCA on a full matrix after some pre-processing to remove the missing values [65, 59, 19, 164, 181], while other works [195, 108] use iterative SVD on a full matrix in an expectation-maximization (EM) procedure. Still other works [27, 96] take the missing ratings as unknown and directly optimize the objective function over the observed ratings only. Our strategy is similar to that of [27, 96], since we also take missing ratings as unknown. We use two representative methods of SVD [181] and OptSpace [96] as our baselines in the experiments.

The differences of our approach and those previously published SVD-based methods can be identified from two aspects. First, we take missing ratings as unknown, while most previous works pre-process the rating matrix to obtain a full matrix on which PCA or SVD is applied. Second, we make use of some auxiliary data besides the target rating data via transfer learning techniques, while the aforementioned works only have a target rating matrix.

**PMF** PMF [177] is a recently proposed method for missing value prediction in a single matrix, which can be reduced from TCF in Eq.(4.6) when \( \mathcal{D} = \mathcal{D}_R \), \( \lambda = 0 \), \( \beta = 0 \) and \( B = I \). The RSTE model [136] generalizes PMF and factorizes a single rating matrix with a regularization term from the user-side social data, which is different from our two-matrix factorization model. The PLRM model [226] generalizes the PMF model to incorporate numerical ratings, *implicit* purchasing data, meta data and social network information, but does not consider the *explicit* auxiliary data of both like and dislike. Mathematically, the PLRM model only considering numerical ratings and *implicit feedback* can be considered as a special case of our TCF framework, CMTF for \( \mathcal{D} = \mathcal{D}_R \), but the learning algorithm is still different since CMTF has closed-form solutions for all steps.

**CMF** CMF [193] is proposed for jointly factorizing two matrices with the constraints of sharing item-specific latent features, and SoRec [138] is proposed for sharing user-specific latent features. CMF and SoRec can be reduced from TCF in Eq.(4.6) when \( \mathcal{D} = \mathcal{D}_R \), \( \beta = 0 \), \( B = \tilde{B} = I \), and only requiring one-side latent feature matrix to be the same, e.g. user-side of \( R \sim UV^T \), \( \tilde{R} \sim U\tilde{V}^T \), or item-side of \( R \sim UV^T \), \( \tilde{R} \sim \tilde{U}V^T \). However, in our problem setting as shown in Figure 4.1, both users and items are aligned. To alleviate the data heterogeneity in CMF or SoRec, we embed a logistic link function in the auxiliary data matrix factorization in our experiments.
There are at least three differences between TCF and CMF. First, TCF is a trilinear method, \( \mathbf{R} = \mathbf{UBV}^T, \tilde{\mathbf{R}} = \mathbf{UBV}^T \), where the inner matrices \( \mathbf{B} \) and \( \tilde{\mathbf{B}} \) are designed to capture the domain-dependent information, while CMF is a bilinear method and cannot be applied to our studied problem (see Figure 4.1). Second, we introduce orthonormal constraints in one variant of TCF, CSVD, which is empirically proved to be more effective on noise reduction, while CMF does not have such constraints and effect. Finally, the learning algorithms of TCF (CSVD), TCF (CMTF) and CMF are different.

**DPMF** Dependent probabilistic matrix factorization (DPMF) [3] is a multi-task version of PMF based on Gaussian processes, which is proposed for incorporating *homogeneous*, but not *heterogeneous*, side information via sharing the inner covariance matrices of user-specific and item-specific latent features. The slice sampling algorithm used in DPMF may be too time consuming for some medium sized problems, e.g. the problems studied in the experiments.

**CST** Coordinate system transfer (CST) [160] is a recently proposed transfer learning method in collaborative filtering to transfer the coordinate system from two auxiliary CF matrices to a target one in an adaptive way. CST performs quite well when the coordinate system is constructed when the auxiliary data is dense, and when the target data is not very sparse [160]. However, when the auxiliary and target data are not so dense, constructing the shared latent feature matrices in a collective way as used in TCF may perform better, since the collective behavior brings in richer interactions when bridging two data sources [33, 200].

Parallel to the PMF family of CMF and DPMF, there is a corresponding NMF [113] family with non-negative constraints:

1. Trilinear method of WNMCTF [221] is proposed to factorize three matrices of user-item, item-content and user-demographics, and

2. Codebook sharing methods of CBT [115] and RMGM [116] can be considered as adaptive and collective extensions of [190, 49]. RMGM-OT [117] is a follow-up work of RMGM [116], which studies the effect of user preferences over time by sharing the cluster-level rating patterns across temporal domains. This work focused on homogeneous user feedbacks of 5-star grades instead of heterogeneous user feedbacks.

Models in the NMF family usually have better interpretability, e.g. the learned la-
tent feature matrices $U$ and $V$ in CBT [115] and RMGM [116] can be considered as memberships of the corresponding users and items, while the top ranking models [104] in collaborative filtering are from the PMF family. We summarize the above related work in Table 6.1, in the perspective of whether having non-negative constraints on the latent variables, and what & how to transfer in transfer learning [157].

Table 4.6: Summary of TCF and other transfer learning methods in collaborative filtering.

<table>
<thead>
<tr>
<th>Knowledge (what to transfer)</th>
<th>Algorithm style (how to transfer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMF [177] family</td>
<td>Adaptive Collective</td>
</tr>
<tr>
<td>Covariance</td>
<td>DPMF [3]</td>
</tr>
<tr>
<td>Latent features</td>
<td>CST [160] SoRec [138], CMF [193], TCF</td>
</tr>
<tr>
<td>Latent features</td>
<td>CBT [115] RMGM [116]</td>
</tr>
<tr>
<td>NMF [113] family</td>
<td>CST [160] SoRec [138], CMF [193], TCF</td>
</tr>
<tr>
<td>Latent features</td>
<td>WNMCTF [221]</td>
</tr>
</tbody>
</table>

Clustering on Relational Data Long et al. [130, 132] study a clustering problem on a full matrix without missing values, which is different from our problem setting for missing rating prediction, while the idea of sharing common subspace or latent feature matrices is similar to ours. Cohn et al. [39] study document clustering using content information and auxiliary information of document-document link information, while the two matrices of term-document and document-document are both full without missing values. Banerjee et al. [10] study clustering of relational data without missing values or the missing entries are imputed with zeros, while our approach take missing values as unknown and aims for missing rating prediction.

Logistic Loss Function in Matrix Factorization There are some matrix factorization methods using logistic loss functions for binary rating data [40, 67, 184]. There are two reasons why we do not use such loss functions. First, using different loss functions, e.g. the logistic loss function in binary PCA [40, 67, 184], is a vertical research direction to our focus of developing transfer learning solutions, and we will study this issue in our future work. Second, it is difficult to justify using logistic loss function [40, 67, 184] in the factorization of the auxiliary binary rating matrix and square loss function in the target numerical rating matrix, since the objective functions are then totally different, and thus the meanings and scales of the user-specific latent feature matrix $U$ in two domains are not comparable (similar for $V$), which may cause
the difficulty of knowledge sharing.

We illustrate the two loss functions bellow,

\[-[r_{ui} \log \hat{r}_{ui} + (1 - r_{ui}) \log(1 - \hat{r}_{ui})] \quad \text{vs.} \quad (r_{ui} - U_u V_i^T)^2\]

where \(r_{ui} \in \{0, 1\}\) is the true binary rating, \(\hat{r}_{ui} = \sigma(U_u V_i^T) \in [0, 1]\) is the predicted rating, and \(\sigma(\theta) = \frac{1}{1 + \exp(-\theta)}\) is the sigmoid function (or logistic link function).

Furthermore, to address the heterogeneities of numerical ratings and binary ratings, we have scaled the 5-star numerical ratings to the range of \([0, 1]\) and then introduced a sigmoid link function (or logistic link function) instead of logistic loss function as follows (see Section 4.4),

\[(r_{ui} - \hat{r}_{ui})^2 \quad \text{vs.} \quad (r_{ui} - U_u V_i^T)^2\]

where \(\hat{r}_{ui} = \sigma(U_u V_i^T) \in [0, 1]\) is the predicted rating.

To sum up, the differences between our proposed transfer learning solution and other works include the following. First, we focus on missing rating prediction instead of clustering [130]. Second, we study auxiliary data of user feedbacks instead of content information [193]. Third, we leverage auxiliary data from frontal side instead of user side [29] or item side [193]. Fourth, we take missing ratings as unknown instead of negative feedbacks of zeros [10] in order to optimize the objective function specifically on the observed ratings only. Fifth, we introduce orthonormal constraints instead of non-negative constraints [221] to resemble the effect of noise reduction. Sixth, we design a collective algorithm instead of an adaptive algorithm for richer interactions between the auxiliary domain and the target domain [33, 200]. Seventh, we transfer knowledge of latent features among all aligned users and items instead of sharing only compressed knowledge of cluster-level rating patterns [115, 116] or covariance matrix [3]. Finally, we extend a trilinear base model instead of a bilinear model [193] to capture both domain-independent knowledge and domain-dependent effect. In summary, the first three points illustrate the novelty of our proposed problem setting, and the next six points show the novelty of our designed algorithm.
4.6 Summary

In this chapter, we investigate how to address the sparsity problem in collaborative filtering via a transfer learning solution. Specifically, we present a novel transfer learning framework of transfer by collective factorization, to transfer knowledge from auxiliary data of explicit binary ratings (like and dislike), which alleviates the data sparsity problem in the target numerical ratings. Our method constructs the shared latent space $U, V$ in a collective manner, captures the data-dependent effect via learning inner matrices $B, \tilde{B}$ separately, and selectively transfer the most useful knowledge via noise reduction by introducing orthonormal constraints. The novelty of our algorithm includes generalizing transfer learning methods in collaborative filtering in a principled way. Experimental results show that TCF performs significantly better than several state-of-the-art baseline algorithms at various sparsity levels.

The problem setting of TCF (Figure 4.1) for heterogeneous explicit user feedbacks is novel and widely applicable in many applications beyond the user-item representation in recommender systems. Examples include query-document in information retrieval, author-word in academic publications, user-community in social network services [230], location-activity in ubiquitous computing [234], and even drug-protein in biomedicine, etc.

For our future work, we will study and extend the transfer learning framework in additional areas and to include more theoretical analysis and larger-scale experiments. E.g. we will address “pure” cold-start recommendation problem for users without any rating, sparse learning and matrix completion [96], partial correspondence between users and items [118], distributed implementation on the Map/Reduce framework [216], adaptive transfer learning [30] in collaborative filtering, more complex user feedbacks of different rating distributions, and different loss functions [40, 67], etc.
CHAPTER 5

TRANSFER LEARNING IN COLLABORATIVE FILTERING WITH FRONTAL-SIDE UNCERTAIN RATINGS

5.1 Introduction

Recently, researchers have developed new methods for collaborative filtering [64, 102, 167]. A new direction is to apply transfer learning to collaborative filtering [116, 159], so that one can make use of auxiliary data to help improve the rating prediction performance. However, in many industrial applications, precise point-wise user feedbacks may be rare, because many users are unwilling or unlikely to express their preferences accurately. Instead, we may obtain estimates of a user’s tastes on an item based on the user's additional behavior or social connections. For example, suppose that a person Peter is watching a 10-minute video. Suppose that Peter stops watching the video after the first 3 minutes. In this case, we may estimate that Peter’s preference on the movie is in the range of 1 to 2 stars with a uniform distribution. As another example in social media, suppose that Peter reads his followees’ posts in a microblog about a certain movie\(^1\). Suppose that his followee John posts a comment on the movie with 3 stars. In addition, Peter’s other followees Bob gives 4 stars, and Alice gives 5 stars. Then, with this social impression data, we should be able to obtain a potential rating distribution for Peter’s preference on the movie. We call such a rating distribution as an uncertain rating, since it represents a rating spectrum involving uncertainty instead of an accurate point-wise score.

5.2 Collaborative Filtering with Uncertain Ratings

5.2.1 Problem Definition

In our problem setting, we have a target user-item numerical rating matrix \( R = [r_{ui}]_{n \times m} \in \{1, 2, 3, 4, 5, ?\}^{n \times m}, \) where the question mark “?” denotes a missing val-

\(^1\)For example, Tencent Video http://v.qq.com/ and Tencent Weibo (microblog) http://t.qq.com/ are connected.
Figure 5.1: Illustration of transfer learning in collaborative filtering from auxiliary uncertain ratings (left: target 5-star numerical ratings; right: auxiliary uncertain ratings represented as ranges or rating distributions). Note that there is a one-one mapping between the users and items from two data.

We use an indicator matrix $Y = [y_{ui}]_{n \times m} \in \{0, 1\}^{n \times m}$ to denote whether the entry $(u, i)$ is observed ($y_{ui} = 1$) or not ($y_{ui} = 0$), and $\sum_{u,i} y_{ui} = q$. Besides the target data, we have an auxiliary user-item uncertain rating matrix $\tilde{R} = [\tilde{r}_{ui}]_{n \times m} \in \{[a_{ui}, b_{ui}], ?\}^{n \times m}$ with $\tilde{q}$ observations, where the entry $[a_{ui}, b_{ui}]$ denotes the range of a certain distribution for the corresponding rating located at $(u, i)$, where $a_{ui} \leq b_{ui}$. The question mark "?" represents a missing value. Similar to the target data, we have a corresponding indicator matrix $\tilde{Y} = [\tilde{y}_{ui}]_{n \times m} \in \{0, 1\}^{n \times m}$ with $\sum_{u,i} \tilde{y}_{ui} = \tilde{q}$. We also assume that there is a one-one mapping between the users and items of $R$ and $\tilde{R}$.

Our goal is to predict the missing values in $R$ by exploiting uncertain ratings in $\tilde{R}$.

The difference between the problem setting studied in this chapter and those of previous works like [159] is that the auxiliary data in this chapter are uncertain and represented as ranges of rating distributions instead of accurate point-wise scores. We illustrate the new problem setting in Figure 5.1.

5.2.2 Challenges

To leverage such uncertain ratings as described above, we plan to exploit techniques in transfer learning [157]. To do this, we have to answer two fundamental questions: “what to transfer” and “how to transfer” in transfer learning [157]. In particular, we have to decide

1. what knowledge to extract and transfer from the auxiliary uncertain ratings, and
2. how to model the knowledge transfer from the auxiliary uncertain rating data to the target numerical ratings in a principled way.
As far as we know, there has not been existing research work on this problem.

Several existing works are relevant to ours. Transfer learning approaches are proposed to transfer knowledge in latent feature space [193, 221, 160, 29, 159, 207], exploiting feature covariance [3] or compressed rating patterns [115, 116]. In collaborative filtering, transfer learning methods can be adaptive [115, 160] or collective [193, 116, 221, 29, 159, 207]. Other works, such as that by Ma et al. [136], tend to use auxiliary social relations and extend the rating generation function in a model-based collaborative filtering method [177]. Zhang et al. [225] generate point-wise virtual ratings from sentimental polarities of users’ reviews on items, which are then used in memory-based collaborative filtering methods for video recommendation. However, these works do not address the uncertain rating problem.

5.2.3 Overview of Our Solution

In this chapter, we develop a novel approach known as TIF (transfer by integrative factorization) to transfer auxiliary data consisting of uncertain ratings as constraints to improve the predictive performance in a target collaborative filtering problem. We assume that the users and items can be mapped in a one-one manner. Our approach runs in several steps. First, we integrate (“how to transfer”) the auxiliary uncertain ratings as constraints (“what to transfer”) into the target matrix factorization problem. Second, we learn an expected rating for each uncertain rating automatically. Third, we relax the constraints and introduce a penalty term for those violating the constraints. Finally, we solve the optimization problem via stochastic gradient descent (SGD). We conduct empirical studies on two movie recommendation data sets of MovieLens10M and Netflix, and obtain significantly better results of TIF over other methods.

5.3 Transfer by Integrative Factorization

5.3.1 Model Formulation

Koren [102] proposes to learn not only user-specific latent features $U_u \in \mathbb{R}^{1 \times d}$ and item-specific latent features $V_i \in \mathbb{R}^{1 \times d}$ as that in PMF [177], but also user bias $b_u \in \mathbb{R}$, item bias $b_i \in \mathbb{R}$ and global average rating value $\mu \in \mathbb{R}$. The objective function of
RSVD [102] is as follows,

\[
\min_{U, V, b, \mu} \sum_{u=1}^{n} \sum_{i=1}^{m} y_{ui} (E_{ui} + R_{ui})
\]

where \(E_{ui} = \frac{1}{2} (r_{ui} - \hat{r}_{ui})^2\) is the square loss function with \(\hat{r}_{ui} = \mu + b_u + b_i + U_u V_i^T\) as the predicted rating, and \(R_{ui} = \frac{\alpha_u}{2} \| U_u \|^2 + \frac{\alpha_v}{2} \| V_i \|^2 + \frac{\beta_u}{2} b_u^2 + \frac{\beta_v}{2} b_i^2\) is the regularization term used to avoid overfitting. To learn the parameters \(U, V, b, \mu\) efficiently, SGD algorithms are adopted, in which the parameters are updated for each randomly sampled rating \(r_{ui}\) with \(y_{ui} = 1\).

In our problem setting, besides the target numerical ratings \(R\), we have some auxiliary uncertain ratings represented as ranges of rating distributions \(\tilde{R} \in \{ [a_{ui}, b_{ui}], \}^{n \times m}\). The semantic meaning of \([a_{ui}, b_{ui}]\) can be represented as a constraint for the predicted rating \(\hat{r}_{ui} \in \mathcal{C}(a_{ui}, b_{ui})\), e.g., \(\hat{r}_{ui} = (a_{ui} + b_{ui})/2\) or \(a_{ui} \leq \hat{r}_{ui} \leq b_{ui}\). Based on this observation, we extend the optimization problem [102] as shown in Eq.(5.1), and propose to solve the following optimization problem,

\[
\min_{U, V, b, \mu} \sum_{u=1}^{n} \sum_{i=1}^{m} y_{ui} (E_{ui} + R_{ui})
\]

\[\text{s.t.} \quad \hat{r}_{ui} \in \mathcal{C}(a_{ui}, b_{ui}), \]
\[\forall y_{ui} = 1, u = 1, \ldots, n, i = 1, \ldots, m\]

where the auxiliary domain knowledge involving uncertain ratings is transferred to the target domain, via integration of constraints into the target matrix factorization problem: \(\hat{r}_{ui} \in \mathcal{C}(a_{ui}, b_{ui}), \tilde{y}_{ui} = 1\). For this reason, we call our approach transfer by integrative factorization (TIF). The knowledge, \(\mathcal{C}(a_{ui}, b_{ui})\), from the auxiliary uncertain ratings can be considered as a rating spectrum with lower bound value of \(a_{ui}\) and upper bound value of \(b_{ui}\), which can be equivalently represented as a rating distribution of \(r \sim P_{ui}(r)\) over \([a_{ui}, b_{ui}]\).

The optimization problem with a hard constraint \(\hat{r}_{ui} \in \mathcal{C}(a_{ui}, b_{ui})\) as shown in Eq.(5.2) is difficult to solve. We relax this hard constraint, move it to the objective function, and derive the following new objective function with an additional penalty term,

\[
\min_{U, V, b, \mu} \sum_{u=1}^{n} \sum_{i=1}^{m} [y_{ui} (E_{ui} + R_{ui}) + \lambda \tilde{y}_{ui} (\tilde{E}_{ui} + \tilde{R}_{ui})]
\]
where $\tilde{E}_{ui}$ includes the predicted rating $\hat{r}_{ui}$ and the observed uncertain rating $[a_{ui}, b_{ui}]$. The tradeoff parameter $\lambda$ is used to balance two loss functions for target data and auxiliary data. We use the same regularization terms $\tilde{R}_{ui} = R_{ui}$ for simplicity. We now show that the distribution $r \sim P_{ui}(r)$ in $\tilde{E}_{ui}$ can be simplified as an expected rating value.

**Theorem 4.** The penalty term $\tilde{E}_{ui}$ over the rating spectrum $[a_{ui}, b_{ui}]$ can be equivalently represented as $\frac{1}{2}(\bar{r}_{ui} - \hat{r}_{ui})^2$, where $\bar{r}_{ui} = \int_{a_{ui}}^{b_{ui}} P_{ui}(r) \cdot r \, dr$ is the expected rating of user $u$ on item $i$.

**Proof.** Similar to the square loss used in RSVD [102], the penalty over rating spectrum $[a_{ui}, b_{ui}]$ can be written as $\tilde{E}_{ui} = \frac{1}{2} \int_{a_{ui}}^{b_{ui}} [P_{ui}(r) \cdot (r - \hat{r}_{ui})^2] \, dr$, where $P_{ui}(r)$ is the probability of rating value $r$ by user $u$ on item $i$. We thus have the gradient formula:

$$
\frac{\partial \tilde{E}_{ui}}{\partial \hat{r}_{ui}} = \frac{\partial \frac{1}{2} \int_{a_{ui}}^{b_{ui}} [P_{ui}(r) \cdot (r - \hat{r}_{ui})^2] \, dr}{\partial \hat{r}_{ui}}
= -\left( \int_{a_{ui}}^{b_{ui}} P_{ui}(r) \cdot r \, dr - \hat{r}_{ui} \int_{a_{ui}}^{b_{ui}} P_{ui}(r) \, dr \right)
= \frac{\partial \frac{1}{2} \left( \int_{a_{ui}}^{b_{ui}} P_{ui}(r) \cdot r \, dr - \hat{r}_{ui} \right)^2}{\partial \hat{r}_{ui}}
= \frac{\partial \frac{1}{2} (\bar{r}_{ui} - \hat{r}_{ui})^2}{\partial \hat{r}_{ui}},
$$

which shows that we can use the expected rating $\bar{r}_{ui}$ to replace the rating distribution $r \sim P_{ui}(r)$ over $[a_{ui}, b_{ui}]$ since it results in the exactly the same gradient. Hence, parameters learned using the same gradient in the widely used SGD algorithm framework in matrix factorization [102] will be the same. 

However, we still find it difficult to obtain an accurate rating distribution $r \sim P_{ui}(r)$ or the expected rating $\bar{r}_{ui}$, because there is not sufficient information besides a rating range $[a_{ui}, b_{ui}]$. One simple approach is to assign the same weight on $a_{ui}$ and $b_{ui}$, that is $\bar{r}_{ui} = \frac{1}{2}(a_{ui} + b_{ui})$. But such a straightforward approach may not accurately reflect the true expected rating value. Furthermore, static expected value may not well reflect personalized taste. Instead, we learn the expected rating value automatically,

$$
\bar{r}_{ui} = \frac{s(a_{ui})a_{ui} + s(b_{ui})b_{ui}}{s(a_{ui}) + s(b_{ui})}, \tag{5.4}
$$
where \( s(x) = \exp(-|\hat{r}_{ui} - x|^{1-\rho}) \) is a similarity function, and \( s(a_{ui}) / [s(a_{ui}) + s(b_{ui})] \) is the normalized weight or confidence on rating \( a_{ui} \). The parameter \( \rho \) can be considered as an uncertainty factor, where a larger value means higher uncertainty. At the start of the learning procedure, we are uncertain of the expected rating, and thus we may set \( \rho = 1 \) and \( \bar{r}_{ui} = (a_{ui} + b_{ui})/2 \). In the middle of the learning procedure, we may gradually decrease the value of \( \rho \) as we are more sure of the expected rating. Note that the similarity function \( s(x) \) in Eq.(5.4) can be other forms if we have additional domain knowledge. We illustrate the impact of \( \rho \) when we estimate the expected rating in Figure 5.2 \((a_{ui} = 4, b_{ui} = 5)\).

Figure 5.2: Illustration of the expected rating estimated using Eq.(5.4) with \( a_{ui} = 4 \) and \( b_{ui} = 5 \).

### 5.3.2 Learning the TIF

Denoting \( f_{ui} = y_{ui}(\bar{E}_{ui} + \bar{R}_{ui}) + \lambda \tilde{y}_{ui}(\tilde{E}_{ui} + \tilde{R}_{ui}) \) as part of the objective function in Eq.(5.3), we have the gradients \( \nabla U_u = \frac{\partial f_{ui}}{\partial U_u} \), \( \nabla V_i = \frac{\partial f_{ui}}{\partial V_i} \), \( \nabla b_u = \frac{\partial f_{ui}}{\partial b_u} \), \( \nabla b_i = \frac{\partial f_{ui}}{\partial b_i} \), \( \nabla \mu = \frac{\partial f_{ui}}{\partial \mu} \) as follows,
\[ \nabla U_u = \begin{cases} -\varepsilon_{ui}V_i + \alpha_u U_u, & \text{if } y_{ui} = 1 \\ -\lambda\varepsilon_{ui}V_i + \lambda\alpha_u U_u, & \text{if } \bar{y}_{ui} = 1 \end{cases} \tag{5.5} \]

\[ \nabla V_i = \begin{cases} -\varepsilon_{ui}U_u + \alpha_v V_i, & \text{if } y_{ui} = 1 \\ -\lambda\varepsilon_{ui}U_u + \lambda\alpha_v V_i, & \text{if } \bar{y}_{ui} = 1 \end{cases} \tag{5.6} \]

\[ \nabla b_u = \begin{cases} -\varepsilon_{ui} + \beta_u b_u, & \text{if } y_{ui} = 1 \\ -\lambda\varepsilon_{ui} + \lambda\beta_u b_u, & \text{if } \bar{y}_{ui} = 1 \end{cases} \tag{5.7} \]

\[ \nabla b_i = \begin{cases} -\varepsilon_{ui} + \beta_v b_i, & \text{if } y_{ui} = 1 \\ -\lambda\varepsilon_{ui} + \lambda\beta_v b_i, & \text{if } \bar{y}_{ui} = 1 \end{cases} \tag{5.8} \]

\[ \nabla \mu = \begin{cases} -\varepsilon_{ui}, & \text{if } y_{ui} = 1 \\ -\lambda\varepsilon_{ui}, & \text{if } \bar{y}_{ui} = 1 \end{cases} \tag{5.9} \]

where \( \varepsilon_{ui} = r_{ui} - \hat{r}_{ui}, \) \( \bar{r}_{ui} = r_{ui} - \hat{r}_{ui} \) are the errors according to the target numerical rating and the auxiliary expected rating, respectively, and \( \bar{r}_{ui} \) is estimated via Eq.(5.4) using the parameters learned in the previous iteration. We thus have the update rules used in the SGD algorithm framework,

\[ U_u = U_u - \gamma \nabla U_u \]  
\[ V_i = V_i - \gamma \nabla V_i \]  
\[ b_u = b_u - \gamma \nabla b_u \]  
\[ b_i = b_i - \gamma \nabla b_i \]  
\[ \mu = \mu - \gamma \nabla \mu. \]

When there are no auxiliary uncertain ratings, our update rules in Eq.(5.10-5.14) reduce to that of RSVD [102].

Finally, we obtain a complete algorithm as shown in Figure 5.3, where we update the parameters \( U_u, V_i, b_u, b_i \) and \( \mu \) for each observed rating. Note that the stochastic gradient descent algorithm used in RSVD [102] is different from ours, since we have auxiliary uncertain ratings, and learn and transfer the expected ratings \( \bar{r}_{ui} \). TIF inherits the advantages of efficiency in RSVD, and reduces to RSVD when there are only target 5-star numerical ratings. The time complexity of TIF is \( O(T(q + \tilde{q})d) \), where \( T \) represents the number of scans over the whole data and is usually smaller than 100, \( q + \tilde{q} \) denotes the number of observed ratings from both target and auxiliary data, and
Input: The target user-item numerical rating matrix \( R \), the frontal-side auxiliary user-item uncertain rating matrix \( \tilde{R} \).

Output: The user-specific latent feature vector \( u \) and bias \( b_u \), the item-specific latent feature vector \( v \) and bias \( b_v \), the global average \( \mu \), where \( u = 1, \ldots, n \), \( i = 1, \ldots, m \).

For \( t = 1, \ldots, T \)
  For \( \text{iter} = 1, \ldots, q + \tilde{q} \)
    Step 1. Randomly pick up a rating from \( R \) or \( \tilde{R} \);
    Step 2. If \( \tilde{y}_{ui} = 1 \), estimate the expected rating \( \tilde{r}_{ui} \) as shown in Eq.(5.4);
    Step 3. Calculate the gradients as shown in Eq.(5.5-5.9);
    Step 4. Update the parameters as shown in Eq.(5.10-5.14).
  End
End
Decrease the learning rate \( \gamma \) and uncertainty factor \( \rho \).

Figure 5.3: The algorithm of transfer by integrative factorization (TIF).

\( d \) is the number of latent dimensions. Similar to RSVD, TIF can also be implemented in a distributed platform like Map/Reduce.

5.4 Experimental Results

In this section, we plan to evaluate the effectiveness of the TIF algorithm and compare it with some well known benchmark approaches. We start by describing the experimental data.

5.4.1 Data Sets and Evaluation Metrics

MovieLens10M Data (ML) The MovieLens\(^2\) rating data contains more than 10\(^7\) ratings with values in \{0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\}, which are given by more than 7.1 \( \times \) 10\(^4\) users on around 1.1 \( \times \) 10\(^4\) movies between 1995 and 2009. We preprocess the MovieLens data as follows: first, we randomly permute the rating records since the original data is ordered with user ID; second, we use the official linux shell script\(^3\) to generate 5 copies of training data and test data, where in each copy 4/5 are used for training and 1/5 for test; third, for each copy of training data, we take 50% ratings as auxiliary data, and the remaining 50% ratings as target data; fourth, for each copy of auxiliary data, we convert ratings of 0.5, 1, 1.5, 2, 3, 3.5 to uncertain ratings.

\( ^2\)http://www.grouplens.org/node/73/

\( ^3\)http://www.grouplens.org/system/files/ml-10m-README.html
with uniform distribution, and ratings of 4, 4.5, 5 to [4, 5]. Note that assuming the uniform distribution with lower bound of \( a_{ui} \) and upper bound of \( b_{ui} \) mainly affects the initial value of the expected rating, instead of the finally learned expected rating value or learned model parameters when the TIF algorithm converges, since the TIF algorithm has the ability to learn the expected ratings automatically as shown in Eq. (5.4).

**Netflix Data (NF)** The Netflix rating data contains more than \( 10^8 \) ratings with values in \{1, 2, 3, 4, 5\}, which are given by more than \( 4.8 \times 10^5 \) users on around \( 1.8 \times 10^4 \) movies between 1998 and 2005. The Netflix competition data contains two sets, the training set and the probe set, and we randomly separate the training set into two parts, 50\% ratings are taken as auxiliary data, and the remaining 50\% ratings as target data. For the auxiliary data, to simulate the effect of rating uncertainty, we convert ratings of 1, 2, 3 to \([1, 3]\), and ratings of 4, 5 to \([4, 5]\). We randomly generate the auxiliary data and target data for three times, and thus get three copies of data.

We summarize the final data in Table 5.1.

Table 5.1: Description of MovieLens10M data (\( n = 71,567, m = 10,681 \)) and Netflix data (\( n = 480,189, m = 17,770 \)). Sparsity refers to the percentage of observed ratings in the user-item preference matrix, e.g. \( \frac{q_{nm}}{\hat{q}_{nm}} \) and \( \frac{\tilde{q}_{nm}}{\hat{q}_{nm}} \) are sparsities for target data and auxiliary data, respectively.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Form</th>
<th>Sparsity</th>
</tr>
</thead>
<tbody>
<tr>
<td>MovieLens10M</td>
<td>target (training) {0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, ?}</td>
<td>0.52%</td>
</tr>
<tr>
<td></td>
<td>target (test) {0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, ?}</td>
<td>0.26%</td>
</tr>
<tr>
<td></td>
<td>auxiliary {0.5, 3.5, [4, 5], ?}</td>
<td>0.52%</td>
</tr>
<tr>
<td>Netflix</td>
<td>target (training) {1, 2, 3, 4, 5, ?}</td>
<td>0.58%</td>
</tr>
<tr>
<td></td>
<td>target (test) {1, 2, 3, 4, 5, ?}</td>
<td>0.017%</td>
</tr>
<tr>
<td></td>
<td>auxiliary {[1, 3], [4, 5], ?}</td>
<td>0.58%</td>
</tr>
</tbody>
</table>

**Evaluation Metrics** We adopt two evaluation metrics: Mean Absolute Error (MAE) and Root Mean Square Error (RMSE),

\[
MAE = \sum_{(u, i, r_{ui}) \in T_E} |r_{ui} - \hat{r}_{ui}| / |T_E|
\]

\[
RMSE = \sqrt{\sum_{(u, i, r_{ui}) \in T_E} (r_{ui} - \hat{r}_{ui})^2 / |T_E|}
\]

\(^4\)http://www.netflix.com/
where $r_{ui}$ and $\hat{r}_{ui}$ are the true and predicted ratings, respectively, and $|T_E|$ is the number of test ratings.

5.4.2 Baselines and Parameter Settings

We compare our TIF method with a state-of-the-art method in Netflix competition, RSVD [102]. For both TIF and RSVD, we use the same statistics of target training data only to initialize the global average rating value $\mu$, user bias $b_u$, item bias $b_i$, user-specific latent feature vector $U_u$, and item-specific latent feature vector $V_i$,

$$\mu = \frac{n}{n} \sum_{u=1}^{n} \sum_{i=1}^{m} y_{ui} r_{ui} / \sum_{u=1}^{n} \sum_{i=1}^{m} y_{ui}$$

$$b_u = \frac{m}{m} \sum_{i=1}^{m} y_{ui} (r_{ui} - \mu) / \sum_{i=1}^{m} y_{ui}$$

$$b_i = \frac{n}{n} \sum_{u=1}^{n} y_{ui} (r_{ui} - \mu) / \sum_{u=1}^{n} y_{ui}$$

$$U_{uk} = (r - 0.5) \times 0.01, k = 1, \ldots, d$$

$$V_{ik} = (r - 0.5) \times 0.01, k = 1, \ldots, d$$

where $r (0 \leq r < 1)$ is a random value.

For both TIF and RSVD, the tradeoff parameters and learning rate are set similarly to that of RSVD [102], $\alpha_u = \alpha_v = 0.01$, $\beta_u = \beta_v = 0.01$, $\gamma = 0.01$. Note that the value of learning rate $\gamma$ decreases after each scan of the whole rating data [102], $\gamma \leftarrow \gamma \times 0.9$. For MovieLens10M data, we set the number of latent dimensions as $d = 20$ [237]; and for Netflix data, we use $d = 100$ [102]. For TIF, we first fix $\lambda = 1$ when comparing to RSVD, and later study the effect of $\lambda$ with different values of $\lambda \in \{0.1, 0.5, 1\}$.

To study the effectiveness of learning an expected rating for each uncertain rating, we also report the result of using static average rating $\bar{r}_{ui} = (a_{ui} + b_{ui})/2$ with $\tilde{y}_{ui} = 1$, which is denoted as TIF(avg.).

The uncertainty factor $\rho$ in TIF is decreased in a similar way as that of the learning rate $\gamma$, which is updated after every 10 scans of the whole data, $\rho \leftarrow \rho \times 0.9$. 

101
5.4.3 Summary of Experimental Results

The prediction performance of RSVD, TIF(avg.) and TIF are shown in Table 5.2 and 5.3. We can have the following observations,

1. TIF is significantly better than TIF(avg.) and RSVD in both data sets, which clearly shows the advantage of the proposed transfer learning approach in leveraging auxiliary uncertain ratings; and

2. for TIF, the parameter $\lambda$ is important, since it determines how large impact will the auxiliary uncertain data make on the target data. TIF with $\lambda = 0.5$ or $\lambda = 1$ is much better than that of $\lambda = 0.1$, which shows that a medium value between 0.5 and 1 is likely to have the best result.

To gain a deep understanding of the performance of RSVD, TIF(avg.) and TIF, we show the prediction performance against different iteration numbers in Figure 5.4, from which we can have the following observations,

1. For RSVD, TIF(avg.) and TIF, the prediction performance becomes relatively stable after 50 iterations; and

2. TIF performs better than RSVD and TIF(avg.) after 20 iterations in both data sets, which again shows the advantages of the proposed transfer learning approach with the ability of leveraging auxiliary uncertain ratings.

We further study the prediction performance on different user groups with respect to the users’ activeness. For MovieLens10M data, we categorize the users in the test data into 10 groups, where users in different groups have different numbers of ratings. Users in training and test data have similar activeness, according to the data generation procedure. From the results as shown in Figure 5.5, we can see,

1. TIF performs best on all user groups; and

2. TIF(avg.) and TIF are more useful for users with fewer ratings, which shows the effect of sparsity reduction of transfer learning methods in collaborative filtering.

Note that the results of MAE and RMSE in Figure 5.5 is calculated over rating instances of users in the same group.
Table 5.2: Prediction performance of RSVD, TIF(avg.) and TIF on MovieLens10M data (ML) and Netflix data (NF). The tradeoff parameter $\lambda$ is fixed as 1, and the number of iterations is fixed as 50.

<table>
<thead>
<tr>
<th>Data</th>
<th>Metric</th>
<th>RSVD</th>
<th>TIF(avg.)</th>
<th>TIF</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML</td>
<td>MAE</td>
<td>0.6438±0.0011</td>
<td>0.6415±0.0008</td>
<td><strong>0.6242±0.0006</strong></td>
</tr>
<tr>
<td></td>
<td>RMSE</td>
<td>0.8364±0.0012</td>
<td>0.8188±0.0009</td>
<td><strong>0.8057±0.0007</strong></td>
</tr>
<tr>
<td>NF</td>
<td>MAE</td>
<td>0.7274±0.0005</td>
<td>0.7285±0.0002</td>
<td><strong>0.7225±0.0004</strong></td>
</tr>
<tr>
<td></td>
<td>RMSE</td>
<td>0.9456±0.0003</td>
<td>0.9323±0.0002</td>
<td><strong>0.9271±0.0002</strong></td>
</tr>
</tbody>
</table>

Table 5.3: Prediction performance of TIF on MovieLens10M data (ML) and Netflix data (NF) with different value of $\lambda$. The number of iterations is fixed as 50.

<table>
<thead>
<tr>
<th>Data</th>
<th>Metric</th>
<th>$\lambda = 0.1$</th>
<th>$\lambda = 0.5$</th>
<th>$\lambda = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML</td>
<td>MAE</td>
<td>0.6399±0.0003</td>
<td>0.6280±0.0007</td>
<td><strong>0.6242±0.0006</strong></td>
</tr>
<tr>
<td></td>
<td>RMSE</td>
<td>0.8307±0.0008</td>
<td>0.8131±0.0007</td>
<td><strong>0.8057±0.0007</strong></td>
</tr>
<tr>
<td>NF</td>
<td>MAE</td>
<td>0.7233±0.0006</td>
<td><strong>0.7172±0.0004</strong></td>
<td>0.7225±0.0004</td>
</tr>
<tr>
<td></td>
<td>RMSE</td>
<td>0.9377±0.0003</td>
<td><strong>0.9242±0.0002</strong></td>
<td>0.9271±0.0002</td>
</tr>
</tbody>
</table>

Figure 5.4: Prediction performance of RSVD, TIF(avg.) and TIF with different iteration numbers (the tradeoff parameter $\lambda$ is fixed as 1).

5.5 Discussions

Collaborative Filtering  Collaborative filtering [64, 102, 167] as an intelligent component in recommender systems [125, 235] has gained extensive interest in both a-
Figure 5.5: Prediction performance of RSVD, TIF(avg.) and TIF on different user groups (using the first fold of the MovieLens10M data). The tradeoff parameter $\lambda$ is fixed as $1$, and the number of iterations is fixed as $50$.

Table 5.4: Overview of TIF in a big picture of traditional transfer learning and transfer learning in collaborative filtering.

<table>
<thead>
<tr>
<th>Transfer learning approaches</th>
<th>Text classification</th>
<th>Collaborative filtering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model-based Transfer</td>
<td>MTL [57]</td>
<td>CBT, RMGM: cluster-level rating patterns</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DPMF: covariance matrix (operator)</td>
</tr>
<tr>
<td>Feature-based Transfer</td>
<td>TCA [156]</td>
<td>CST, CMF, WNMCTF: latent features</td>
</tr>
<tr>
<td>Instance-based Transfer</td>
<td>TrAdaBoost [44]</td>
<td>TIF: rating instances</td>
</tr>
</tbody>
</table>

Table 5.5: Summary of TIF and other transfer learning methods in collaborative filtering.

<table>
<thead>
<tr>
<th>Knowledge (what to transfer)</th>
<th>Algorithm style (how to transfer)</th>
<th>PMF family</th>
<th>NMF family</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covariance</td>
<td>Adaptive (PMF)</td>
<td>DPMF [3]</td>
<td>CBT [115]</td>
</tr>
<tr>
<td>Latent features</td>
<td>Collective (CMF)</td>
<td>CMF [193]</td>
<td>RMGM [116]</td>
</tr>
<tr>
<td>Constraints</td>
<td>Integrative (TIF)</td>
<td>TIF</td>
<td>WNMCTF [221]</td>
</tr>
<tr>
<td>Codebook</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latent features</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

cademia and industry, while most previous works can only make use of point-wise ratings. In this chapter, we go one step beyond and study a new problem with uncertain ratings via transfer learning techniques, as shown in Figure 5.1.
**Transfer Learning** Transfer learning [32, 157] as a new learning paradigm extracts and transfers knowledge from auxiliary data to help a target learning task [57, 44, 156]. From the perspective of model-based transfer, feature-based transfer and instance-based transfer [157], TIF can be considered as a *rating instance*-based transfer. We make a link between traditional transfer learning methods in text classification and transfer learning methods in collaborative filtering from a unified view, which is shown in Table 7.2.

**Transfer Learning in Collaborative Filtering** There are some related work of transfer learning in collaborative filtering, CMF [193], CBT [115], RMGM [116], WNMCFTF [221], CST [160], DPMF [3], etc. Please see [159] for a detailed analysis and comparison from the perspective of “what to transfer” and “how to transfer” in transfer learning [157].

Comparing to previous works on transfer learning in collaborative filtering, we can categorize TIF as an *integrative* style algorithm (“how to transfer”) via transferring knowledge of constraints (“what to transfer”). We thus summarize the related work as discussed in [159] and our TIF method in Table 5.5, where we can see that TIF extends previous works from two dimensions, “what to transfer” and “how to transfer” in transfer learning [157].

5.6 **Summary**

In this chapter, we study a new problem of transfer learning in collaborative filtering when the auxiliary data are uncertain ratings. We propose a novel efficient transfer learning approach, *transfer by integrative factorization* (TIF), to leverage auxiliary data of uncertain ratings represented as rating distributions. In TIF, we take the auxiliary uncertain ratings as constraints and integrate them into the optimization problem for the target matrix factorization. We then reformulate the optimization problem by relaxing the constraints and introducing a penalty term. The final optimization problem inherits the advantages of the efficient SGD algorithm in large-scale matrix factorization [102]. Experimental results show that our proposed transfer learning solution significantly outperforms the state-of-the-art matrix factorization approach without using the auxiliary data.
CHAPTER 6

VIP RECOMMENDATION IN HETEROGENEOUS MICROBLOGGING SOCIAL NETWORKS

Recommending famous people or VIPs to ordinary users in a microblogging social network is a strategically important task, since good recommendations may improve users’ activities, like following and retweet. However, a microblog like Tencent Weibo has more than 200 million ordinary users, which makes Resnick’s rule, a classic memory-based collaborative filtering method, inapplicable, since calculating the similarity between pairs of users can become an extremely time-consuming task, if not impossible. Furthermore, the user-VIP following relations in Tencent Weibo are very sparse, which makes it difficult for us to accurately estimate the similarity between two users. Two important characteristics of the following data in Tencent Weibo are “big data” and “sparse data”, raising major computational challenges. In this chapter, we propose a novel large-scale transfer-learning based solution to address these two challenges in a single framework, which is called Social Relation based Transfer (SORT). In SORT, we shift from a focus on “similarity” as in Resnick’s rule to a relation-oriented concept. SORT focuses on inferring the target relations of user-VIP following in Tencent Weibo by transferring knowledge from auxiliary relations of user-user friendship in Tencent QQ (a chatting service), user-user following and VIP-VIP following in Tencent Weibo. SORT makes use of existing relations to address the scalability challenge by avoiding the similarity computation; it also transfers friendship relations from Tencent QQ to enrich the knowledge in Tencent Weibo and thus addresses the sparsity challenge. We demonstrate the effectiveness of the proposed transfer learning solution via experimental results on large real data from Tencent Weibo and Tencent QQ.
6.1 Introduction

Social network services of microblogging (e.g. Twitter\(^1\), Tencent Weibo\(^2\)) and instant messenger (e.g. Skype\(^3\), Tencent QQ\(^4\)) are playing an increasingly important role in users’ daily lives, including relationship maintenance and building, information sharing and seeking, and other online social activities. Similar to the fundamental motivation of information overload [203] in recommender systems [173], users may feel difficult to find other interesting users to follow from the hundreds of millions of users within the same social network platform. One example of this challenge is in microblogging services, where hundred-thousands or even millions of new users join the network everyday. Effective solutions for people recommendation must overcome the challenge of “user overload” in such a social network, similar to the challenge of “information overload” in online shopping sites like Amazon\(^5\).

In the Chinese microblogging social network of Tencent Weibo, some famous people (known as VIPs) contribute to the social network development significantly. For example, Dr. Kai-Fu Lee is one such super star in Tencent Weibo who has more than 19 million followers and has posted more than 1.8 thousand messages. VIP recommendation is thus a strategically important task, since good VIP recommendation brings in more relations and activities in the online social community. However, even for VIP recommendation, the problem of user overload, or more precisely VIP overload, still exists, since several thousands of famous people (VIPs) have registered in Tencent Weibo. Like the recommender systems for books [125], videos [46] and academic papers [198], the recommendation engine under Tencent Weibo has to suggest some interesting VIPs for each user to follow, and the challenge is accuracy despite the data scale and sparsity. We illustrate the scenario of VIP recommendation in Tencent Weibo in Figure 6.1, where a personalized list of VIPs is generated once a user logs into the social network.

There are two main challenges for the task of people recommendation. First, the following relation data in a microblogging social network are very sparse, making it

\(^{1}\)http://twitter.com/
\(^{2}\)http://t.qq.com/
\(^{3}\)http://www.skype.com/
\(^{4}\)http://im.qq.com/
\(^{5}\)http://www.amazon.com/
difficult to apply traditional similarity based techniques. Second, this data is extremely large, thus pairwise similarity calculation would be infeasible. To solve these problems, in this chapter, we propose a relation-oriented transfer-learning method, which extracts useful knowledge from the auxiliary data consisting of other services and social networks in Tencent, and applies their common knowledge to help improve people recommendations in Tencent Weibo. Transfer learning is a machine learning method that discovers common knowledge among seemingly different data for the purpose of improving the learning performance of a target data [157, 115, 160]. We call our new method Social Relation based Transfer (SORT). The SORT method has two major advantages over traditional memory-based methods like the Resnick’s rule [172]. First, it is very efficient for extremely large user set, since it avoids the time consuming step of similarity calculation. Second, it recommends accurately by leveraging additional knowledge from a mature social network of instant messenger via transfer-learning techniques.

Our main contributions include two aspects. First, we define and study a new problem of one-class collaborative filtering across two real heterogeneous social networks as shown in Figure 6.2. Second, we propose a novel efficient recommendation algorithm using transfer learning techniques.

The organization of the chapter is as follows. We first discuss some related work in Section 6.2. We then give a formal definition of the problem in Section 6.3 and describe our solution in detail in Section 6.4. We present experimental results on real-world data sets in Section 6.5. Finally, we give some concluding remarks and future
works in Section 6.6.

6.2 Related Work

Table 6.1: Summary of SORT and other transfer learning methods in collaborative filtering.

<table>
<thead>
<tr>
<th>Model-based</th>
<th>Knowledge (what to transfer)</th>
<th>Algorithm style (how to transfer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMF [177] family</td>
<td>Covariance</td>
<td>Adaptive_</td>
</tr>
<tr>
<td></td>
<td>Latent features</td>
<td>Collective</td>
</tr>
<tr>
<td>NMF [113] family</td>
<td>Codebook</td>
<td>Integrative</td>
</tr>
<tr>
<td></td>
<td>Latent features</td>
<td></td>
</tr>
<tr>
<td>Memory-based</td>
<td>Social relations</td>
<td></td>
</tr>
</tbody>
</table>

The proposed solution transfers knowledge from a bidirectional social network of instant messenger to a unidirectional social network of microblog in order to recommend famous people or VIPs. In this section, we discuss some related works in three areas, people recommendation, recommendation using social trust, and transfer learning methods in collaborative filtering.

6.2.1 People Recommendation

Guy et al. [73] propose a recommender engine called StrangerRS and conduct a user study ($n = 516$) of strangers recommendation within a company. The users’ existing familiarity network is first removed from the users’ similarity network, which is mined from co-occurrence information, like co-tagging, co-bookmarking, co-membership, etc.. And thus, the remaining unfamiliar employees (strangers) can be recommended to the user. This work uses the Jaccard index for similarity measures [73]. One drawback of this method is inefficiency: it is time consuming when the user space is very large (e.g. $n > 10^8$) to complete the similarity calculation. The same computational problem exists in software-item recommendation [74] and familiar-people recommendation [72]. A recent work on software item-installation prediction using auxiliary social networks [158] demonstrated that this method cannot scale up to large-scale problems.

Armentano et al. [7] study followee recommendation in the Twitter system using a topology-based algorithm, which recommends the followees of a user $u$’s co-followers.
to the user, where user $u$ and user $w$ are co-followers if they have followed at least one same followee [7]. The proposed topology-based method cannot be used in VIP recommendation since some VIPs have $10^7$ followers. Thus, the number of co-followers of a target user $u$ is also $10^7$, a challenge not observable in a small data set. For example, the data set used in the experiment of [7] is a subset of Twitter social network with $1.44 \times 10^6$ users and $3.46 \times 10^6$ following relations.

Hannon et al. [75] convert the followee recommendation problem to a query-based search problem via a pre-processing step of user profiling. This work represents the user’s profile with the user’s Tweets, the followers’ ID, the followees’ ID, the followers’ Tweets and the followees’ Tweets. For any target user $u$, the most similar $k$ users returned by Lucene\(^6\) are recommended to the target user. This approach is proved to be useful for ordinary user recommendation in symmetric social networks formed by user-user relations, but may be not suitable for VIP recommendation, which is antisymmetric in nature, since the profiles of VIP are very different from that of the ordinary users. For example, their followers, followees, and Tweets are all asymmetric, and thus calculating the similarity between an ordinary user and a VIP via their profiles may not work well.

Our work differs from the aforementioned works in two aspects. First, we study an extremely large social recommendation problem with $10^8$ users, instead of a small-sized data. Second, we propose to shift the focus from the core concept of “similarity” in people recommendation or “proximity” in link prediction [124] to a new concept of social “relations”, or social chains, for both the auxiliary social networks and the target social networks. We will show more details of our solution in the following sections.

### 6.2.2 Trust-based Recommendation

The trust-aware recommender systems [144] generalize the well known Resnick’s formula [172] as follows,

\[
\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{w \in T_u^+} t_{uw} (r_{wi} - \bar{r}_w)}{\sum_{w \in T_u^+} t_{uw}}
\]

(6.1)

where the set of selected nearest raters is replaced by the set of trusted users $T_u^+$, and the similarity between user $u$ and user $w$ is replaced by trust value $t_{uw}$. The authors

\(^6\)http://lucene.apache.org/
also mention to combine trust information and similarity information [144],

\[ \hat{r}_{ui} = \bar{r}_{u} + \frac{\sum_{w \in N_u} PCC(u, w)(r_{wi} - \bar{r}_{w}) + \sum_{w \in T_u^+} t_{uw}(r_{wi} - \bar{r}_{w})}{\sum_{w \in N_u} PCC(u, w) + \sum_{w \in T_u^+} t_{uw}}, \]

where similarity value \( PCC(u, w) \), trust value \( t_{uw} \), nearest raters \( N_u \) and trusted users \( T_u^+ \) are all used in the prediction rule in a hybrid way. This method is also known as MoleTrust [9], since the trust value \( t_{uw} \) is estimated by the MoleTrust algorithm [9] in a depth-first search manner.

The FilmTrust system [63] adopts a simplified prediction rule as compared to Eq.(6.1),

\[ \hat{r}_{ui} = \frac{\sum_{w \in T_u^+} t_{uw}r_{wi}}{\sum_{w \in T_u^+} t_{uw}} \]  

(6.2)

where \( t_{uw} \) is again the trust value between user \( u \) and user \( w \), and \( T_u^+ \) is the set of trusted users of user \( u \). The FilmTrust system mainly contains four steps. First, it searches raters on item \( i \) that the target user \( u \) knows, using \( k \)-step connections where \( k \) starts with 1 and stops until some raters are found. Second, it calculates the trust value between user \( u \) and the founded rater \( w \), using the TidalTrust algorithm [62] in a breadth-first manner. Third, it selects a set of trusted raters, \( T_u^+ \), with maximum trust values (above a certain threshold). Finally, it predicts the rating of user \( u \) on item \( i \) using the prediction rule in Eq.(6.2).

O’Donovan et. al [150] propose to replace the trust value \( t_{uw} \) and trusted users \( T_u^+ \) in Eq.(6.1) as follows,

\[ t_{uw} \leftarrow \frac{2PCC(u, w)t_{uw}}{PCC(u, w) + t_{uw}}, \quad T_u^+ \leftarrow N_u \cap T_u^+ \]  

(6.3)

where \( N_u \) is the set of selected nearest raters of user \( u \). O’Donovan et. al [150] introduce a trust-based weighting strategy, a trust-based filtering strategy, and a hybrid approach. Note that the trust value \( t_{uw} \) in [150] is not estimated from an auxiliary social network, but from the process of rating prediction. Specifically, each user \( w \) can be considered as a committee member of user \( u \), and if the rating of user \( w \) is different from that of user \( u \), then the trust will be reduced on-the-fly [150]. The trust value [150] can also be in a finer granularity, e.g. user \( u \) may be influenced by user \( w \) only on a certain topic [201].
Victor et al. [208] propose to combine the similarity value and trust value in a single prediction rule,

\[
\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{w \in T_u^+} t_{uw}(r_{wi} - \bar{r}_w) + \sum_{w \in N_u \setminus T_u^+} PCC(u, w)(r_{wi} - \bar{r}_w)}{\sum_{w \in T_u^+} t_{uw} + \sum_{w \in N_u \setminus T_u^+} PCC(u, w)},
\]

which combines Resnick’s formula [172] and trust-based method in Eq.(6.1), and can help improve the coverage over each single approach. Victor et al. [208] mainly discuss the effect of distrust, which is observed in a real social network of Epinion\(^7\).

Jamali et al. [84] turn to use a random walk algorithm over the trust network with item-item similarities into consideration. The main idea is that when a trusted user \(w\) of user \(u\) has not rated the item \(i\), but rated a similar item \(j\), then the rating \(r_{wj}\) can still be used to predict the rating of user \(u\) on item \(i\), \(\hat{r}_{ui}\).

We can see that the core concept in the trust-based recommendation is “trust value” between user \(u\) and user \(w\). Different algorithms are proposed to estimate or generalize the trust value, e.g. trust propagation algorithms of TidalTrust [62] used in [63] and Mole Trust [9] used in [144], blending of user-user trust value and user-user similarity as used in [150, 208], and combination of user-user trust value and item-item similarity as used in [84], etc.

In our transfer learning solution, we drop the concept of “similarity” in the Resnick’s formula [172] or the “trust value” in Eq.(6.1), and turn to use existing social relations for both the auxiliary social network and the target social network, since the similarity or trust value may not be estimated efficiently and accurately in a big and sparse data.

### 6.2.3 Transfer Learning Methods in Collaborative Filtering

In the past, researchers have applied transfer learning techniques to collaborative filtering problems. For example, collective matrix factorization (CMF) [193] jointly factorize two data of user-item rating matrix and item-content matrix, and share the learned item-side latent feature matrix to achieve knowledge transfer. Codebook transfer (CBT) [115] and rating-matrix generative model (RMGM) [116] share some latent compressed rating patterns from two different domains of books and movies, which are built based on the assumption that the high level rating behaviors of user groups

\(^7\)http://www.epinions.com/
on item categories are stable and relatively consistent across two heterogeneous domains. Weighted nonnegative matrix co-tri-factorization (WNMCTF) [221] combines non-negative matrix factorization (NMF) [113] and CMF [193] to jointly factorize three data of user-item rating matrix, user-content matrix and item-content matrix, and share the latent feature matrices in a similar way as that in CMF [193]. Coordinate system transfer (CST) [160] leverages both user-side and item-side auxiliary data of implicit feedbacks via biased regularization on latent feature matrices. Dependent matrix factorization (DPMF) [3] shares latent features’ covariance matrix. Finally, transfer by collective factorization (TCF) [159] models data-independent knowledge and data-dependent effect simultaneously for heterogeneous user feedbacks of 5-star numerical grades and like/dislike binary ratings. Pan et al. [159] give a detailed discussion on this from the perspective of “what to transfer” and “how to transfer” in transfer learning [157] and collaborative filtering. These works can be categorized as model-based transfer, in parallel to the the binary categorization of model-based methods and memory-based methods in collaborative filtering [26].

The proposed solution, SORT, can be considered as a memory-based transfer learning approach, which transfers knowledge of social relations from an auxiliary data. Different from existing adaptive styles [115, 160] and collective styles [193, 116, 221, 3, 159] of the transfer learning algorithms, SORT follows an integrative style, which will be discussed about in more detail in following sections. A brief summary of related transfer learning works and our SORT method is given in Table 6.1, which shows that SORT is different from existing transfer learning works in both dimensions, “what to transfer” (social relations) and “how to transfer” (integrative).

6.3 VIP Recommendation

6.3.1 Problem Definition

In a target domain consisting of a microblogging social network, we have \( n \) users and \( m \) VIPs, where the \( m \) (\( 10^3 \sim 10^4 \)) VIPs are selected by human experts considering the various factors including social impact and business influence. Our goal is to recommend the top-\( k \) VIPs among the given \( m \) VIPs for each of \( n \) (\( \sim 10^8 \)) users. Due to the sparsity of the user-VIP matrix of following relations, we are concerned about the efficiency and effectiveness of the solution. Thus, we wish to exploit an auxiliary data;
this is our new problem setting for VIP recommendation using auxiliary data.

Mathematically, we have a matrix \( R = [r_{ui}]_{n \times m} \in \{1, ?\}^{n \times m} \), where “1” denotes the observed following relation between user \( u \) and VIP \( i \), and the question mark “?” denotes a missing value (unobserved value). Note that the following relations are usually considered as weak ties. We use a mask matrix \( Y = [y_{ui}]_{n \times m} \in \{0, 1\}^{n \times m} \) to denote whether the entry \((u, i)\) is observed \((y_{ui} = 1)\) or not \((y_{ui} = 0)\). Similarly, in the auxiliary domain of instant messenger, we have a matrix \( X = [x_{uw}]_{n \times n} \in \{1, ?\}^{n \times n} \), where “1” denotes the observed friendship relation between user \( u \) and user \( w \), and “?” denotes the missing value. Since the instant messenger of Tencent QQ has been developed for more than twelve years\(^8\) and the friendship relations represent strong ties, we simplify the friendship relation matrix as \( X = [x_{uw}]_{n \times n} \in \{1, 0\}^{n \times n} \), where “0” denotes the non-friend relation between user \( u \) and user \( w \). Note that there is an one-one mapping between the users of \( R \) and \( X \). Our goal is to help each user \( u \) find a personalized list of top-\( k \) VIPs \((y_{ui} = 0)\) by transferring knowledge from \( X \).

Note that the auxiliary social network of instant messenger, \( X \), can also be replaced by the following relations between users \( S_1 \in \{1, ?\}^{n \times n} \) or VIPs \( S_2 \in \{1, ?\}^{m \times m} \) within the same target social network of microblog. Considering the “distance” or “analogy” of \( X \) and \( R \), and that of \( S_1 \) (or \( S_2 \)) and \( R \), these two settings can be considered as far transfer and near transfer, respectively [78].

In a brief summary, the proposed problem setting can be considered as transferring knowledge over two real heterogeneous social networks of instant messenger and microblog,

\[
\begin{align*}
X \Rightarrow R, & \quad \text{far transfer} \\
S_1, S_2 \Rightarrow R, & \quad \text{near transfer}
\end{align*}
\]

where far transfer represents knowledge transfer across two heterogeneous social networks of instant messenger and microblog, and near transfer for that within the target social network of microblog. Our goal is to predict the missing values in \( R \), and thus we can rank and recommend VIPs for each user. \( X, S_1, S_2 \) represent the auxiliary user-user friendship, user-user following and VIP-VIP following relations, respectively. As far as we know, there is no previous work studying the same problem as ours, which is shown in Figure 6.2.

---

\(^8\)Tencent’s instant messenger was first launched in February 1999. Please refer to http://en.wikipedia.org/wiki/Tencent QQ for more information.
6.3.2 Overview of Our Solution

We propose a memory-based transfer learning solution called Social Relation based Transfer (SORT), to address the two challenges from “big data” and “sparse data”. Since it is extremely time consuming to estimate the similarities among more than 200 million users and the estimated similarities may be not accurate due to the sparsity problem, we wish to shift our attention from the core concept “pairwise similarity” in memory based collaborative filtering methods to relationship transfer. We introduce the concept “relation” for both the target and auxiliary social networks; that is, we use existing relations to replace the procedure of similarity calculation. Our idea of using existing social relations to replace similarity calculation is novel:

1. we derive an efficient relation-oriented prediction method to address the first challenge of scalability;

2. we propose a transfer learning approach to leverage social relations from the auxiliary data to address the second challenge of sparsity in the target domain.

The knowledge from the auxiliary social network of instant messenger can thus be efficiently and effectively transferred to the target microblogging social network for VIP recommendation.
6.4 Social Relation based Transfer

6.4.1 Prediction Method

There are two main branches of collaborative filtering methods, memory-based methods and model-based methods [26]. In this chapter, we focus on the memory-based methods for VIP recommendation, which have excellent ability in interpreting the recommendation results and need little parameter tuning work. These considerations are particularly important for a real-world recommender system.

We introduce our ideas in three steps. First, we start with the well-known memory-based method of numerical collaborative filtering for 5-star grade prediction, show its inapplicability for our one-class collaborative filtering problem of VIP recommendation, and then derive a simplified prediction rule. Second, based on a novel idea of using existing social relations to avoid similarity calculation, we further derive a relation-oriented prediction method extended from the simplified prediction rule. Finally, we propose our social relation based transfer learning solution for VIP recommendation in microblog.

A Simplified Prediction Rule

Pearson correlation coefficient (PCC) [172] is a widely adopted similarity measure of two users $u$ and $w$ based on the ratings on their commonly rated items,

$$PCC(u, w) = \frac{\sum_i y_{ui} y_{wi} (r_{ui} - m_u)(r_{wi} - m_w)}{\sqrt{\sum_i y_{ui} y_{wi} (r_{ui} - m_u)^2} \sqrt{\sum_i y_{ui} y_{wi} (r_{wi} - m_w)^2}},$$

where $m_u = \frac{\sum_i y_{ui} r_{ui}}{\sum_i y_{ui}}$ is the average rating of user $u$, and $m_w = \frac{\sum_i y_{wi} r_{wi}}{\sum_i y_{wi}}$ is the average rating of user $w$. The normalized similarity between users $u$ and $w$ can then be calculated as follows,

$$s_{uw} = \frac{PCC(u, w)}{\sum_{u' \in N_u} PCC(u, u')}^{-1},$$

where $N_u$ is the set of some nearest neighboring users of user $u$ according to $PCC$ measurement. Finally, we can predict the rating of user $u$ on item $i$ [172],

$$\hat{r}_{ui} = \bar{r}_u + \sum_{w \in N_u} y_{ui} s_{uw} (r_{wi} - m_w), \quad (6.5)$$
where \( \bar{r}_u = \frac{\sum_i y_{ui} r_{ui}}{\sum_i y_{ui}} \) is the average rating of user \( u \) [172] on all items rated by user \( u \). We can equivalently re-write Eq.(6.5) as follows,

\[
\hat{r}_{ui} = \bar{r}_u - \sum_{w \in N_u} y_{ui} s_{uw} m_{uw} + \sum_{w \in N_u} y_{wi} s_{uw} r_{wi},
\]  

(6.6)

where the first term represents user \( u \)'s global average rating, and the second term represents the aggregation of \( |N_u| \) nearest neighbors’ local average ratings. For the one-class collaborative filtering problem of VIP recommendation, \( \bar{r}_u = 1, m_{uw} = 1 \), and thus such average ratings do not contain any discriminative information, and we may safely discard them. Finally, we obtain a simplified prediction rule,

\[
\hat{r}_{ui} = \sum_{w \in \tilde{N}_u} y_{wi} s_{uw} r_{wi},
\]  

(6.7)

which means that the rating of user \( u \) on item \( i \) can be estimated from user \( u \)'s \( |N_u| \) nearest neighbors’ preferences on item \( i \) via a weighted aggregation.

A Relation-Oriented Prediction Method

There are two main difficulties associated with the simplified prediction rule in Eq.(6.7). First, for \( 10^8 \) users in microblog, it’s extremely time consuming to calculate the similarity \( s_{uw} \) between every two users \( u \) and \( w \) and then find some nearest neighbors \( N_u \) for each user \( u \) according to the similarities, even using distributed computing techniques. Second, as a newly built-up microblogging social network (Tencent Weibo), the social network is very sparse and thus the similarity \( s_{uw} \) may be not accurate.

Can we address the scalability challenge and sparsity challenge using some auxiliary data? In this chapter, we propose a novel idea of replacing the the similarity calculation in the target data with existing relations from an auxiliary data. Specifically, we use an auxiliary well-developed social network of instant messenger, which can avoid the procedure of similarity calculation and neighborhood search. We replace \( N_u \) in Eq.(6.7) with \( \tilde{N}_u \), and \( s_{uw} \) with \( x_{uw} \), and then obtain a revised prediction rule,

\[
\hat{r}_{ui} = \sum_{w \in \tilde{N}_u} y_{wi} x_{uw} r_{wi},
\]  

(6.8)

where \( \tilde{N}_u \) represents the set of user \( u \)'s friends in the social network of instant messenger, and \( x_{uw} \) represents the relationship of user \( u \) and his/her friend \( w \). To consider
each friend equally, we set $x_{uw} = 1$ in Eq.(6.8), and obtain,

$$
\hat{r}_{ui} = \sum_{w \in \tilde{N}_u} y_{wi} r_{wi}.
$$

(6.9)

For the one-class collaborative filtering problem in the social network of microblog, we can further replace the term $y_{wi} r_{wi}$ in Eq.(6.9) with $f_{wi} = \begin{cases} 1, & \text{user } u \text{ has followed VIP } i \\ 0, & \text{otherwise} \end{cases}$, and have,

$$
\hat{r}_{ui} = \sum_{w \in \tilde{N}_u} f_{wi}.
$$

(6.10)

where the prediction method can be interpreted as follows, “if user $u$ has $|\tilde{N}_u|$ friends in the social network of instant messenger, and $\sum_{w \in \tilde{N}_u} f_{wi}$ of them have followed VIP $i$ in the social network of microblog, then the score or preference of user $u$ on VIP $i$ is $\sum_{w \in \tilde{N}_u} f_{wi}$”. We can see that two real heterogeneous social networks of microblog (the following relation $f_{wi}$) and instant messenger (the friendship relations $\tilde{N}_u$) are integrated together in such an intuitive way as shown in Eq.(6.10). The knowledge of social relations, $\tilde{N}_u$, of instant messenger is embedded naturally in the prediction method.

In the above, we can see that the predicted score $\hat{r}_{ui}$ in Eq.(6.10) must be an integer since $f_{wi}$ is either 1 or 0, and one user $u$ may have same score on several different VIPs, where we can not distinguish the ranking positions. To address this problem, we further introduce a popularity score for each VIP $i$, $0 \leq p_i \leq 1$, $i = 1, \ldots, m$, which can be considered as an approach of secondary sort of VIPs with same score as estimated from Eq.(6.10). We thus reach the prediction method,

$$
\hat{r}_{ui} = p_i + \sum_{w \in \tilde{N}_u} f_{wi}.
$$

(6.11)

The proposed approach transfers friendship social relations, $\tilde{N}_u$, from an auxiliary social network of instant messenger to a target VIP prediction problem in microblog. We can see that the procedures of similarity calculation and neighbor search in Resnick’s rule [172] is avoided.

The idea of relation-oriented prediction method can be represented as a social chain [94] with heterogeneous relations of friendship and following,

$$
\text{user} \sim \text{friend} \rightarrow \text{VIP}
$$

(6.12)
where the first sub-chain “user \sim friend” represents the friendship relation of user and friend in the social network of instant messenger, and the second sub-chain “friend \rightarrow VIP” means the following relation of friend and VIP in the social network of microblog. We illustrate recommendation procedure in Figure 6.3.

![Figure 6.3: Illustration of the recommendation procedure using instant messenger X and microblog R. From the friendship relations in instant messenger, we can find five friends of user A: B, C, D, E, F, and according to the following relations in microblog (B \rightarrow X, C \rightarrow X, D \rightarrow X and E \rightarrow Y, F \rightarrow Y), we can recommend VIP X and Y to user A.](image)

**Social Relation Based Transfer**

The concept of “friend” in the social network of an instant messenger (such as MSN, or Tencent QQ) can be generalized to include “followee” and “follower” in the social network of microblog. As shown in the problem setting in Eq.(6.4) and the proposed recommendation procedure in Eq.(6.12), we can have two social chains for VIP recommendation,

\[
\begin{align*}
\{ & \text{user} \sim \text{friend} \rightarrow \text{VIP}, \quad \text{far transfer} \\
& \text{user} \sim \text{followee} \rightarrow \text{VIP}, \quad \text{near transfer} 
\end{align*}
\]  

(6.13)

where “user \sim friend” is from the user-user friendship relation matrix X, “friend \rightarrow VIP” is from the user-VIP following relation matrix R, “user \sim followee” is from the user-user and user-VIP following relation matrices S\(\_\_\_1\) and R, and “followee \rightarrow VIP” is from the VIP-VIP following relation matrix S\_\_\_2. Inspired by the prediction method in Eq.(6.11), we propose to combine these two social chains into one via neighborhood expansion and obtain an integrated solution,

\[
\hat{r}_{ui} = p_i + \sum_{w \in N_u} f_{wi} + \sum_{w \in \bar{N}_u} f_{wi}. 
\]  

(6.14)
where $\tilde{N}_u$ and $\bar{N}_u$ represent sets of user $u$’s friends and followees, respectively.

We can see that the knowledge of social relations used in social chains are different: one is from instant messenger relations and the other is from microblogging networks, where the former are strong bidirectional ties while the later are weak unidirectional ties. The knowledge of social relations from instant messenger ("what to transfer") are thus transferred via neighborhood expansion ("how to transfer") to the target problem of VIP recommendation in microblog, which answers two fundamental questions of “what to transfer” and “how to transfer” in transfer learning [157]. More specifically, SORT is built based on social chains from heterogeneous social networks: the social network of instant messenger (Tencent QQ) is first embedded in the prediction method, and then the results from two social chains are further integrated via neighborhood expansion or preference blending.

### 6.4.2 Analysis

The time complexity of SORT is linear in the number of social relations. SORT can be implemented in a distributed computing platform of Hadoop⁹. Take the social chain of “user $\sim$ friend $\rightarrow$ VIP” as an example, each user $\sim$ friend or friend $\rightarrow$ VIP is first stored in one line in the Hadoop file system. In Hadoop, we use friend as a key for Map’s output (or equivalently Reduce’s input) in a first Map/Reduce job, and then accumulate the co-occurrence of user $\rightarrow$ VIP in a second Map/Reduce job, in which we obtain the accumulated preference score of each user on each reached VIP. Thus, the time complexity is $O(q + \tilde{q})$, where $q$ is the number of following relations in $R$ and $\tilde{q}$ is the number of friendship relations in $X$. The time complexity of the other social chain of “user $\sim$ followee $\rightarrow$ VIP” is similar.

### 6.5 Experimental Results

We conduct experiments to verify two hypotheses: first, we believe that the proposed transfer learning method (either far transfer or near transfer) can improve VIP recommendation in microblog, since it makes use of auxiliary data to address the sparsity problem; second, we believe that far transfer and near transfer are complementary, and shall behave differently on users with different sparsity levels.

⁹http://hadoop.apache.org/
6.5.1 Data Sets and Evaluation Metrics

**Data Set of Tencent Weibo**  The microblogging social network data of Tencent Weibo contains more than 200 million users by August 2011. The distribution of the *following* relations is extremely unbalance [58], since some super-star users or VIPs may have more than one million followers, while most ordinary users only have dozens of followers. In the experiment, we use the whole data set, and focus on recommending VIPs from a selected VIP pool ($10^3 < m < 10^4$) to each user. Note that the selection of the VIP pool is conducted by human experts considering various social and business factors.

**Data Set of Tencent QQ** The instant messenger social network data of Tencent QQ contains about 1 billion registered users and about 80 billion *friendship* relations. In the experiments, we transfer the *friendship* relations of all users that registered both in Tencent QQ and Tencent Weibo.

The instant messenger social network data of Tencent QQ is relatively stable, and we use the data by August 15, 2011 as the auxiliary data. We use the microblogging data of Tencent Weibo by August 21, 2011 as training data, and use newly added *following* relations of August 22-24, 2011 as test data. Note that millions of users add new *following* relations in Tencent Weibo everyday. The data sets are new and among the largest ones in published papers on recommender systems as far as we know.

**Evaluation Metrics** We adopt the widely used evaluation metrics in information retrieval and recommender systems, precision, recall, F1 and NDCG. According to the predicted preferences (or ratings) of user $u$ on VIPs, we can get a ranked list of top-$k$ VIPs,

$$i(1), \ldots, i(\ell), \ldots, i(k),$$

where $i(\ell)$ represents the VIP located at position $\ell$.

1. The precision is defined as follows,

$$\text{Pre}_u@k = \frac{1}{k} \sum_{\ell=1}^{k} y_{u,i(\ell)},$$

where $\sum_{\ell=1}^{k} y_{u,i(\ell)}$ represents the number of matched VIPs from top-$k$ VIPs recommended to user $u$ and the user $u$’s truly followed VIPs.
2. The recall is defined as follows,

$$Rec_u@k = \frac{1}{\sum_{i=1}^{m} y_{ui}} \sum_{\ell=1}^{k} y_{u,i(\ell)};$$

where $\sum_{i=1}^{m} y_{ui}$ denotes the number of user $u$’s truly followed VIPs.

3. The F1 score is defined based on the precision and recall,

$$F1_u@k = 2 \times \frac{Pre_u@k \times Rec_u@k}{Pre_u@k + Rec_u@k}.$$

4. The NDCG score is defined as follows,

$$NDCG_u@k = \frac{1}{Z_u} \sum_{\ell=1}^{k} \frac{2^{r_{u,i(\ell)}} - 1}{\log(\ell + 1)},$$

where $r_{u,i(\ell)}$ is user $u$’s true rating on the VIP at position $\ell$, and $Z_u$ is a normalization term with value of the best $DCG@k$ score. In our one-class collaborative filtering problem, we use $y_{u,i(\ell)}$ for $r_{u,i(\ell)}$.

In our experiments, we report the average score of the above four evaluation metrics on all test users,

$$Pre@k = \frac{1}{|T_E|} \sum_{u=1}^{|T_E|} Pre_u@k,$$

$$Rec@k = \frac{1}{|T_E|} \sum_{u=1}^{|T_E|} Rec_u@k,$$

$$F1@k = \frac{1}{|T_E|} \sum_{u=1}^{|T_E|} F1_u@k,$$

$$NDCG@k = \frac{1}{|T_E|} \sum_{u=1}^{|T_E|} NDCG_u@k.$$

where $|T_E|$ is the number users who added new following relations between August 22 and August 24, 2011.

### 6.5.2 Baselines and Parameter Settings

We study the transfer learning solution SORT with three variants,
1. social chain “user $\sim$ friend $\rightarrow$ VIP”, which is denoted as SORT-friend or Friend for short;

2. social chain “user $\sim$ followee $\rightarrow$ VIP”, which is denoted as SORT-followee or Followee; and

3. both chains, which is denoted as SORT-friend-followee or SORT.

We also study the performance of a VIP-side memory based method, since the number of VIPs is relatively small compared to that of ordinary users ($m \ll n$). The similarity of VIP $i$ and VIP $j$ is calculated using Jaccard index,

$$ s_{ij} = \sum_{u=1}^{n} y_{ui}y_{uj} / \sum_{u=1}^{n} \max(y_{ui}, y_{uj}) $$

where the numerator and denominator represent the intersection and union of followers of VIP $i$ and VIP $j$, respectively. With the VIP-VIP similarity, we have the following prediction rule,

$$ \hat{r}_{ui} = \sum_{j=1}^{m} s_{ij}y_{uj}f_{uj} $$

where $\hat{r}_{ui}$ represents the user $u$’s preference score on VIP $i$. We denote this memory-based method as “Memory”.

**Preliminary Studies** We study the performance of another social relation based method in the early stage of the system development,

$$ user \sim followee \rightarrow VIP $$

which is denoted as SORT-follower. The result of SORT-follower is much worse than the aforementioned methods. It is very interesting to see that the follower-based approach [75] performs well on ordinary user recommendation (not VIP recommendation) in Twitter. We believe that the reason is that the user bases of Twitter and Chinese microblog constitute two different cultural groups, with different tasks of ordinary user recommendation and VIP recommendation.

We also implement a distributed version of probabilistic matrix factorization (PMF) [177, 152], which was used in Netflix\textsuperscript{10} and Yahoo! music recommendation\textsuperscript{11} competitions, while the result is extremely poor in our experiments. We think that there are

\textsuperscript{10}http://www.netflix.com/

\textsuperscript{11}http://kddcup.yahoo.com/
Table 6.2: Prediction performance of matrix factorization and “Memory” in our preliminary study (training: accumulated data till May 31, 2011, test: new following relations between June 1 and June 7, 2011).

<table>
<thead>
<tr>
<th></th>
<th>Precision@30</th>
<th>Recall@30</th>
<th>F1@30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix factorization</td>
<td>0.0036</td>
<td>0.0200</td>
<td>0.0050</td>
</tr>
<tr>
<td>Memory</td>
<td>0.0230</td>
<td>0.2438</td>
<td>0.0366</td>
</tr>
</tbody>
</table>

at least two reasons: first, the extreme unbalance of the distribution of the user-VIP following relations; second, sampling negative user-VIP following relations in microblog may not well represent the real relationships, which is different from that of ratings assigned by users on movies [152]. Specifically, in our preliminary study of matrix factorization, we randomly sample the same number of negative following relations as that of observed positive following relations, and implement a stochastic gradient descent (SGD) based matrix factorization algorithm on the Hadoop Map/Reduce platform. We have tried both basic matrix factorization and matrix factorization with item bias, different numbers of latent dimensions of \{5, 10, 15, 50\}, different iteration numbers of \{10, 20, 30\}, and different tradeoff parameters of \{0.001, 0.005, 0.01\}. The learning rate is fixed as 0.005. The best result in our preliminary studies on an early data set is shown in Table 6.2. Note that the method “Memory” as shown in Table 6.2 calculates the VIP-VIP similarity using the VIPs’ profiles instead of Jaccard index, but that does not make big differences on the result in our observations.

According to our preliminary studies, we discard SORT-follower and PMF during our system development stage, since they are much worse than the memory-based method “Memory”.

### 6.5.3 Summary of Experimental Results

To gain some deep understanding of the performance of different methods on users with different sparsity levels [116, 159], we denote those users who have followed some VIPs as warm-start users and those who have not followed any VIP as cold-start users. We study the recommendation performance on these different users separately. The results of NDCG@k on test data (unavailable during training) are shown in Figures 6.4. From the figure, we can have the following observations,

1. The overall performance on three user sets is (from best to worst): warm-start
user set, whole user set and cold-start user set, which is consistent with existed observations of the effect of sparsity in various transfer learning works, e.g. [116, 159].

2. The non-transfer learning method “Memory” performs worst on the whole user set and warm-start user set, since it does not exploit the collective intelligence among users; and is not applicable for cold-start users, since there is no observed preference data (or following relations) for cold-start users.

3. SORT-friend-followee performs best overall, which shows the effect of knowledge transfer from instant messenger to microblog.

4. For the two methods using single social chain, SORT-friend performs better than SORT-follower for top-k results when \( k \leq 5 \) on the whole user set and the warm-start user set, and worse when \( k > 5 \), which shows the complementary recommendation ability of the two methods, and also confirms the effect of knowledge transfer via social relations.

5. For the cold-start user set, SORT-friend performs much better than SORT-follower,.
### Table 6.3: Prediction performance on the whole user set.

<table>
<thead>
<tr>
<th>$k$</th>
<th>Memory</th>
<th>Friend</th>
<th>Followee</th>
<th>SORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0212</td>
<td>0.0377</td>
<td>0.0271</td>
<td>0.0374</td>
</tr>
<tr>
<td>2</td>
<td>0.0197</td>
<td>0.0347</td>
<td>0.0294</td>
<td>0.0361</td>
</tr>
<tr>
<td>3</td>
<td>0.0188</td>
<td>0.0330</td>
<td>0.0301</td>
<td>0.0355</td>
</tr>
<tr>
<td>4</td>
<td>0.0182</td>
<td>0.0317</td>
<td>0.0306</td>
<td>0.0351</td>
</tr>
<tr>
<td>5</td>
<td>0.0178</td>
<td>0.0307</td>
<td>0.0306</td>
<td>0.0345</td>
</tr>
<tr>
<td>10</td>
<td>0.0168</td>
<td>0.0269</td>
<td>0.0293</td>
<td>0.0318</td>
</tr>
<tr>
<td>15</td>
<td>0.0157</td>
<td>0.0242</td>
<td>0.0272</td>
<td>0.0290</td>
</tr>
<tr>
<td>20</td>
<td>0.0152</td>
<td>0.0220</td>
<td>0.0250</td>
<td>0.0265</td>
</tr>
<tr>
<td>25</td>
<td>0.0148</td>
<td>0.0203</td>
<td>0.0237</td>
<td>0.0244</td>
</tr>
<tr>
<td>30</td>
<td>0.0143</td>
<td>0.0188</td>
<td>0.0215</td>
<td>0.0226</td>
</tr>
</tbody>
</table>

### Precision

<table>
<thead>
<tr>
<th>$k$</th>
<th>Memory</th>
<th>Friend</th>
<th>Followee</th>
<th>SORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0118</td>
<td>0.0227</td>
<td>0.0162</td>
<td>0.0229</td>
</tr>
<tr>
<td>2</td>
<td>0.0215</td>
<td>0.0410</td>
<td>0.0339</td>
<td>0.0431</td>
</tr>
<tr>
<td>3</td>
<td>0.0306</td>
<td>0.0579</td>
<td>0.0517</td>
<td>0.0625</td>
</tr>
<tr>
<td>4</td>
<td>0.0395</td>
<td>0.0757</td>
<td>0.0897</td>
<td>0.0816</td>
</tr>
<tr>
<td>5</td>
<td>0.0483</td>
<td>0.0888</td>
<td>0.0867</td>
<td>0.0997</td>
</tr>
<tr>
<td>10</td>
<td>0.0914</td>
<td>0.1525</td>
<td>0.1646</td>
<td>0.1796</td>
</tr>
<tr>
<td>15</td>
<td>0.1278</td>
<td>0.2039</td>
<td>0.2263</td>
<td>0.2432</td>
</tr>
<tr>
<td>20</td>
<td>0.1646</td>
<td>0.2462</td>
<td>0.2751</td>
<td>0.2937</td>
</tr>
<tr>
<td>25</td>
<td>0.2000</td>
<td>0.2819</td>
<td>0.3158</td>
<td>0.3354</td>
</tr>
<tr>
<td>30</td>
<td>0.2307</td>
<td>0.3124</td>
<td>0.3499</td>
<td>0.3703</td>
</tr>
</tbody>
</table>

### Recall

<table>
<thead>
<tr>
<th>$k$</th>
<th>Memory</th>
<th>Friend</th>
<th>Followee</th>
<th>SORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0134</td>
<td>0.0254</td>
<td>0.0182</td>
<td>0.0256</td>
</tr>
<tr>
<td>2</td>
<td>0.0177</td>
<td>0.0329</td>
<td>0.0275</td>
<td>0.0345</td>
</tr>
<tr>
<td>3</td>
<td>0.0200</td>
<td>0.0366</td>
<td>0.0330</td>
<td>0.0395</td>
</tr>
<tr>
<td>4</td>
<td>0.0214</td>
<td>0.0386</td>
<td>0.0369</td>
<td>0.0428</td>
</tr>
<tr>
<td>5</td>
<td>0.0224</td>
<td>0.0398</td>
<td>0.0393</td>
<td>0.0448</td>
</tr>
<tr>
<td>10</td>
<td>0.0251</td>
<td>0.0407</td>
<td>0.0442</td>
<td>0.0480</td>
</tr>
<tr>
<td>15</td>
<td>0.0252</td>
<td>0.0391</td>
<td>0.0438</td>
<td>0.0468</td>
</tr>
<tr>
<td>20</td>
<td>0.0253</td>
<td>0.0370</td>
<td>0.0418</td>
<td>0.0444</td>
</tr>
<tr>
<td>25</td>
<td>0.0254</td>
<td>0.0350</td>
<td>0.0397</td>
<td>0.0419</td>
</tr>
<tr>
<td>30</td>
<td>0.0250</td>
<td>0.0331</td>
<td>0.0376</td>
<td>0.0395</td>
</tr>
</tbody>
</table>

since for cold-start users, the user-user and VIP-VIP following relations in the microblog are relatively few, and thus using the friendship social relations in instant messenger can help more.

We also report detailed results of precision, recall and F1 of different methods with different $k$ in Tables 6.3, 6.4, 6.5, where the observation is similar to that of Figure 6.4. This, from the empirical aspect, further confirms the effect of the social relation based transfer learning framework SORT.

### 6.6 Summary

In this chapter, we have exploited a novel transfer-learning method for solving the VIP recommendation problem in a microblogging social network. The proposed transfer
Table 6.4: Prediction performance on the *warm-start user set*.

<table>
<thead>
<tr>
<th>$k$</th>
<th>Memory</th>
<th>Friend</th>
<th>Followee</th>
<th>SORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0272</td>
<td><strong>0.0402</strong></td>
<td>0.0297</td>
<td>0.0399</td>
</tr>
<tr>
<td>2</td>
<td>0.0253</td>
<td>0.0371</td>
<td>0.0324</td>
<td><strong>0.0390</strong></td>
</tr>
<tr>
<td>3</td>
<td>0.0241</td>
<td>0.0333</td>
<td>0.0335</td>
<td><strong>0.0384</strong></td>
</tr>
<tr>
<td>4</td>
<td>0.0233</td>
<td>0.0339</td>
<td>0.0342</td>
<td><strong>0.0381</strong></td>
</tr>
<tr>
<td>5</td>
<td>0.0228</td>
<td>0.0330</td>
<td>0.0343</td>
<td><strong>0.0376</strong></td>
</tr>
<tr>
<td>10</td>
<td>0.0216</td>
<td>0.0289</td>
<td>0.0329</td>
<td><strong>0.0348</strong></td>
</tr>
<tr>
<td>15</td>
<td>0.0201</td>
<td>0.0260</td>
<td>0.0304</td>
<td><strong>0.0319</strong></td>
</tr>
<tr>
<td>20</td>
<td>0.0195</td>
<td>0.0238</td>
<td>0.0280</td>
<td><strong>0.0291</strong></td>
</tr>
<tr>
<td>25</td>
<td>0.0190</td>
<td>0.0219</td>
<td>0.0259</td>
<td><strong>0.0268</strong></td>
</tr>
<tr>
<td>30</td>
<td>0.0183</td>
<td>0.0204</td>
<td>0.0241</td>
<td><strong>0.0249</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.0151</td>
<td>0.0238</td>
<td>0.0176</td>
<td>0.0241</td>
</tr>
<tr>
<td>2</td>
<td>0.0276</td>
<td>0.0430</td>
<td>0.0372</td>
<td><strong>0.0457</strong></td>
</tr>
<tr>
<td>3</td>
<td>0.0393</td>
<td>0.0608</td>
<td>0.0570</td>
<td><strong>0.0666</strong></td>
</tr>
<tr>
<td>4</td>
<td>0.0507</td>
<td>0.0777</td>
<td>0.0772</td>
<td><strong>0.0873</strong></td>
</tr>
<tr>
<td>5</td>
<td>0.0619</td>
<td>0.0937</td>
<td>0.0964</td>
<td><strong>0.1071</strong></td>
</tr>
<tr>
<td>10</td>
<td>0.1172</td>
<td>0.1610</td>
<td>0.1828</td>
<td><strong>0.1942</strong></td>
</tr>
<tr>
<td>15</td>
<td>0.1639</td>
<td>0.2160</td>
<td>0.2511</td>
<td><strong>0.2638</strong></td>
</tr>
<tr>
<td>20</td>
<td>0.2112</td>
<td>0.2613</td>
<td>0.3051</td>
<td><strong>0.3192</strong></td>
</tr>
<tr>
<td>25</td>
<td>0.2566</td>
<td>0.2997</td>
<td>0.3501</td>
<td><strong>0.3651</strong></td>
</tr>
<tr>
<td>30</td>
<td>0.2959</td>
<td>0.3327</td>
<td>0.3883</td>
<td><strong>0.4037</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.0172</td>
<td>0.0268</td>
<td>0.0198</td>
<td>0.0271</td>
</tr>
<tr>
<td>2</td>
<td>0.0228</td>
<td>0.0348</td>
<td>0.0302</td>
<td><strong>0.0369</strong></td>
</tr>
<tr>
<td>3</td>
<td>0.0256</td>
<td>0.0387</td>
<td>0.0365</td>
<td><strong>0.0424</strong></td>
</tr>
<tr>
<td>4</td>
<td>0.0274</td>
<td>0.0410</td>
<td>0.0410</td>
<td><strong>0.0461</strong></td>
</tr>
<tr>
<td>5</td>
<td>0.0287</td>
<td>0.0424</td>
<td>0.0439</td>
<td><strong>0.0485</strong></td>
</tr>
<tr>
<td>10</td>
<td>0.0322</td>
<td>0.0434</td>
<td>0.0494</td>
<td><strong>0.0524</strong></td>
</tr>
<tr>
<td>15</td>
<td>0.0323</td>
<td>0.0419</td>
<td>0.0489</td>
<td><strong>0.0513</strong></td>
</tr>
<tr>
<td>20</td>
<td>0.0325</td>
<td>0.0398</td>
<td>0.0467</td>
<td><strong>0.0487</strong></td>
</tr>
<tr>
<td>25</td>
<td>0.0325</td>
<td>0.0377</td>
<td>0.0443</td>
<td><strong>0.0461</strong></td>
</tr>
<tr>
<td>30</td>
<td>0.0320</td>
<td>0.0357</td>
<td>0.0420</td>
<td><strong>0.0435</strong></td>
</tr>
</tbody>
</table>

127
Table 6.5: Prediction performance on the *cold-start* user set.

<table>
<thead>
<tr>
<th></th>
<th>Memory</th>
<th>Friend</th>
<th>Followee</th>
<th>SORT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Precision</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>0.0285</td>
<td>0.0179</td>
<td>0.0282</td>
</tr>
<tr>
<td>2</td>
<td>N/A</td>
<td>0.0264</td>
<td>0.0185</td>
<td>0.0261</td>
</tr>
<tr>
<td>3</td>
<td>N/A</td>
<td>0.0248</td>
<td>0.0182</td>
<td>0.0253</td>
</tr>
<tr>
<td>4</td>
<td>N/A</td>
<td>0.0237</td>
<td>0.0180</td>
<td>0.0244</td>
</tr>
<tr>
<td>5</td>
<td>N/A</td>
<td>0.0228</td>
<td>0.0176</td>
<td>0.0235</td>
</tr>
<tr>
<td>10</td>
<td>N/A</td>
<td>0.0198</td>
<td>0.0168</td>
<td>0.0210</td>
</tr>
<tr>
<td>15</td>
<td>N/A</td>
<td>0.0176</td>
<td>0.0156</td>
<td>0.0189</td>
</tr>
<tr>
<td>20</td>
<td>N/A</td>
<td>0.0159</td>
<td>0.0144</td>
<td>0.0171</td>
</tr>
<tr>
<td>25</td>
<td>N/A</td>
<td>0.0146</td>
<td>0.0134</td>
<td>0.0156</td>
</tr>
<tr>
<td>30</td>
<td>N/A</td>
<td>0.0134</td>
<td>0.0124</td>
<td>0.0144</td>
</tr>
<tr>
<td><strong>Recall</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>0.0187</td>
<td>0.0111</td>
<td>0.0186</td>
</tr>
<tr>
<td>2</td>
<td>N/A</td>
<td>0.0340</td>
<td>0.0225</td>
<td>0.0337</td>
</tr>
<tr>
<td>3</td>
<td>N/A</td>
<td>0.0473</td>
<td>0.0329</td>
<td>0.0482</td>
</tr>
<tr>
<td>4</td>
<td>N/A</td>
<td>0.0598</td>
<td>0.0431</td>
<td>0.0615</td>
</tr>
<tr>
<td>5</td>
<td>N/A</td>
<td>0.0715</td>
<td>0.0526</td>
<td>0.0737</td>
</tr>
<tr>
<td>10</td>
<td>N/A</td>
<td>0.1224</td>
<td>0.1002</td>
<td>0.1278</td>
</tr>
<tr>
<td>15</td>
<td>N/A</td>
<td>0.1610</td>
<td>0.1385</td>
<td>0.1704</td>
</tr>
<tr>
<td>20</td>
<td>N/A</td>
<td>0.1927</td>
<td>0.1690</td>
<td>0.2037</td>
</tr>
<tr>
<td>25</td>
<td>N/A</td>
<td>0.2190</td>
<td>0.1944</td>
<td>0.2301</td>
</tr>
<tr>
<td>30</td>
<td>N/A</td>
<td>0.2406</td>
<td>0.2143</td>
<td>0.2522</td>
</tr>
<tr>
<td><strong>F1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>0.0206</td>
<td>0.0124</td>
<td>0.0205</td>
</tr>
<tr>
<td>2</td>
<td>N/A</td>
<td>0.0264</td>
<td>0.0179</td>
<td>0.0263</td>
</tr>
<tr>
<td>3</td>
<td>N/A</td>
<td>0.0288</td>
<td>0.0205</td>
<td>0.0294</td>
</tr>
<tr>
<td>4</td>
<td>N/A</td>
<td>0.0301</td>
<td>0.0222</td>
<td>0.0310</td>
</tr>
<tr>
<td>5</td>
<td>N/A</td>
<td>0.0307</td>
<td>0.0231</td>
<td>0.0317</td>
</tr>
<tr>
<td>10</td>
<td>N/A</td>
<td>0.0309</td>
<td>0.0258</td>
<td>0.0325</td>
</tr>
<tr>
<td>15</td>
<td>N/A</td>
<td>0.0291</td>
<td>0.0255</td>
<td>0.0310</td>
</tr>
<tr>
<td>20</td>
<td>N/A</td>
<td>0.0272</td>
<td>0.0244</td>
<td>0.0291</td>
</tr>
<tr>
<td>25</td>
<td>N/A</td>
<td>0.0255</td>
<td>0.0232</td>
<td>0.0271</td>
</tr>
<tr>
<td>30</td>
<td>N/A</td>
<td>0.0239</td>
<td>0.0218</td>
<td>0.0254</td>
</tr>
</tbody>
</table>
learning solution, **Social Relation based Transfer (SORT)**, addresses two fundamental challenges, scalability and sparsity, that are caused by the large-scale social microblogging systems, including the issues of “big data” and “sparse data”. The VIP recommendation task is modeled as a one-class collaborative filtering problem, for which we design a simplified relation-oriented prediction rule to address the “big data” challenge, and propose to transfer knowledge from an auxiliary social network of instant messenger to address the challenge of “sparse data”. Experimental results on large-scale data sets show that the proposed transfer learning solution performs significantly better than a non-transfer learning method and two single chain based methods (SORT-friend and SORT-followee).

For future works, we plan to study the performance of SORT in other large-scale applications, e.g., application-advertisement recommendation in an online zone[^12], multimedia recommendation[^13] in a video streaming system, etc. We are also interested in generalizing the SORT framework via incorporating topological features [114], and more user-side [136], item-side [193] and frontal-side [159] auxiliary data, e.g. user-side social trust, item-side semantic network, and frontal-side heterogeneous user behavior and interaction data, etc.

[^12]: http://qzone.qq.com/
[^13]: http://v.qq.com/
CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

Transfer learning in collaborative filtering is a new and exciting research area, which also has close relationships with real industry applications. In this thesis, we have done the following works,

1. we survey transfer learning works w.r.t. model-based transfer, instance-based transfer and feature-based transfer [157], and collaborative filtering works w.r.t. model-based methods and memory-based methods [26];

2. we give a formal definition of transfer learning in collaborative filtering and categorize the related works according to the auxiliary data from four dimensions of content, context, network and feedback;

3. we propose four new problem settings of movie recommendation and people recommendation, and then design our transfer learning solutions correspondingly,

(a) transfer learning from two-sided implicit feedbacks via coordinate system transfer (CST),

(b) transfer learning from frontal-side binary ratings via transfer by collective factorization (TCF),

(c) transfer learning from frontal-side uncertain ratings via transfer by integrative factorization (TIF), and

(d) transfer learning from user-side heterogeneous social network via social relation based transfer (SORT).

According to the first dimension of “what to transfer” and “how to transfer” in transfer learning [157], and the second dimension of model-based methods and memory-based methods in collaborative filtering [26], we summarize our work and some closely related works of SoRec [138], CMF [193], CBT [115], RMGM [116], WNMCT-F [221], DPMF [3] in Table 7.1.
Table 7.1: Overview of our work in a big picture of transfer learning in collaborative filtering.

<table>
<thead>
<tr>
<th>CF Techniques</th>
<th>Knowledge (what to transfer)</th>
<th>Algorithm style (how to transfer)</th>
<th>Adaptive</th>
<th>Collective</th>
<th>Integrative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model-based</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMF family</td>
<td>Covariance</td>
<td>DPMF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latent features</td>
<td>CST</td>
<td>SoRec, CMF, TCF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constraints</td>
<td></td>
<td>TIF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NMF family</td>
<td>Codebook</td>
<td>CBT</td>
<td></td>
<td>RMGM</td>
<td></td>
</tr>
<tr>
<td>Latent features</td>
<td>WNMCTF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory-based</td>
<td>Social relations</td>
<td>SORT</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We also make a link between transfer learning methods in text mining and transfer learning methods in collaborative filtering from those three major branches of model-based transfer, instance-based transfer and feature-based transfer, which is shown in Table 7.2. We can see that our solutions of CST and TCF belong to feature-based transfer, and TIF and SORT belong to instance-based transfer.

Table 7.2: Overview of our work in a big picture of traditional transfer learning and transfer learning in collaborative filtering.

<table>
<thead>
<tr>
<th>TL Approaches</th>
<th>Text Mining</th>
<th>Collaborative Filtering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model-based Transfer</td>
<td>MTL [57]</td>
<td>CBT, RMGM: cluster-level rating patterns DPMF: covariance matrix (operator)</td>
</tr>
<tr>
<td>Instance-based Transfer</td>
<td>TrAdaBoost [44]</td>
<td>TIF: rating instances</td>
</tr>
<tr>
<td>Feature-based Transfer</td>
<td>TCA [156]</td>
<td>SoRec, CMF, WNMCTF: latent features CST, TCF: latent features</td>
</tr>
</tbody>
</table>

7.2 Future Work

When we develop transfer learning techniques in collaborative filtering, we mainly answer the questions of “what to transfer” and “how to transfer” in transfer learning [157] and try to improve the prediction accuracy in collaborative filtering. We have not considered much about some other important issues for a real recommender system, e.g. the interpretability and diversity of the recommendation result, and not studied the third fundamental question in transfer learning [157], “how to transfer”, theoretically.

In the future, we plan to develop this exciting and fertile interdisciplinary area from two aspects, systems and techniques.
1. **Recommender Systems** We plan to develop real recommender systems with industry partners from mainland China, e.g. people recommendation, APP recommendation and news recommendation with Baidu.

We are particularly interested in developing recommender systems for mobile devices or users. There are two main reasons. First, we are more likely to leverage all four dimensions of auxiliary data for each single user with the help of his or her physical device instead of a user ID. For example, we can obtain the UGC (content), real-time location (context), contact/following list (network), and various actions (feedback). We believe that mobile devices will be a rich auxiliary data source for recommender systems. Recommendation on mobile devices also gives us an opportunity to study the importance of different auxiliary data sources in our general transfer learning framework, either empirically or theoretically. Second, recommender systems are more needed for mobile users, since the power of automatic recommendation is more significant due to the limited operability of the small devices for users. We believe that recommender systems will be a standard feature for a smart phone in the near future.

We are also very interested in combining some specific recommender systems and the existing general search engine. For example, when a user enters a query like “Hotel at Haidian District”, we may integrate some recommendation results from a hotel recommender system into the ranking result of the search engine, which will make the search engine more personalized and accurate. For another example, we may transfer users’ search behaviors on portal search (http://www.baidu.com/) and vertical search (http://map.baidu.com/) into the hotel recommender system, since the users’ history queries on portal search and behaviors on the map may help identify the users’ preferences on hotels.

2. **Transfer Learning Techniques** We are interested in pursuing novel transfer learning techniques used in real recommender systems. In particular, we believe that “online-to-offline”, “large-scale”, “real-time” and “non-negative” are four important characteristics for transfer learning techniques used in a real recommender system.

- **Online-to-Offline (O2O) Transfer** We plan to study on how to transfer users’ online auxiliary data to help recommend users’ offline consumption. For example, we may transfer users’ online behaviors in a certain free
video streaming system to recommend offline non-free movies from some cinemas. To achieve this, we have to design algorithms to bridge the online domain and offline domain via mining the shared knowledge, and also modeling the domain specific user behaviors.

- **Large-Scale Transfer** We plan to study and develop large transfer learning algorithms using both distributed computing techniques and stochastic gradient descent algorithms. We plan to conduct empirical studies using real world industry data with more than several millions of users and thousands (e.g. advertisement) or millions of items (e.g. news stories, queries). To achieve this, we have to design algorithms that can well balance the learning efficiency and the prediction accuracy.

- **Real-Time Transfer** We plan to study on how to identify and extract useful knowledge from real-time auxiliary data (or streaming data) to help recommend items to users. For example, if a user $u$ posts a message of a recent best seller in a microblog or enters a search query for a recent best seller, we then may immediately recommend that book to the user’s friend who is visiting a certain online book store. To address such dynamic problems, we have to design incremental and online transfer learning algorithms, which shall achieve knowledge transfer more adaptively.

- **Non-Negative Transfer** We plan to study on how to avoid negative transfer in real applications, both theoretically and empirically. In particular, we plan to integrate the mechanisms of reinforcement learning, active learning and crowdsourcing into the transfer learning framework, which may help avoid negative transfer to some extent. We are also interested to study the robustness of the transfer learning methods in collaborative filtering: whether negative transfer will happen in cases both with and without spam or manipulations in the target user-item preference data.
REFERENCES


[200] Charles Sutton and Andrew McCallum. Composition of conditional random fields for transfer learning. In *Proceedings of the conference on Human Lan-


[225] Weishi Zhang, Guiguang Ding, Li Chen, and Chunping Li. Augmenting chinese online video recommendations by using virtual ratings predicted by review sentiment classification. In Proceedings of the 2010 IEEE International Conference on...


