TRANSFER LEARNING FOR ONE-CLASS RECOMMENDATION BASED ON MATRIX FACTORIZATION

by

RUIMING XIE

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

June 2013, Hong Kong

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.

_____

RUIMING XIE

18 June 2013

TRANSFER LEARNING FOR ONE-CLASS RECOMMENDATION BASED ON MATRIX FACTORIZATION

by

RUIMING XIE

This is to certify that I have examined the above M.Phil. 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

18 June 2013

ACKNOWLEDGMENTS

The past two years research would be much harder without the guidance of my supervisor, the assistance from my friends and the support of my parents.

First of all, I would like to express my deepest gratitude to my supervisor Prof. Qiang Yang for his kindly instructions on both my research and my life. Without the supervision of Prof. Yang, I would never have been able to explore and to exploit in the area of recommender system.

I would like to show my appreciation to Dr. Wei Xiang, who gave me great help while developing my research topic on selective knowledge transfer. I would also like to thank Yin Zhu, Erheng Zhong, Weike Pan, Lili Zhao, Nathan Liu and Sinno Pan, who have been working closely with me in the past two years and gave valuable suggestions on my study. In addition, I want to thank Dr. Hao Hu, Dr. Vincent Zheng, Kaixiang Mo, Ben Tan, Bin Wu, Naiyan Wang and other fellow students for their companionship.

I am grateful to Prof. Dit-Yan Yeung and Prof. Chi-Wing Wong for serving in the thesis examination committee. And many thanks to the CSE staffs, Mr. Isaac Ma and Ms. Connie Lau, for their administration work.

Finally, I am thankful to my parents for their unreserved support all these years.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

TRANSFER LEARNING FOR ONE-CLASS RECOMMENDATION BASED ON MATRIX FACTORIZATION

by

RUIMING XIE

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

ABSTRACT

One Class Recommender System aims at predicting users' future behaviors according to their historical actions. In these problems, the training data usually contain only binary data reflecting the behavior is happened or not. Thus, the data is sparser than traditional rating prediction problems. Recently, there are two ways to tackle the problem. 1, using knowledge transferred from other domains to mitigate the data sparsity problem. 2, providing methods to distinguish negative data and unlabeled data. However, it's not easy to simply transfer knowledge from source domain to target domain since their observations may be inconsistent. And without data from external source, distinguising negative and unlabeled data is sometimes infeasible.

In this paper, we propose a novel matrix tri-factorization method to transfer the useful information from source domain to target domain. Then we embed this method to a cluster-based SVD(singular value decomposition) framework. In several real-world datasets, we show our method achieve better prediction precision than other state-of-the-art methods. The cluster-based SVD method has been online for 2 months in a online shopping site, and its performance is among the best.

CHAPTER 1

INTRODUCTION

1.1 Motivation

Recommendation systems have become extremely common in recent years, typical recommendation system recommends items (movies, music, books, etc.) that users may be interested in. Collaborative filtering approaches build a model from users' past behavior to predict items that the user may have an interest in. In real-world recommendation systems, users and items are all very large, so users can only rate a small fraction of items. Thus, the user-item matrix can be extremely sparse. What's more, sometimes we can't observe explicit ratings, only implicit feedback is provided(e.g click, pageview and purchase). Such problem may lead to poor performance in CF models.

Recently, different transfer learning methods have been developed to improve the performance of the model.In [17, 18], they use a rating-pattern sharing scheme to share user-item ratings pattern across different domains. In [27, 24], implicit feedback data is available, knowledge is transferred via latent-feature sharing. In [37, 10] they try to exploit correlations among multiple domains. However, most of the methods are develeoped for rating prediction problems. For example, in a music & book rating website, a user can have high or low rating for an album. The ratings are usually trustful, thus can be used to recommend books to the same users. But in a website where only implicit feedback is available(e.g advertisement), the behavior can be much more noisy and with less information. So to acheive better performance, we much transfer more knowledge from source domain while be very careful about the noise.

Some works have been done on solving one-class recommendation problem [14, 22]. They all try to model the freqency of actions by a confidence matrix. For example, if you clicked an item A for 10 times, item B for 1 time. It's more confident that you like A, but not quite sure that you like B. On the other side, if you are a heavy user and you didn't click a popular item A, then it's highly possible that you don't like A. But these works only explore the original matrix, in real-world there are many other useful informations which can be used to improve performance.

We collect several users' clicking and purchasing behaviors from two online shopping site. After taking careful analysis, we find that users' behaviors on clicking and purchasing are similar, but not the same. Based on that, we develop a matrix tri-factorization method(TRIMF) to transfer knowledge from side to side. TRIMF can be used to achieve different goals, (e.g optimize for Ctr(Cvr)).

Further, to make the method online, we develeop a clustering-based matrix factorization method(CBMF) using hadoop. CBMF collect all kinds of user data and convert them into a single matrix per task. For cold-start users, a weighted recommendation from their neighbors will be provided. While for registered users, results are mixed with direct matrix factorization and CBMF.

1.2 Contributions

Our main contributions are summarized as follows:

- First, we find that in implicit datasets, more data must be shared to acheive better performance. To transfer more knowledge, a matrix tri-factorization method is proposed to transfer knowledge from user side and item side(TRIMF).
- Second, implicit datasets can consist many noises. To transfer useful knowledge, we

develop a clustering-pattern transfer function. For each task, a base clustering pattern matrix is provided, the function only do some cluster-level transformation. Thus we can share knowledge more accurately without losing too much information.

- Third, we propose a modified version of TRIMF which can be used for large scale recommendation. And it is used in an Internet company, it's performance is among the best in all online algorithms.

1.3 Thesis Outline

The rest of the thesis is organized as follows: we first provide the background of the research on Transfer Learning, Collaborative Filter and Matrix Factorization in Chapter 2. Then, we discuss the technique grounds of the proposed matrix tri-factorization method in Chapter 3. We present the details of our proposed STLCF framework in Chapter 4 and the experiments in Chapter 5. Finally, we share our thoughts of possible future work and conclude the thesis in Chapter 6.

CHAPTER 2

BACKGROUND

In this chapter, we would like to give a brief review of the related literatures. We classify our work to be most related to the works in the areas of cross-domain collaborative filtering.

In Table 2.1, we summarize the related works under the cross-domain collaborative filtering context. To the best of our knowledge, no previous work for collaborative filtering has ever focused on knowledge transfer between implicit datasets and utilize both latent factor and rating pattern transfer.

In the following, we would like to discuss the state-of-the-art methods for both Collaborative Filtering, Matrix Factorization and Transfer Learning.

Table 2.1: Overview of TRIMF in Cross-Domain Collaborative Filtering context.

	Rating-Pattern Sharing	Latent-Feature Sharing	Other
$Rating \rightarrow Rating$	RMGM [17]	CMF [35]	
$Implicit \rightarrow Rating$		CST [26], TCF [25]	TIF [27]
$Implicit \rightarrow Implicit$	TRIMF		

2.1 Collaborative Filtering

Collaborative filtering ([16], [30]) as an intelligent component in recommender systems ([38], [20]) has gained extensive interest in both academia and industry.

Collaborative filtering(CF) methods are based on collecting and analyzing a large amount of information on users' behaviors, activities or preferences and predicting what users will like in the future based on their similar users. The underlying assumption of the collaborative filtering

approach is that, if a person A has the same opinion as B on an issue, A is more likely to have B's opinion on a different issue x than to have the opinion on x of a randomly chosen person. For example, a collaborative filtering recommendation system for television tastes could make predictions about which television show a user should like given a partial list of this user's tastes (likes or dislikes, ratings, etc).

There are three types of CF: memory-based, model-based and hybrid.

2.1.1 Memory-based CF

This mechanism uses user rating data to compute the similarity between users or items. The similarity is then used for making recommendations. The memory-based method is used in many commercial systems, because it is easy to implement and is effective given plenty of records and doesn't produce a model. Typical examples of this mechanism are neighborhood based CF and item-based/user-based top-N recommendations[36].

The advantages of this approach include:

- The explainability of the results, which is an important aspect of recommendation systems.
- It is easy to setup and use.
- New data can be added easily and incrementally.
- It need not consider contents of the items being recommended.
- The mechanism scales well with co-rated items.

However, there are several disadvantages with this approach:

- It requires plenty of human ratings.

- Its performance decreases when data gets sparse, which is a common phenomenon with web related items.
- Although it can efficiently handle new users, adding new items becomes more complicated since that representation usually relies on a specific vector space. That would require to include the new item and re-insert all the elements in the structure. This prevents the scalability of this approach.

2.1.2 Model-based CF

Models are developed using data mining, machine learning algorithms to find patterns based on training data. This approach has a more holistic goal to uncover latent factors that explain observed ratings. Most of the models are based on creating a classification or clustering technique to identify the users in the test set. Various models have been proposed, including factorization models [16, 27, 28, 30], probabilistic mixture models [13, 15], Bayesian networks [29] and restricted Boltzman machines [32].

There are several advantages with this paradigm:

- It handles the sparsity better than memory based ones.
- This helps with scalability with large data sets.
- It improves the prediction performance.
- It gives an intuitive rationale for the recommendations.

The disadvantage of this approach is the expensive model building. On the one hand, the modern recommendation system usually have petabytes of records as input; On the other hand, the convergence of most models requires intensive computation. One needs to have a tradeoff between prediction performance and scalability.

Given the accuracy of model-based CF, how to overcome the scalability issue has attracted much concern. With the rapid development of parallel computation, researchers have been exploring the use of parallel system to speed up the complex model building. For example in [3], the authors showed that a variety of machine learning algorithms including k-means, logistic regression, naive Bayes, SVM, PCA, gaussian discriminant analysis, EM and backpropagation (NN) could be speeded up by Google's map-reduce [5] paradigm. In [34], the authors showed there is an inverse dependency of training set size and training speed in SVM(linear kernel). That is, if you get more training instances, you can speed up your training speed.

In our method TRIMF, it's hard to parallellize the update step, so we develop a modified version of TRIMF and put it online.

2.1.3 Hybrid models

A number of applications combine the memory-based and the model-based CF algorithms. These overcome the limitations of native CF approaches. It improves the prediction performance. Importantly, it overcomes the CF problems such as sparsity and loss of information. However, they have increased complexity and are expensive to implement. Usually most of the commercial recommender systems are hybrid, for example, Google news recommender system [4].

2.2 Matrix Factorization

In the mathematical discipline of linear algebra, a matrix decomposition or matrix factorization is a factorization of a matrix into a product of matrices. There are many different matrix decompositions; each finds use among a particular class of problems. In CF, usually user-item matrix

is very sparse, we can decompose the original matrix into different low-rank matrix and then recover the dense matrix by multiply the low-rank matrix to produce recommendation results. There are two main methods of matrix factorizaion which are widely applied. 1. Singular value decomposition(SVD), 2. Non-negative matrix factorization(NMF).

2.2.1 Singular Value Decomposition

In linear algebra, the singular value decomposition (SVD) is a factorization of a real or complex matrix, with many useful applications in signal processing and statistics.

Formally, the singular value decomposition of an $m * n$ real or complex matrix M is a factorization of the form $M = U \sum V$, where U is an $m * m$ real or complex unitary matrix, $\sum$ is an $m*n$ rectangular diagonal matrix with non-negative real numbers on the diagonal, and V is an $n * n$ real or complex unitary matrix. Singular value decomposition is used in recommender systems to predict people's item ratings [33]. $\sum$ consists of singular values of M, and we can select the k-biggest values and set other entries of $\sum$ to zero. Then put $M' = U \sum V$, M' is our recommendation result.

2.2.2 Non-negative Matrix Factorizaion

Non-negative matrix factorization (NMF) is a group of algorithms in multivariate analysis and linear algebra where a matrix V is factorized into (usually) two matrices W and H, with the property that all three matrices have no negative elements. It can be regard as a latent factor model [16].

Latent factor models are an alternative approach that tries to explain the ratings by characterizing both items and users on, say, 20 to 100 factors inferred from the ratings patterns. In movie recommendation, the discovered factors might measure obvious dimensions such as comedy versus drama, amount of action, or orientation to children. For users, each factor measures how

much the user likes movies that score high on the corresponding movie factor. For movies, each factor measures the property of that movie.

NMF decompose an original matrix V into two matrices W and H, s.t $V = WH$. V is a $m * n$ matrix, W is a $m * d$ matrix, H is a $d * n$ matrix. Usually $d \ll m, n$, is the dimension of latent factor, NMF methods put users and items into one common latent space. When judging whether a user likes an item, we can simply calculate by inner product.

2.2.3 Non-negative Matrix Tri-factorizaion

As a transformation of NMF, Non-negative Matrix Tri-factorizaion(NMTF) decompose a matrix X into three non-negative part : $X = USV$. Instead of mapping users and items to a same latent space, the three parts of NMTF can be interpreted as:

- U users' soft-clustering matrix
- S users' clusters vs items' clusters(cluster relationship matrix)
- V items' soft-clustering matrix

In [7], the authors proved that NMF is equivalent to k-means clustering. In [6] the authors also proved that NMTF can be regarded as a way of clustering. NMTF is well known in document processing, [19] uses prior knowledge in lexical and NMTF to tackle the sentiment analysis problem, [39] exploits the relationship between word clusters and document classes in text classification problem.

Because the property of NMTF, if we get some prior knowledge(e.g word cluster, document class), we can easily adopt them in the model. Thus can acheive better performance than tradictional NMF methods. Our method(TRIMF) uses NMTF to leverage auxiliary data, align cluster and do cluster-level sharing. NMTF is also very common in the field of transfer learning, where clusters can be shared across different domains.

2.3 Transfer Learning

Pan and Yang [23] surveyed the field of transfer learning. A major assumption in many machine learning and data mining algorithms is that the training and future data must be in the same feature space and have the same distribution. However, in many real-world applications, this assumption may not hold. For example, we have a task in recommedation, users and items form a joint distribution in training data. But in test data, users may be different with the training as well as items, their relationship may varies too. Thus the latter data has different feature spaces or distribution than the training data. In such cases, knowledge transfer, if done successfully, would greatly improve the performance of learning by avoiding much expensive data-labeling effort.

2.3.1 Transfer Learning for Collaborative Filtering

Some works on transfer learning are in the context of collaborative filtering. Mehta and Hofmann [21] consider the scenario involving two systems with shared users and use manifold alignment methods to jointly build neighborhood models for the two systems. They focus on making use of an auxiliary recommender system when only part of the users are aligned, which does not distinguish the consistency of users' preferences among the aligned users. [35] designed a collective matrix factorizaion framework, where two matrix M, N are factorized into $M = UV^T$, $N = US^T$. The sharing part U can be a bridge to transfer knowledge from M to N(or N to M), base on that, there are some following work in cross-domain collaborative filtering using matrix factorization tecnique. Li *et al.* [18] designed a regularization framework to transfer knowledge of cluster-level rating patterns, they use matrix tri-factorization and cluster level rating patterns are shared. Pan *et al.* [25], [26] used a matrix factorization framework to

transfer knowledge in latent feature space. Knowledge is transfered from an implicit feedback dataset to a rating dataset, but this method can deal with knowledge transfer in both implicit domains. Cao *et al.* [2] exploited correlations among different CF domains via learning. E.g we factorize each matrix X_d by $X_d = F_d G_d^T$ where F_d and G_d are the user and the item latent features, respectively. This approach tries to explore the correlations between user domains F_d and/or item domains G_d and the knowledge can be transferred across domains through the correlation matrices.

Our method(TRIMF) leverage rating pattern sharing and latent feature sharing carefully by designing a matrix tri-factorization framework. TRIMF can handle knowledge transfer from implicit(explicit) dataset A to implicit(explicit) dataset B. Also, TRIMF can be set to suit different tasks.

2.3.2 Large Scale Transfer Learning

So far, transfer learning has been mostly considered in the off-line learning settings, which do not emphasize the scalability and computation speed. Due to the rapid development of storage technique and flourish of internet services, the real world problems in recent recommendation systems are mostly based on some large data sets. Little work on large scale transfer learning has been published in previous literature, though it is badly desirable. To cope with the growing needs of today's recommendation system, we would like to discover the parallelizing possibility in our experiments. There are already some researchers working on the large scale collaborative filtering, [4] designed a map-reduce framework for online news recommendation. In our approach, we investigate the parallel framework and put them on an online shopping site.

Map-Reduce Framework

MapReduce is a framework for processing parallelizable problems in huge datasets using a large number of computers (nodes). A MapReduce program comprises a Map() procedure that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name) and a Reduce() procedure that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies).

- **"Map" step:** The master node takes the input, divides it into smaller sub-problems, and distributes them to worker nodes. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes the smaller problem, and passes the answer back to its master node.
- **"Sort" step:** The master node sort all key-value pairs according to their key. Thus in the reduce step, same key appears sequentially.
- **"Reduce" step:** The master node then collects the answers to all the sub-problems and combines them in some way to form the output, i.e. the answer to the problem it was originally trying to solve.

We will show that our methods(modified version of TRIMF) can be plugged into the Map-Reduce framework for parallelization.

CHAPTER 3

TRANSFER LEARNING IN ONE CLASS CF FOR SHOPPING PREDICTION

3.1 Problem settings

3.1.1 Background

In real-world, a person usually has different kinds of behaviour before buying one thing. For online shopping sites, Their goal is to let users buy their products, but the user-item matrix for deal is extremely sparse(less than 0.001%). So if we only use the information of deal, we can't acheive good even reasonable performance. In Yixun(Tencent's online shopping site), there are two main actions - click and purchase, both consists only binary data(1-action happened, 0-unknown). We know that deal matrix X_d is very sparse, althrough click matrix X_c is also sparse, but is much denser than X_d. So we developed a transfer learning algorithm(TRIMF) that leverage these data to predict a user's future purchasing. Compared with former methods which shares rating patterns or latent features only, our method shares both rating patterns and latent features throught cluster-level transform and overlapping matrix. Experiments show that my algorithm performs better then other baseline(Transfer) methods.

3.1.2 Problem definition

- Input: [0-1 matrix:user click matrix $X_C(m_c * n_c)$, user deal matrix $X_d(m_d * n_d)$] , m_c, m_d denote the number of users, n_c, n_d denote the number of items. Users and items are partially shared.

- Output: Two prediction matrix $P_C(m_c * n_c), P_d(m_d * n_d)$, which predict users' purchasing behaviour.

3.2 TRIMF

Former one-class CF methods use weighted matrix factorization to tackle the problem that all observed rating are 1. In this method we adopt this weighting scheme to give missing values proper weights.

User's deal data is very sparse, e.g users will buy n_d items one day while click n_c items. Then $n_d << n_c$. So only use deal data is not sufficient.

We want to use users' click data to learn a better cluster-level rating pattern S, compared with only using users' purchase data. But what a user like to click is not always the item he wants to buy. So these rating pattern should be somehow related but not the same. In Yixun, there are only 4 common items in top-20 clicked items and top-20 purchased items. So we can't simply apply the same pattern in prediction.

There are some items with higher conversion rate(user tends to buy after clicking), but some items not. And there are some users who like window-shopping while others will buy it right after clicking. These are all cluster-level features, so I decide to learn a mapping function to let the learnt S suit data better.

Finally we use a weighted non-negative matrix tri-factorization method to deal with the problem.

Objective Function:

$$min_{F,G,S,U,V} W_c \odot ||X_c - (F; F_c)S(G; G_c)'||_2 + W_d \odot ||X_d - (F; F_d)(USV)(G; G_d)'||_2 + (regularization)$$

- W_c, W_d is the weight for X_C, X_d, every observed entry has weight 1 while others have

weight 0.5.

- F, G is the soft clustering result matrix for overlapped users(items), they are forced to have the same cluster distributions. Others are unique users(items).

- U, V are two diagonal matrix, U_{ii} scales every S_{i*} to $U_{ii}S_{i*}$, it models the users' tranform from click to deal. While V_{jj} scales every S_{*j} to $S_{*j}V_{jj}$, it models the items' tranform from click to deal.

- When predicting, we use $(F; F_d)(USV)(G; G_d)$ to predict users who have deal data. And since we got two mapping matrix U, V, we apply U, V back to click matrix to predict users who have click data, i.e we use $(F; F_c)(USV)(G; G_c)$ to predict.

3.3 Solution to the problem & Algorithm

We use an alternately iterative algorithm to solve the objective function.We use F_1, F_2 to denote $(F; F_c)$ and $(F; F_d)$, G_1, G_2 to denote $(G; G_c)$ and $(G; G_d)$

The update rules for S, U, V are simple, here we decribe the update rules for F and F_c:

$$F = F.*\sqrt{\frac{Ifc' * W_c * X_c * G_1 * S' + Ifd' * W_d * X_d * G_2 * S'}{Ifc' * W_c * ((Ifc * F + Ifcc) * F_c * S * G_1) * G_1 * S + Ifd' * W_d * ((Ifd * F_f + Ifdd * F_d) *}}$$

$$F_c = F_c. * \sqrt{\frac{Ifc' * W_c * X_c * G_1 * S'}{Ifcc' * (W_c * ((Ifc * F + Ifcc * F_c) * S * G_1')) * G_1 * S' + a * F_c}}$$

We define four block matrix $Ifc, Ifcc, Ifd, Ifdd$ that $(Ifc, Ifcc) * (F; F_c) = I * F_1$ and $(Ifd, Ifdd) * (F; F_d) = I * F_2$

The user-item matrix is typically very sparse with $z << nm$ non-zero entries while k is typically also much smaller than n, m. By using sparse matrix multiplications and avoiding dense intermediate matrices, the updates can be very efficiently and easily implemented. In

particular, updating F, S, G each takes $O(k^2(m+n)+kz)$, and the algorithm usually reach convergence in less than 200 iterations.

—

Algorithm:

```
Init F,G,S,U,V
Set overlap numbers for users and items
for i = 1 to k
    update F,G,S
    update U,V
output F,G,S,U,V
```

3.4 Experiment

3.4.1 Datasets

Yixun short term data: we got users' click & deal data from date 20130801 to 20130814, first week used as training data, second week used as test data.Click matrix: 16240 users, 1932 items. Deal matrix: 2520 users, 1791 items. There are 2029 overlapped users and 1642 overlapped items.

Yixun long term data: we trace 1k users' 6 months' activity, there are 6k items clicked and 2k items boughted. We select the last 5 deal activities as test data, others as training data. 1800 items are overlapped.

3.4.2 Metrics

We use prec@5 and prec@10 as evaluation meatures.

3.4.3 Methods

For all non-transfer methods, we try 3 training matrix:deal, click, deal+click, and report their best performace. We choose parameters by cross validation.

- non-transfer:
  - Most Popular
  - SVD:Pure SVD
    - rank = {5,10,20,30,40,50}
  - NMF
    - rank = {10,20,40,60,100}
  - PMF:Probabilistic Matrix Factorization
    - rank = {10,20,30,40,50}
  - BPRMF:Bayesian Personalized Ranking from Implicit Feedback
    - We initialized BPR with most popular results.
    - We set $iteration = \#n * 100$, ($\#n$ in the number of observations)
  - WRMF:One-class collaborative filtering
    - rank = {5,10,15,20,25}
- transfer:
  - CMF:Collective Matrix Factorization
    - Shared latent space = {5,10,15,20,25}
    - For each data, we make two matrix the same dimension(in order to share a latent factor) by adding zeroes rows & columns.
  - TCF: Transfer Learning to Predict Missing Ratings via Heterogeneous Feedbacks

Method	Prec@5	Prec@10
Most Popular	0.0323	0.0289
SVD	0.0438	0.0367
NMF	0.0403	0.0324
PMF	0.0435	0.0372
BPRMF	0.0444	0.0364
WRMF	0.049	0.0403
CMF	0.0436	0.0350
TCF	0.0453	0.0369
TRIMF	**0.0525**	**0.0410**

Table 3.1: Comparison on user who have deal data.

  - We set the deal matrix with random sampled zeros as the rating matrix, click matrix as the implicit feed back matrix.
  - We make matrix the same dimension by adding zeroes rows & columns.

- TRIMF: My method
  - We set latent factor = 30, iter = 200, λ = 0.001.

3.4.4 Result

Yixun short term data

Since the user overlap of deal and click matrix are small, so we perform two test, one on deal matrix X_d and one on click matrix X_c.

The main result are showed in Table 1 and Table 2.

Yixun long term data

Since the user are manually selected, we only test X_d. The result is showed in Table 3.

Method	Prec@5	Prec@10
Most Popular	0.0090	0.0085
SVD	0.0123	0.00113
NMF	0.0091	0.0089
PMF	0.0121	0.0112
BPRMF	0.0142	0.0130
WRMF	0.0174	0.0144
CMF	0.0176	0.0139
TCF	0.0158	0.0127
TRIMF	**0.0189**	**0.0153**
TRIMF(without remap)	0.0175	0.0146

Table 3.2: Comparison on user who have click data.

Method	Prec@5	Prec@10
Most Popular	0.00508	0.00405
SVD	0.00453	0.00413
NMF	0.00401	0.00389
PMF	0.00421	0.00312
BPRMF	0.00542	0.00430
WRMF	0.00485	0.00345
CMF	0.00512	0.00432
TCF	0.00534	0.00502
TRIMF	**0.00720**	**0.00606**

Table 3.3: Comparison on yixun long term data.

Method	Prec@5	Prec@10
shareFG	**0.0436**	**0.0350**
not share	0.0335	0.0306
random share	0.0344	0.0299

Table 3.4: The effect of mappig UV

3.5 Result analysis

3.5.1 The effects of mapping UV

Mapping UV aims at the difference between hot click items and hot deal item. We check the value of V and see item clusters that have higher values are items that people tends to buy after clicking, e.g toothbrush, snacks. While lower value of V more reflects that items are popular, e.g cell phones, laptops.

3.5.2 The effects of sharing

To see that sharing F, G really works, I select another 6000users to test, I tried 'not share','share' and 'random share', the prec@n here are not comparable with the first experiment(Table 4).

CHAPTER 4

SELECTIVE TRANSFER LEARNING FOR COLLABORATIVE FILTERING

4.1 Illustration of the STLCF settings

A common assumption in previous transfer learning literature under the context of recommendation system is that the knowledge in source domain is always helpful in target domain tasks. In other words, they assume that all source domains have equally positive impacts on the target domain and the user or item preferences (depending on what is shared between the target domain and the source domains) are consistent in both source and target domains. Such an assumption is widely adopted in several works, such as Collective Matrix Factorization (CMF) [35] and its extensions.

Clearly, the above assumption is too strong for many real-world applications. Take the ratings of traditional Chinese music for example. In a Chinese local music rating web site, Chinese users may be critical for the traditional Chinese music; while in an international music rating web site, the ratings on those traditional Chinese music could be diverse due to the culture differences: those users with Chinese culture background would give trustful ratings, others could be inaccurate. Now, suppose there is a new Chinese local music rating web site, which is the target domain, and we try to transfer knowledge from both the existing Chinese local web site and the international web site to help the target domain rating prediction task. In this case, it would not be wise to treat the data from these two existing web sites equally. In addition, it would cause negative transfer if we use the entire data sets from the international Web sites

without selection. Therefore, we should carefully analyze the source domain at both domain level and the instance level before applied to the target domain.

4.2 Logic behind STLCF

As we have discussed, using the source domain data without selection may harm the target domain learning. By proposing the selective knowledge transfer with the novel factors (empirical error and variance of empirical error), we come up with the details of Selective Transfer Learning framework for CF in this section.

As illustrated in the second example in Figure **??** where the domains have mutual user set, we would like to transfer knowledge of those items' records that consistently reflect the user's preferences. Because of our finding that the consistent records have small empirical error and variance, the selection shall consider these two factors. We embed these two factors into a boosting framework, where the source data with small empirical error and variance receive higher weights since they are consistent with the target data. This boosting framework models the cross-domain CF from two aspects:

- on one hand, we take more care of those mis-predicted target instances, following the traditional boosting logic;
- on the other hand, we automatically identify the consistency of the source domains during the learning process and selective use those source domains with more trustful information.

Based on the above consideration, our boosting based selective transfer learning framework for the collaborative filtering (STLCF) focuses on two levels: both domain level and instance level. At the domain level, STLCF boosts each source domain based on a notion of transferability proposed by Eaton *et al.* [9]. STLCF increases the weights of all instances from a source

Algorithm 1: Algorithm for Selective TLCF.

Input: $\mathbf{X}^d$, $\mathbf{X}^s$, T
$\mathbf{X}^d \in \mathbb{R}^{m \times n^d}$: the target training data
$\mathbf{X}^s \in \mathbb{R}^{m \times n^s}$: the auxiliary source data
G: the weighted TLCF model wTGPLSA
T: number of boosting iterations
Initialize: Initialize $\mathbf{w}^s : w_i^s \leftarrow \frac{1}{n^s}$, $\mathbf{w}^d : w_i^d \leftarrow \frac{1}{n^d}$
for $iter$ = 1 to T **do**
  Step 1: Apply G to generate a weak learner $G(\mathbf{X}^d, \mathbf{X}^s, \mathbf{w}^d, \mathbf{w}^s)$ that minimize Eq.(**??**)
  Step 2: Get weak hypothesis for both the d and s domains $h^{iter} : \mathbf{X}^d, \mathbf{X}^s \rightarrow \hat{\mathbf{X}}^d, \hat{\mathbf{X}}^s$
  Step 3: Calculate empirical error E^d and E^s using Eq.(4.4)
  Step 4: Calculate fitness weight β^{s_k} for each source domain s_k using Eq.(4.11)
  Step 5: Choose model weight α^{iter} via Eq.(4.7)
  Step 6: Update source item weight $\mathbf{w}^s$ via Eq.(4.10)
  Step 7: Update target item weight $\mathbf{w}^d$ via Eq.(4.9)
end for
Output: Hypothesis $Z = H(\mathbf{X}^{(d)}) = \Sigma_{t=1}^{T} \alpha^t h^t(\mathbf{X}^{(d)})$

domain if the source domain shows positive transferability, and decreases all weights if the source domain shows negative transferability. Simultaneously at instance level, STLCF adaptively increases the weight of mis-predicted instances to emphasize learning on those "tough" instances. Effectively, this adjusts the distribution of instance weights within each source domain. A formal description of the framework is given in Algorithm 1.

4.3 Selective Transfer Learning for Collaborative Filtering

4.3.1 Algorithm for STLCF

As shown in Algorithm 1, in each iteration, we apply base model TGPLSA over weighted instances to build a weak learner $G(\cdot)$ and hypothesis h^{iter}. Then to update the source and target item weights, domain level fitness weight β^{s_k} is chosen for each source domain s_k based on domain level consistency [9]. And α^{iter} for base model is also updated, considering empirical errors and variances. Accordingly, the weights of mis-predicted target items are increased and the weights of those less helpful source domains are decreased in each iteration. The final ensemble is given by an additive model, which gives larger weights to the hypotheses with

lower errors. We provide a detailed derivation of STLCF in the rest of this section.

4.3.2 Derivation of STLCF

Target Function of a Single Weak Learner in STLCF

In previous works in collaborative filtering, the mean absolute error (MAE) is usually chosen as the loss function. In addition to the MAE loss, if we tolerate some prediction error τ, we define:

$$l_1(\mathbf{X}_{*i}, \hat{\mathbf{X}}_{*i}) = \begin{cases} -1, & \sum_{x_{ui} \in \mathbf{X}_{*i}} |\hat{x}_{ui} - x_{ui}| \leq \tau \cdot nnz(\mathbf{X}_{*i}) \\ +1, & \sum_{x_{ui} \in \mathbf{X}_{*i}} |\hat{x}_{ui} - x_{ui}| > \tau \cdot nnz(\mathbf{X}_{*i}) \end{cases} \tag{4.1}$$

where $nnz(\cdot)$ is the number of observed ratings. $\mathbf{X}_{*i}$ and $\hat{\mathbf{X}}_{*i}$ denote the true values and predictions respectively. We may also define the item level MAE error for target domain with respect to τ as:

$$\epsilon_i^d = l_1(\mathbf{X}_{*i}^d, \hat{\mathbf{X}}_{*i}^d) \tag{4.2}$$

To facilitate the optimization, we consider the following exponential loss for empirical risk minimization:

$$l_2(i) = l_2(\mathbf{X}_{*i}^d, \hat{\mathbf{X}}_{*i}^d) = e^{\epsilon_i^d} \tag{4.3}$$

As stated in previous section, the lower variance of empirical errors can provide more confident consistency estimation, we combine these factors and reformulate the loss function:

$$\mathcal{L} = \sum_{i=1}^{n^d} l_2(i) + \gamma \sqrt{\sum_{i>j}^{n^d} (l_2(i) - l_2(j))^2} \tag{4.4}$$

Above all, the model minimize the above quantity for some scalar $\gamma > 0$:

Additive Ensemble of Weak Learners

Assume that the function of interest $\mathcal{H}$ for prediction is composed of the hypothesis h^t from each weak learner. The function to be output would consist of the following additive model over the hypothesis from the weak learners:

$$\hat{x}_{ui}^d = f(x_{ui}^d) = \sum_{t=1} \alpha^t h^t(x_{ui}^d) \tag{4.5}$$

where $\alpha^t \in \mathbb{R}^+$.

Since we are interested in building an additive model, we assume that we already have a function $h(\cdot)$. Subsequently, we derive a greedy algorithm to obtain a weak learner $G^t(\cdot)$ and a positive scalar α^t such that $f(\cdot) = h(\cdot) + \alpha^t G^t(\cdot)$.

Derivation of Weak Learners

In the following derivation, for the convince of presentation, we omit the model index t, and use G to represent G^t, α to represent α^t.

By defining $\gamma_1 = (1 + (n-1)\gamma)$, $\gamma_2 = (2 - 2\gamma)$, α, $w_i^d = e^{l_1(h(\mathbf{X}_{*i}^d), \mathbf{X}_{*i}^d)}$ and $G_i^d = l_1(G(\mathbf{X}_{*i}^d), \mathbf{X}_{*i}^d)$, Eq.(4.4) can be equivalently posed as optimizing the following loss with respect to α:

(4.6)

$$\begin{aligned}\mathcal{L} \;=\; & \gamma_1\Big(\sum_{i\in I}(w_i^d)^2 e^{2\alpha} + \sum_{i\in J}(w_i^d)^2 e^{-2\alpha}\Big) + \gamma_2 \sum_{i>j:i,j\in I}^{n^d} w_i^d w_j^d e^{2\alpha} \\ & + \gamma_2 \sum_{i>j:i,j\in J}^{n^d} w_i^d w_j^d e^{-2\alpha} + \gamma_2 \sum_{i>j:i\in I,j\in J or i\in J,j\in I}^{n^d} w_i^d w_j^d\end{aligned}$$

For brevity, we define the following sets of indices as $I = \{i : G_i^d = +1\}$ and $J = \{i : G_i^d = -1\}$. Here J denotes the set of items whose prediction by $G(\cdot)$ falls into the fault tolerable

range, while I denotes the rest set. By making the last transformation in Eq.(4.6) equal to zero, we get:

$$\alpha = \frac{1}{4}\log\left(\frac{(1-\gamma)(\sum_{i\in I} w_i^d)^2 + \gamma n^d \sum_{i\in I}(w_i^d)^2}{(1-\gamma)(\sum_{i\in J} w_i^d)^2 + \gamma n^d \sum_{i\in J}(w_i^d)^2}\right) \tag{4.7}$$

If we set $\gamma = 0$, then it is reduced to the form of AdaBoost:

$$\alpha = \frac{1}{4}\log\left(\frac{(\sum_{i\in I} w_i^d)^2}{(\sum_{i\in J} w_i^d)^2}\right) = \frac{1}{2}\log\left(\frac{(\sum_{i\in I} w_i^d)}{(\sum_{i\in J} w_i^d)}\right) \tag{4.8}$$

Finally, the updating rule for w_i^d is

$$w_i^d \leftarrow w_i^d e^{(-\alpha G_i^d)} \tag{4.9}$$

And for the instance weight w_i^d in the source domain, we can also adopt the similar updating rule in Eq.(4.9).

Other than the instance level selection discussed above, we also want to perform the domain level selection to penalize those domains that are likely to be irrelevant, so that the domains with more relevant instances speak loudly. Following the idea of task-based boosting [8], we further introduce a re-weighting factor β for each source domain to control the knowledge transfer. So we formulate the updating rule for w_i^s to be:

$$w_i^s \leftarrow w_i^s e^{(-\alpha G_i^s - \beta)} \tag{4.10}$$

where β can be set greedily in proportion to the performance gain of the single source domain transfer learning:

$$\beta = \frac{\sum w_i^d(\varepsilon_i - \vec{\varepsilon_i})}{||\mathbf{w}^d||_1} \tag{4.11}$$

where ε_i is the training error of the transfer learning model, and $\vec{\varepsilon_i}$ is the training error of the non-transfer learning model, which utilizes only the observed target domain data.

4.3.3 STLCF framework as a Whole

To sum up, STLCF is a boosting based ensemble method, using certain generative single models as weak learner. We have seen an instance of generative model (TGPLSA) in Chapter **??**. The ensemble function has been given as an weighted sum of several weak learners. We also presented the detailed update rules for each weak learner in Section 4.3.2.

CHAPTER 5

EXPERIMENTS

5.1 Data Sets and Experimental Settings

We evaluate the proposed method on four data sources: Netflix[1], Douban, IMDB[2], and Wikipedia[3] user editing records. The Netflix rating data contains more than 100 million ratings with values in $\{1, 2, 3, 4, 5\}$, which are given by more than 4.8×10^5 users on around 1.8×10^4 movies. Douban contains movie, book and music recommendations, with rating values also in $\{1, 2, 3, 4, 5\}$. IMDB hyperlink graph is employed as a measure of similarity between movies. In the graph, each movie builds links to its 10 most similar movies. The Wikipedia user editing records provide a $\{0, 1\}$ indicator of whether a user concerns or not about a certain movie.

The data sets used in the experiments are described as follows. For Netflix, to retain the original features of the users while keeping the size of the data set suitable for the experiments, we sampled a subset of $10,000$ users. In Douban data sets, we obtained 1.2×10^6 ratings on 7.5×10^3 music, 5.8×10^5 ratings on 3.5×10^3 books, and 1.4×10^6 ratings on 8×10^3 movies, given by 1.1×10^4 users. For both the IMDB data set and the Wikipedia data set, we filtered them by matching the movie titles in both the Netflix and the Douban Movie data sets. After pre-processing, the IMDB hyperlink data set contains $\sim 5 \times 10^3$ movies. The Wikipedia user editing records data set has 1.1×10^6 editing logs by 8.5×10^3 users on the same $\sim 5 \times 10^3$

[1] http://www.netflix.com

[2] http://www.imdb.com

[3] http://en.wikipedia.org

Table 5.1: Datasets in our experiments.

Notation	Data Set	Data Type	Instances No.
D1	Douban Music	Rating [1,5]	1.2×10^6
D2	Douban Book	Rating [1,5]	5.8×10^5
D3	Douban Movie	Rating [1,5]	1.4×10^6
D4	Netflix	Rating [1,5]	1.8×10^4
D5	Wikipedia	Editing Log	1.1×10^6
D6	IMDB	Hyperlink	5.0×10^3

movies as IMDB data set. To present our experiments, we use the shorthand notations listed in Table 5.1 to denote the data sets.

We evaluate the proposed algorithm on five cross-domain recommendation tasks, as follows:

- The first task is to simulate the cross-domain collaborative filtering, using the Netflix data set. The sampled data is partitioned into two parts with disjoint sets of movies but identical set of users. One part consists of ratings given by $8,000$ movies with 1.6% density, which serves as the source domain. The remaining $7,000$ movies are used as the target domain with different levels of sparsity density.
- The second task is a real-world cross-domain recommendation, where the source domain is Douban Book and the target domain is Douban Movie. In this setting, the source and the target domains share the same user set but have different item sets.
- The third task is on Netflix and Douban data. We extract the ratings on the $6,000$ shared movies from Netflix and Douban Movie. Then we get 4.9×10^5 ratings from Douban given by 1.2×10^4 users with density 0.7%, and 10^6 ratings from Netflix given by 10^4 users with density 1.7%. The goal is to transfer knowledge from Netflix to Douban Movie. In this task, item set is identical across domains but user sets are totally different.
- The fourth task is to evaluate the effectiveness of the proposed algorithm under the context of multiple source domains. It uses both Douban Music and Douban Book as the source

Table 5.2: Prediction performance of STLCF and the baselines.

Datasets	Source sparseness	Target sparseness	Non-TL		Non-Selective TL		**Selective TL**	
			GPLSA	PMF	TGPLSA	CMF	STLCF(E)	STLCF(EV)
D4(Simulated) to D4(Simulated)	1.6%	0.1%	1.0012	0.9993	0.9652	0.9688	0.9596	**0.9533**
		0.2%	0.9839	0.9814	0.9528	0.9532	0.9468	**0.9347**
		0.3%	0.9769	0.9728	0.9475	0.9464	0.9306	**0.9213**
D2 to D3	1.5%	0.1%	0.8939	0.8856	0.8098	0.8329	0.7711	**0.7568**
		0.2%	0.8370	0.8323	0.7462	0.7853	0.7353	**0.7150**
		0.3%	0.7314	0.7267	0.7004	0.7179	0.6978	**0.6859**
D4 to D3	1.7%	0.1%	0.8939	0.8856	0.8145	0.8297	0.7623	**0.7549**
		0.2%	0.8370	0.8323	0.7519	0.7588	0.7307	**0.7193**
		0.3%	0.7314	0.7267	0.7127	0.7259	0.6982	**0.6870**

domains and transfer knowledge to Douban Movie domain.

- The fifth task varies the type of source domains. It utilizes the Wikipedia user editing records and IMDB hyperlink graph, together with Douban Movie as the source domains to perform rating predictions on the Netflix movie data set.

For evaluation, because we adopt the error rate style objective function in our optimization, we calculate the Root Mean Square Error (RMSE) on the heldout $\sim 30\%$ of the target data:

$$RMSE = \sqrt{\sum_{(u,i,x_{ui})\in T_E} (x_{ui} - \hat{x}_{ui})^2 / |T_E|}$$

where x_{ui} and $\hat{x}_{ui}$ are the true and predicted ratings, respectively, and $|T_E|$ is the number of test ratings.

5.2 STLCF and Baselines Methods

We implement two variations of our STLCF method. STLCF(E) is an STLCF method that only take training error into consideration when performing selective transfer learning. STLCF(EV) not only considers training error, but also utilizes the empirical error variance.

To demonstrate the significance of our STLCF, we selected the following baselines:

- **PMF** [31] is a recently proposed method for missing value prediction. Previous work showed that this method worked well on the large, sparse and imbalanced data set.
- **GPLSA** [12] is a classical non-transfer recommendation algorithm.
- **CMF** [35] is proposed for jointly factorizing two matrices. Being adopted as a transfer learning technique in several recent works, CMF has been proven to be an effective cross-domain recommendation approach.
- **TGPLSA** is an uniformly weighted transfer learning model, which utilize all source data to help build the target domain model. It is used as one of the baselines because we adopt it as the base model of our boosting-based selective transfer learning framework.

Parameters for these baseline models are fine-tuned via cross validation.

5.3 Experimental Results

5.3.1 Performance Comparisons

We test the performance of our STLCF methods against the baselines. The results of the collaborative filtering tasks under three different target domain sparseness are shown in Table 5.2.

First, we observe that the non-transfer methods, i.e. GPLSA and PMF, fail to give accurate predictions, especially when the target domain is severely sparse. With the help of source domains, the (non-selective) transfer learning methods with equally weights on the source domains, like TGPLSA and CMF, can increase the accuracy of the rating predictions. And our selective transfer learning methods (i.e., STLCF(E) and STLCF(EV)) can do even better. The fact that our STLCF outperforms others is expected because by performing the *selective* knowledge transfer, we use the truly helpful source domain(s), which is designed to handle the sparseness issue in CF problems.

Second, comparing the two non-selective TLCF methods with the other two selective TLCF methods, we observe that on the last two real world tasks (D2 to D3 and D4 to D3) when the target domain is extremely sparse (say 0.1%), the improvement of accuracy achieved by our STLCF methods against the non-selective transfer learning methods is more significant than it does on the simulation data set based on Netflix (D4 to D4). Notice that the inconsistency of the target domain and the source domains on the simulation data sets is much smaller than that on the real-world cases. The experiment results show that our STLCF algorithm is effective in handling the inconsistency between the sparse target domain and the source domains.

Third, we notice that some factors, like empirical error variance, may affect the prediction. In Table 5.2, we compare our two STLCF methods, i.e., STLCF(E) and STLCF(EV) when the target domain sparsity is 0.1%. We can find that on the task "D2 to D3", i.e., Douban Book to Movie, STLCF(EV) is much better than STLCF(E). But on the task "D4(Simulated) to D4(Simulated)", the improvement of STLCF(EV) is not so significant against STLCF(E). These observations may be due to the domain consistency. For the tasks "D4(Simulated) to D4(Simulated)", both the source and target entities are movie ratings from Netflix data set, while the task "D2 to D3" tries to transfer the knowledge from a book recommendation system to the movie recommendation system, which may contain some domain specific items. When the target domain is very sparse, i.e. the user's ratings on the items are rare, there are chances to get high prediction accuracy occasionally on the observed data with a bad model on the source domains that are inconsistent with target domain. In this case, it is important to consider the variance of empirical error as well. Comparing to STLCF(E), STLCF(EV), which punishes the large variance, can better handle the domain inconsistency in transfer learning, especially when the target domain is sparse.

Table 5.3: Prediction performance of STLCF for Long-Tail Users on the D2 to D3 task.

Ratings per user	Non-TL	Non-Selective TL		**Selective TL i.e. STLCF**	
	GPLSA	TGPLSA	CMF	(E)	(EV)
1-5	1.1942	0.9294	0.9312	0.8307	**0.8216**
6-10	0.9300	0.7859	0.7929	0.7454	**0.7428**
11-15	0.8296	0.7331	0.7390	**0.7143**	0.7150
16-20	0.7841	0.7079	0.7113	**0.7042**	0.7050
21-25	0.7618	0.6941	0.6947	0.6942	**0.6910**
26-30	0.7494	0.6918	0.6884	0.6917	**0.6852**
31-35	0.7370	0.6909	0.6911	0.6915	**0.6818**
36-40	0.7281	0.6896	0.6856	0.6907	**0.6776**
41-45	0.7219	0.6878	0.6821	0.6890	**0.6740**
46-50	0.7187	0.6881	0.6878	0.6800	**0.6734**

5.3.2 Results on Long-Tail Users

To better understand the impact of STLCF with the help of the source domain, we conduct a fine-grained analysis on the performance improvement on Douban data sets, with Douban Book as source domain and Douban Movie as target domain. The results on different user groups in the target domain are shown in Table 5.3. In the table, method STLCF(E) does not punish the large variance of empirical error, while STLCF(EV) does. First, we observe that the STLCF models, i.e., STLCF(E) and STLCF(EV) can achieve better results on those long-tail users who have very few ratings in historical logs. Such fact implies that our STLCF methods could handle the long-tail users that really need a fine-grained analysis when performing knowledge transfer from source domains. Current CF models without any fine-grained analysis on the specific

Table 5.4: Prediction performance of STLCF with multiple source domains containing much irrelevant information.

Source Domain:		None	D3	D3 & D5	D3 & D6	D5 & D6	D3 & D5 & D6
Target (D4) sparseness	0.1%	0.9983	0.9789	0.9747	0.9712	0.9923	**0.9663**
	0.2%	0.9812	0.9625	0.9583	0.9572	0.9695	**0.9505**
	0.3%	0.9703	0.9511	0.9409	0.9464	0.9599	**0.9383**

Table 5.5: Prediction performance of STLCF with multiple source domains (Douban).

Source Domain:		None	D1	D2	D1 & D2
Target (D3) sparseness	0.1%	0.8856	0.7521	0.7568	**0.7304**
	0.2%	0.8323	0.7163	0.7150	**0.6904**
	0.3%	0.7267	0.6870	0.6859	**0.6739**

users usually fail to capture the preferences of the long-tail users, while our STLCF methods work well because they can selectively augment the weight of the corresponding source domain instances with respect to those long-tail cases at both instance level and domain level. Second, STLCF(EV) works better than STLCF(E) on those non-long-tail users, i.e., with more than 25 ratings per user in the historical log. This is expected because users with more ratings can benefit more from the error variance analysis to avoid negative knowledge transfer.

5.3.3 STLCF with Multiple Source Domains

We apply STLCF(EV) on the extremely sparse target movie domain, with two sets of source domains: one is composed of Douban Music and Douban Book, the other is composed of Douban Movie, IMDB hyperlink graph and Wikipedia user editing records. The results are in Table 5.5 and Table 5.4 respectively. We demonstrate our STLCF method can utilize multiple source domains of various types by handling the inconsistency between the target and the source domains.

First, for the Douban experiments shown in Table 5.5, we observe that comparing to only using either Douban Book or Douban Music as source domain, there are significant improvements when both of them are used. The result is expected because each of the source domains has its own parts of effective information for the target domain. For example, a user who show much interests in the movie "The Lord of the Rings" may have consistent preferences in its novel. In this case, with the help of more auxiliary sources, better results are expected.

Second, we explore the generalization of the choices of source domains by introducing domains like Wikipedia user editing records and IMDB hyperlink graph, which are not directly related to the target domain but still contain some useful information in helping the target task (Netflix rating prediction). The results are shown in Table 5.4. Comparing the results of the experiment that uses no source domain (non-transfer) to those that use source domains D5 & D6, we observe that although the Wikipedia user editing records or IMDB hyperlink graph is not closely related to the target domain and can hardly be adopted as source domains by previous transfer learning techniques, our STLCF method can still transfer useful knowledge successfully.

In addition, comparing the results of the experiment that uses single source domain D3 to those that use source domains D3 & D5, D3 & D6, or D3 & D5 & D6, we find that the Wikipedia user editing records or IMDB hyperlink graph could provide some useful knowledge that is not covered by the related movie source domains. Despite of the noise and heterogeneous setting, our STLCF method can still utilize these source domains to help the target domain tasks. As we have discussed in Chapter 4, our STLCF performs selective transfer learning at both domain level and instance level. On one hand, the domain level selective transfer can block the noisy information globally. As we can see, D5 & D6 are noisy and therefore contain much data that are inconsistent with the observed data in the target domain, therefore the overall transfer of D5 & D6 is penalized. On the other hand, the instance level selective transfer learning can further eliminate the affections of those irrelevant source instances.

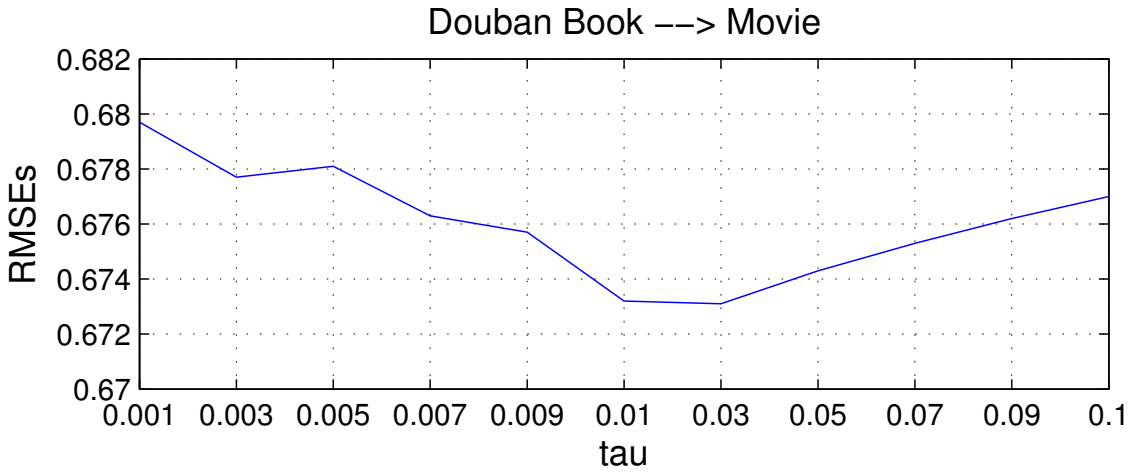

Figure 5.1: Change of the RMSEs with different τs. (Douban book to Movie)

Above all, our STLCF is highly adaptive to utilize source domains that are relatively inconsistent with the target domain, even when the target domain is rather sparse.

5.3.4 Parameters Analysis of STLCF

There are two parameters in our STLCF, i.e., the prediction error threshold τ and the empirical error variance weight γ. Since τ and γ are independent, we fix one and adjust another.

We fix the empirical error variance weight to be $\gamma = 0.5$ and adjust the parameter τ. Based on our results shown in Figure 5.1 and Figure 5.2 for two different transfer learning tasks, the model has good performance when τ is of order 10^{-2}. We also tuned the parameter γ, which balances the empirical error and its variance. We fix the prediction error threshold to be $\tau = 0.03$ in tuning γ. As shown in Figure 5.3 and Figure 5.4, when we vary the parameter γ from 0 to 1 in two settings, the best choices of γ are found to be around $0.4 - 0.5$.

5.3.5 Convergence and Overfitting Test

Figure 5.5 shows the RMSEs of STLCF(EV) as the number of weak learners changes on the Douban Book to Movie task. From the figure, we observe that STLCF(EV) converges well after

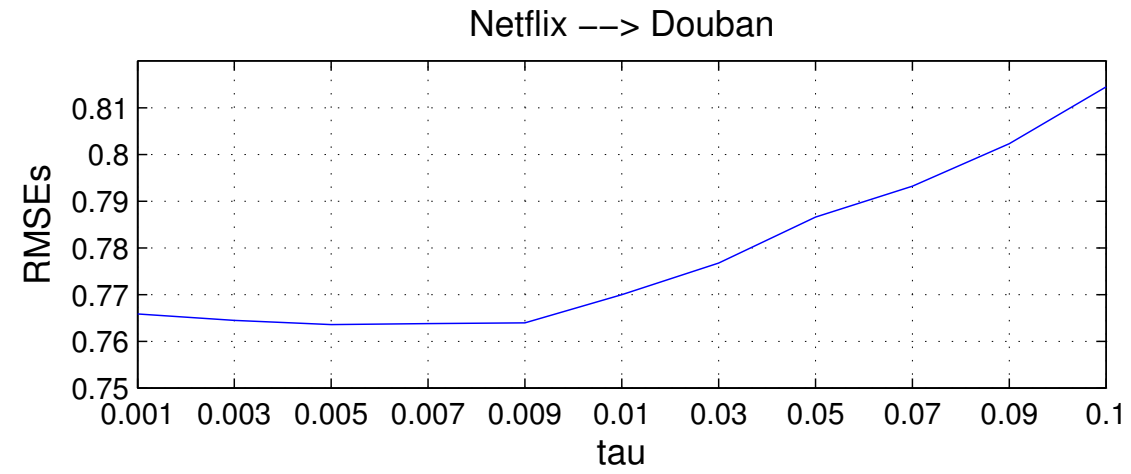

Figure 5.2: Change of the RMSEs with different τs. (Netflix to Douban Movie)

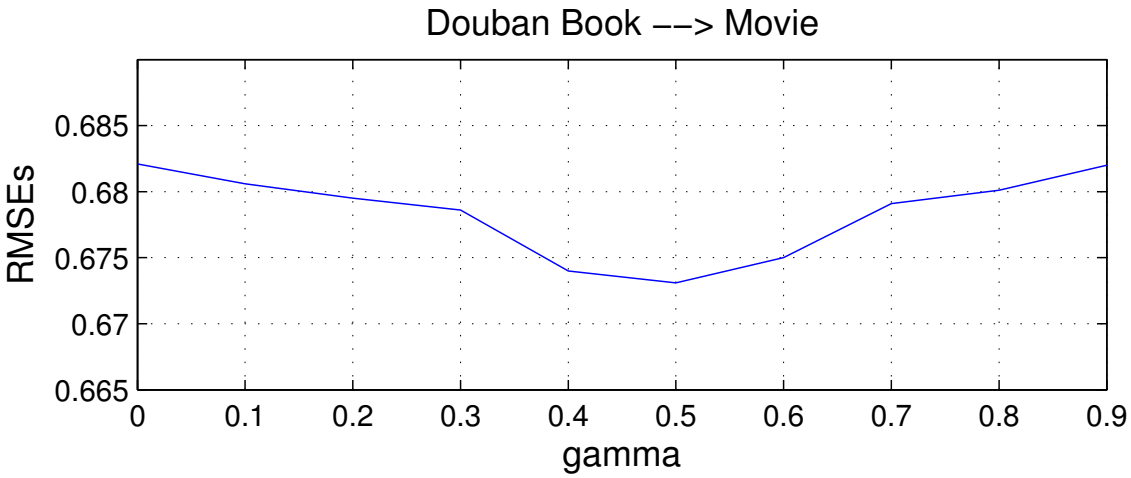

Figure 5.3: Change of the RMSEs with different γs. (Douban book to Movie)

40 iterations. In Figure 5.6, we can find that the corresponding α also converge to around 0.68 after 40 iterations as well. Empirically, we find STLCF converges in less than 50 iterations.

The number of latent topics of the base learner TGPLSA reflects the model's ability to fit training data. When we keep increasing the number of latent topics, the model tends to better fit the training data. But if the number of latent topics is too large, the model may suffer from overfitting.

We investigate the overfitting issue by plotting the training and testing RMSEs of the non-transfer learning model GPLSA, the non-selective transfer learning model TGPLSA and our selective transfer learning model STLCF(EV) over different numbers of latent topics in Figure 5.7. The data sparsity for the target domain is around 0.3%.

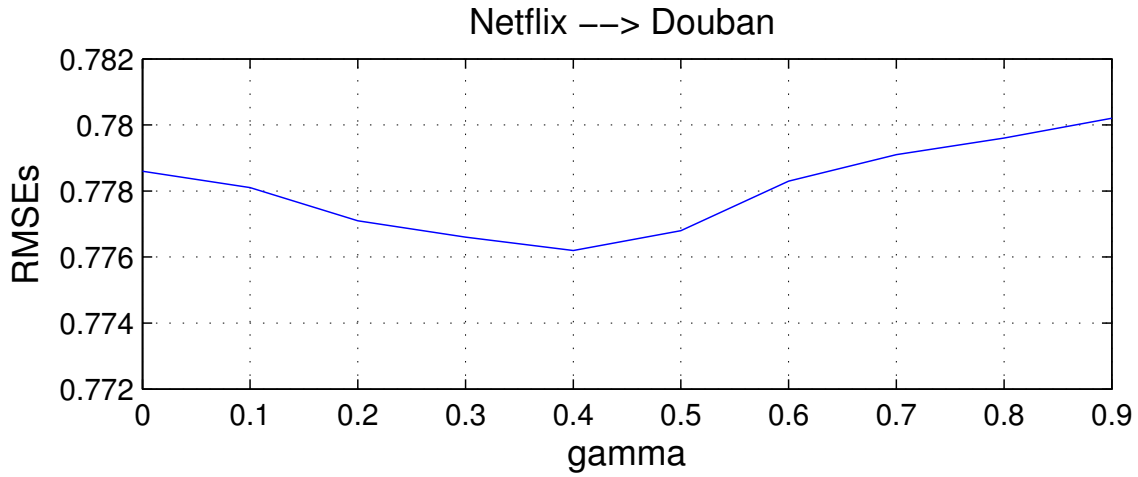

Figure 5.4: Change of the RMSEs with different γs. (Netflix to Douban Movie)

We can observe that comparing to our STLCF, the training RMSEs of GPLSA and TGPLSA decrease faster, while their testing RMSEs go down slower. When k is about 50, the testing RMSEs of GPLSA start to go up. And for TGPLSA, its testing RMSEs also go up slightly when k is larger than 75. But the testing RMSEs of our STLCF keep decreasing until $k = 125$ and even when k is larger than 125, the raise of our STLCF's testing RMSEs is not obvious. Clearly when the target domain is very sparse, our STLCF method is more robust against the overfitting, by inheriting the advantage from boosting techniques and the fine-grained selection on knowledge transfer.

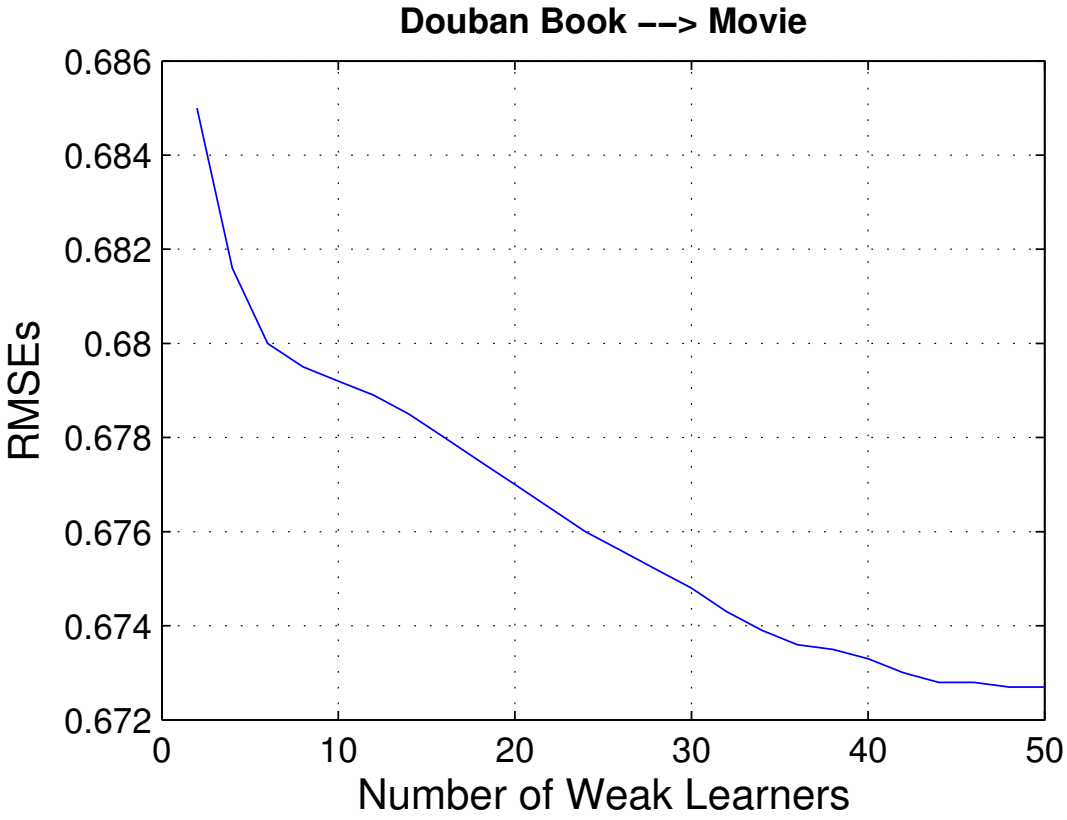

Figure 5.5: Change of the RMSEs when more and more weak learners join in the committee.

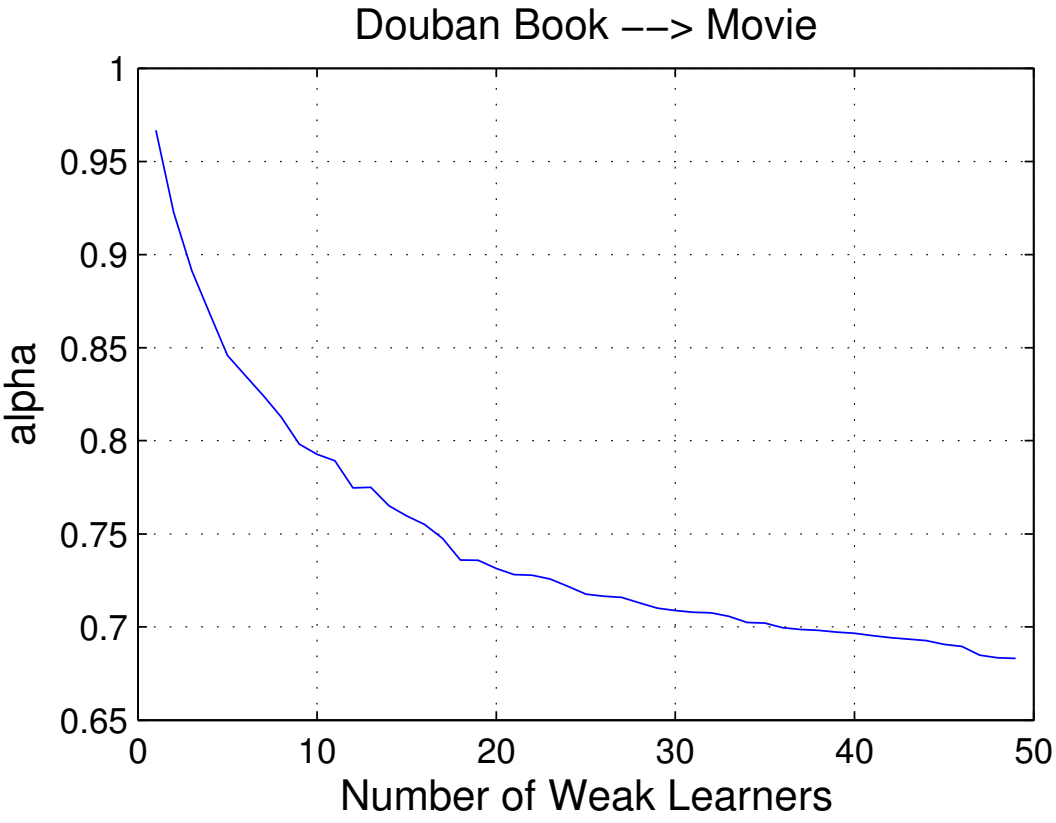

Figure 5.6: Change of αs when more and more weak learners join in the committee.

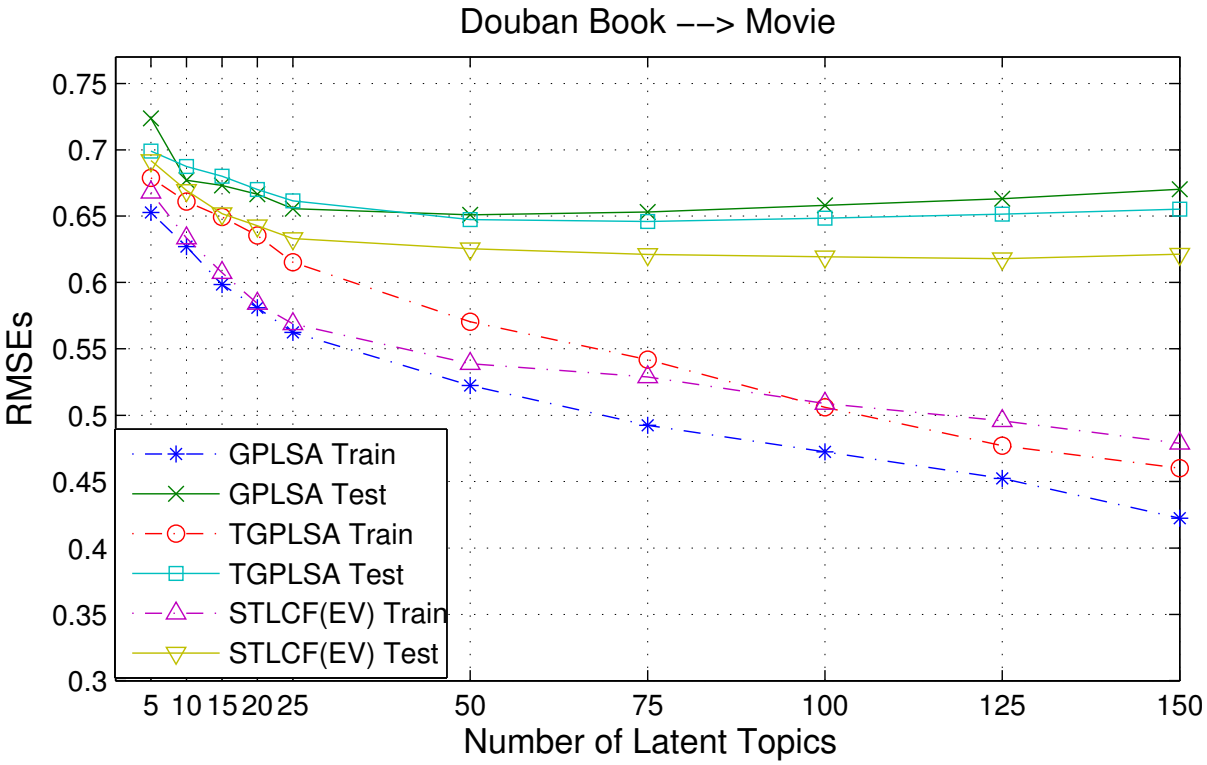

Figure 5.7: Change of the RMSEs with different numbers of latent topics.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis, we proposed to perform *selective* knowledge transfer for CF problems and came up with a systematical study on how the factors such as variance of empirical errors could leverage the selection. We found although empirical error is effective to model the consistency across domains, it would suffer from the sparseness problem in CF settings. By introducing a novel factor - variance of empirical errors to measure how trustful this consistency is, the proposed criterion can better identify the useful source domains and the helpful proportions of each source domain. We embedded this criterion into a boosting framework to transfer the most useful information from the source domains to the target domain. The experimental results on real-world data sets showed that our selective transfer learning solution performs better than several state-of-the-art methods at various sparsity levels. Furthermore, comparing to existing methods, our solution works well on long-tail users and is more robust to overfitting.

However, we notice that there are limitations in the work. First, in STLCF, the knowledge transfer is item-based. That is, each item / user is evaluated independently. Therefore, the implicit relationships between items / users are omitted. Second, The computational cost of STLCF is expensive, even though the parallel implementation makes it possible to run on large clusters. Third, we require the full correspondence on either user set or the item set as a bridge for the knowledge transfer. This requirement limits the applications of STLCF in the real world, because most of the real system will not be able to provide the full correspondence information. Fourth, we are aware that although the STLCF performs well on the long-tails (target domain tasks with very limited observations, for example the experiment in Section 5.3.2), it still can not handle the case where no record of target domains is exist.

We believe the Selective Transfer Learning has practical applications in the real world and would be a promising research topic. STLCF is our initial attempt on this topic. In the future to make Selective Transfer Learning be more robust, we propose the following approaches:

- **Model-based Selective Transfer Learning.** Instead of item-based knowledge transfer, we would like to explore the model-based transfer. That is, the domain information is first generalized as model and then be applied to later tasks. This will improve the universality of the source domain information and reduce the storage demand, as the information is generalized by models.

- **Relationship Regularized Selective Transfer Learning.** On the one hand, with the rapid growing of social networks in the internet, we have access to plenty of online user relationship. On the other hand, previous researches on taxonomy have made it possible to build relationship between items. Relationship regularized selection of helpful knowledge is naturally the next work.

- **Selection of Domain Correspondence.** Due to either record corruption or the absence of data in the industry, it is not always possible to obtain the full correspondence between the source domains and the target domain for knowledge transfer. To make Selective Transfer Learning be practical, we want to research on the selection of correspondence between domains when only part of it could be helpful. For example, in the settings where the user set is shared among the source and the target domains, we would like to select parts of the users during the transfer learning processes.

- **Boosting in Multi-Dimension.** The technique in this article can be viewed as boosting over either the item or user dimension. Can we extend it to multi-dimensional boosting? For example, would the interest of a user towards certain items evolute over time? With the evolution of user interests, is it possible to make a better prediction on the future ratings?

REFERENCES

[1] Wolfram Burgard and Dan Roth, editors. *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. AAAI Press, 2011.

[2] Bin Cao, Nathan N Liu, and Qiang Yang. Transfer learning for collective link prediction in multiple heterogenous domains. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 159–166, 2010.

[3] Cheng Chu, Sang Kyun Kim, Yi-An Lin, YuanYuan Yu, Gary Bradski, Andrew Y Ng, and Kunle Olukotun. Map-reduce for machine learning on multicore. *Advances in neural information processing systems*, 19:281, 2007.

[4] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pages 271–280. ACM, 2007.

[5] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[6] Chris Ding. Orthogonal nonnegative matrix tri-factorizations for clustering. In *In SIGKDD*, pages 126–135. Press, 2006.

[7] Chris Ding, Xiaofeng He, and Horst D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *in SIAM International Conference on Data Mining*, 2005.

[8] Eric Eaton and Marie desJardins. Selective transfer between learning tasks using task-based boosting. In Burgard and Roth [1].

[9] Eric Eaton, Marie desJardins, and Terran Lane. Modeling transfer relationships between learning tasks for improved inductive transfer. In *The European Conference on Machine Learning*, pages 317–332, 2008.

[10] Hoda Eldardiry and Jennifer Neville. Across-model collective ensemble classification. In Burgard and Roth [1].

[11] Zoubin Ghahramani, editor. *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Oregon, USA, June 20-24, 2007*, volume 227 of *ACM International Conference Proceeding Series*. ACM, 2007.

[12] Thomas Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *SIGIR*, pages 259–266, 2003.

[13] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.

[14] Yifan Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, pages 263–272, Dec 2008.

[15] Rong Jin, Luo Si, Chengxiang Zhai, and Jamie Callan. Collaborative filtering with decoupled models for preferences and ratings. In *Proceedings of CIKM 2003*, 2003.

[16] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[17] Bin Li, Qiang Yang, and Xiangyang Xue. Can movies and books collaborate?: cross-domain collaborative filtering for sparsity reduction. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 2052–2057, 2009.

[18] Bin Li, Qiang Yang, and Xiangyang Xue. Transfer learning for collaborative filtering

via a rating-matrix generative model. In *International Conference on Machine Learning*, volume 26, 2009.

[19] Tao Li, Yi Zhang, and Vikas Sindhwani. A non-negative matrix tri-factorization approach to sentiment classification with lexical prior knowledge. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 244–252, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[20] Marek Lipczak and Evangelos Milios. Efficient tag recommendation for real-life data. *ACM Transactions on Intelligent Systems and Technology (ACM TIST)*, 3(1):2:1–2:21, October 2011.

[21] Bhaskar Mehta and Thomas Hofmann. Cross system personalization and collaborative filtering by learning manifold alignments. In *KI 2006: Advances in Artificial Intelligence*, pages 244–259. 2007.

[22] Rong Pan, Yunhong Zhou, Bin Cao, N.N. Liu, R. Lukose, M. Scholz, and Qiang Yang. One-class collaborative filtering. In *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, pages 502–511, Dec 2008.

[23] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October 2010.

[24] Weike Pan, Nathan N. Liu, Evan W. Xiang, and Qiang Yang. Transfer learning to predict missing ratings via heterogeneous user feedbacks. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2318–2323. AAAI Press, 2011.

[25] Weike Pan, Nathan Nan Liu, Evan Wei Xiang, and Qiang Yang. Transfer learning to predict missing ratings via heterogeneous user feedbacks. In *IJCAI*, pages 2318–2323,

2011.

[26] Weike Pan, Evan Xiang, Nathan Liu, and Qiang Yang. Transfer learning in collaborative filtering for sparsity reduction. 2010.

[27] Weike Pan, Evan W. Xiang, and Qiang Yang. Transfer learning in collaborative filtering with uncertain ratings. In *AAAI*, 2012.

[28] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. *Proceedings of KDD Cup and Workshop*, 2007.

[29] David M. Pennock, Eric Horvitz, Steve Lawrence, and C. Lee Giles. Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach. In *Proc. of UAI*, pages 473–480, 2000.

[30] Steffen Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (ACM TIST)*, 3:19:1–19:22, May 2012.

[31] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *NIPS*, 2007.

[32] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey E. Hinton. Restricted boltzmann machines for collaborative filtering. In Ghahramani [11], pages 791–798.

[33] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Application of dimensionality reduction in recommender system – a case study. In *IN ACM WEBKDD WORKSHOP*, 2000.

[34] Shai Shalev-Shwartz and Nathan Srebro. Svm optimization: Inverse dependence on training set size. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 928–935, New York, NY, USA, 2008. ACM.

[35] Ajit Paul Singh and Geoffrey J. Gordon. Relational learning via collective matrix factorization. In *KDD*, pages 650–658, 2008.

[36] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009:4, 2009.

[37] Yu Zhang, Bin Cao, and Dit-Yan Yeung. Multi-domain collaborative filtering. In *UAI*, pages 725–732, 2010.

[38] Yu Zheng and Xing Xie. Learning travel recommendations from user-generated gps traces. *ACM Transactions on Intelligent Systems and Technology (ACM TIST)*, 2(1):2:1–2:29, January 2011.

[39] Fuzhen Zhuang, Ping Luo, Hui Xiong, Qing He, Yuhong Xiong, and Zhongzhi Shi. Exploiting associations between word clusters and document classes for cross-domain text categorization. *Stat. Anal. Data Min.*, 4(1):100–114, February 2011.