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

December 2014, 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 December 2014

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. QIANG YANG, HEAD OF DEPARTMENT

Department of Computer Science and Engineering

18 December 2014

ACKNOWLEDGMENTS

The past two years and a half 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 guidance of Prof. Yang, I would never have the chance to exploit so many possibilities in my life. Also I would never be able to explore my interest in the area of recommender system.

I would like to express my great thankfulness to Lili Zhao and Liya Ji for their companionship and valuable suggestions on my study, also for their kindness and tolerance as a childish boy's roommate.

I would like to thank Ben Tan, Bo Liu, Bin Wu, Kaixiang Mo, Zhongqi Lv, Lianghao Li for their ideas, suggestion and company. And thank Dr. Wei Xiang and Dr. Qian Xu for their guidance in future career. In addition, I would like to thank Dr. Yong Li, Xiaoping Lai, Xiaopeng Zhang, Lei Xiao for their encouragement and criticism.

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 mom for her support and belief.

Wish all my friends good luck, and have fun!

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 [28, 25], implicit feedback data is available, knowledge is transferred via latent-feature sharing. In [39, 9] they try to exploit correlations among multiple domains. However, most of the methods are developed 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 achieve better performance, we must transfer more knowledge from source domain while be very careful about the noise.

Some works have been done on solving one-class recommendation problem [12, 23, 20]. They all try to model the frequency 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 an experienced 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 a 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 click-through-rate/conversion-rate).

Further, to make the method online, we develop 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.

1.2 Contributions

Our main contributions are summarized as follows:

- First, we find that in implicit datasets, more data must be shared to achieve 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 linear cluster-level transformation. Thus we can share knowledge more accurately without losing too much information.

- Third, we propose a modified version of TRIMF(CBMF) which can be used for large scale recommendation. 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 Collaborative Filtering, Matrix Factorization and Transfer Learning in Chapter 2. Then, we discuss the technique details of the proposed matrix tri-factorization method(TRIMF) and experiments on real-world datasets in Chapter 3. We present the details of our proposed CBMF framework and experiments in online website in Chapter 4 . Finally, we share our thoughts of possible future work and conclude the thesis in Chapter 5.

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 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
$Rating \to Rating$	RMGM [17]	CMF [37]	
$Implicit \to Rating$		CST [27], TCF [26]	TIF [28]
$Implicit \to Implicit$	TRIMF		

2.1 Collaborative Filtering

Collaborative filtering ([15], [31]) as an intelligent component in recommender systems ([40], [21]) 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[38].

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 [15, 28, 29, 31], probabilistic mixture models [11, 14], Bayesian networks [30] and restricted Boltzman machines [34].

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 [4], 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 [6] paradigm. In [36], 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 [5].

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: Singular value decomposition(SVD) and 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 [35]. $\sum$ 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 [15].

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 [8], the authors proved that NMF is equivalent to k-means clustering. In [7] 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, [41] 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 [24] 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 [22] 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. [37] 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.* [26], [27] 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.* [3] 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 dataset A to implicit 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, [5] 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 behavior 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 it, we can't achieve good, even reasonable performance.

In Yixun(Tencent's online shopping site), there are two main actions - click and purchase, both can form a matrix which consists only binary data(1-action happened, 0-unknown). Let's denote X_d to be the matrix of deal, X_c to be the matrix of click. We know that deal matrix X_d is very sparse, although click matrix X_c is also sparse, but is much denser than X_d. In order to leverage more data, we developed a transfer learning algorithm(TRIMF) that leverage click 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 through cluster-level transform and overlapping matrix. Experiments show that our algorithm performs better then other baseline(transfer and non-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 overlapped.

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

3.2 TRIMF

3.2.1 Weighting scheme of TRIMF

Former one-class CF methods [12], [23] use weighted low-rank approximation to tackle the problem that all observed ratings are 1. Given a rating matrix $R = (R_{ij})_{m*n} \in \{0,1\}^{m*n}$ with m users and n items and a corresponding non-negative weight matrix $W = (W_{ij})_{m*n} \in R^{m*n^+}$, weighted low-rank approximation aims at finding a low rank matrix $X = (X_{ij})_{m*n}$minimizing the objective of a weighted Frobenius loss function as follows : $L(X) = \| \sum W_{ij}(R_{ij} - X_{ij})\|_2$.

In [12], the authors consider actions that happens more than once(e.g click an item multiple times). Negative entries are ignored, for each positive entry, its weight is proportional to its frequency, since more frequent can mean that we are more confident about the entry. For example, user i had view item j for n times, then $W_{ij} = 1 + log(n)$. In [23], positive entries are all with the same weight 1, while negative entries are considered in different ways. According to their experiments, user-oriented weighting scheme can achieve best performance. That is, for negative entries $W_{ij} \propto \sum_j R_{ij}$, it's idea is that if a user has more positive examples, it is more likely that she does not like the other items, that is, the missing data for this user is negative with higher probability.

In our method we adopt this weighting scheme to give missing values proper weights, i.e for positive entries we use the weighting scheme in [12] and negative entries we use user-oriented weighting.

$$W_{ij} = \begin{cases} 1 + log(n) & X_{ij} = 1 \\ log(\sum_j R_{ij}) & X_{ij} = 0 \end{cases}$$

3.2.2 Transfer learning in TRIMF

Usually users' deal data is very sparse, e.g users will buy n_d items in one day while click n_c items. Then often we have $n_d \ll n_c$. So only use deal data is not sufficient. Traditional transfer learning methods use matrix factorization and share a certain part of low-rank matrix to achieve knowledge transfer(e.g user-latent factor, rating pattern). But none of them apply the selective-sharing scheme as our do.

In TRIMF, rating matrix are factorized into three parts : $X = USV^T$. The first part U stands for user clustering results or latent factor, V stands for item clustering results or latent factor while S stands for clustering relationships between user clusters and item clusters. We want to learn a better cluster-level rating pattern S from users' deal data and with the help of users' click data, not only using users' deal data. So we factorize two matrix X_c, X_d together, in order to transfer knowledge, we must make sure that their latent spaces are the same. Especially for one user who has click and deal actions, it would be nice and reasonable that his latent vectors factorized from X_c, X_d are the same.

So for overlap users and items, we want their clustering vector U, V to be the same. What's more, we want even more knowledge transfer from the matrix S which stands for cluster relationship or rating patterns. But what a user like to click is not always the item he wants to buy.In Yixun, there are only 2 common items in top-10 click items and top-10 purchase items (Table 3.2.2). So these rating patterns should be somehow related but not the same. We can't simply make the pattern matrix S the same in prediction.

Top click items	Top purchase items
Iphone 5s	Tissue
Xiaomi 3	Laundry soap powder
Thinkpad	Xiaomi 3
CPU	Snacks
Hard disk	Battery
Router	Iphone 5s
Earphone	Mouse

Table 3.1: Top 10 click items and purchase items in Yixun.

After careful observation we found that there are some items which belongs to the same category with higher conversion rate(user tends to buy after clicking), but some categories not. And there are some users who like window-shopping while others will buy it right after clicking. These are all cluster-level features, we design a mapping function to let the learnt S suit data better.

We design two mapping vectors: U, V, if we've learnt a rating pattern S_c from click matrix, then for the deal matrix pattern we have $S_d^{ij} = U_i * S_c^{ij} * V_j$. The transformation is based on S_c to enable knowledge transfer, while after multiplied by U and V we can capture the difference between them, in cluster-level.

3.2.3 Object function

We use a weighted non-negative matrix tri-factorization method to deal with the problem as illustrated below.

Objective Function:

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

- W_c, W_d is the weight for X_C, X_d, every observed entry has weight $1 + log(frequency)$. while others have weight $W_{ij} = \sum_j I(R_{ij})$.

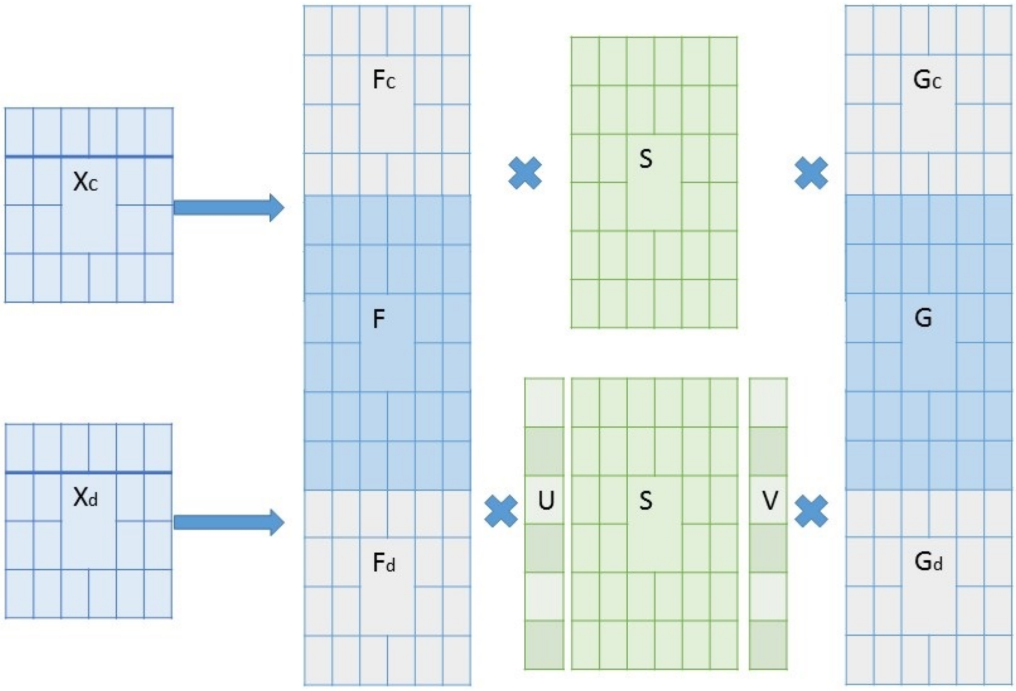

Figure 3.1: Graphical model of TRIMF.

- 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' cluster-level transform from click to deal. While V_{jj} scales every S_{*j} to $S_{*j}V_{jj}$, it models the items' cluster-level transform 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 TRIMF & Algorithm

Follow the update rules in [41], we use an alternately iterative algorithm to solve the objective function.

Firstly, we declare some denotement:

- Ic, Icc, Icd, Id : $(Ic, Icc) * (F; F_c) = I * (F; F_c)$ and $(Icd, Id) * (F; F_d) = I * (F; F_d)$
- sg : $S * G' * G * S'$
- F_1, F_2 : $[F; F_c]$, $[F; F_d]$

In each round of iteration these matrices are updated as :

$$F \leftarrow F. * \sqrt{\frac{Icc' * (W_c. * X_c) * G * S' + Icd' * (W_d. * X_d) * G * S'}{(Icc' * Icc * F + Icc' * Ic * F_c + Icd' * (Icd * F + Id * F_d)) * sg)}}$$

$$F_c \leftarrow F_c. * \sqrt{\frac{Ic' * (W_c. * X_c) * G * S'}{Ic' * (Icc * F + Ic * F_c) * sg}}$$

$$F_d \leftarrow F_d. * \sqrt{\frac{Id' * (W_d. * X_d) * G * S'}{Id' * (Icd * F + Id * F_d) * sg}}$$

$$G \leftarrow G. * \sqrt{\frac{W_c. * X_c * F_1 * S + (W_d. * X_d)' * F_2 * S}{(G * (S' * F_1' * F_1 * S + S' * F_2' * F_2 * S)}}$$

$$U \leftarrow U. * \sqrt{\frac{F_2' * (W_d. * X_d) * G * V' * S'}{F_2' * F_2 * U * S * V * G' * G * V' * S'}}$$

$$V \leftarrow V. * \sqrt{\frac{S' * F_2' * (W_d. * X_d) * G}{S' * F_2' * F_2 * S * V * G' * G}}$$

The user-item matrix is typically very sparse with $z \ll nm$ 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 1: Algorithm for TRIMF.

Input: $\mathbf{X}_c$, $\mathbf{X}_d$
$\mathbf{X}_c \in \mathbb{R}^{m_c \times n_c}$: the purchase data
$\mathbf{X}_d \in \mathbb{R}^{m_d \times n_d}$: the click data
Initialize: Initialize $W_c, W_d : (1 + log(freq))$ for observed, $\sum_j I(R_{ij})$ for unseen, $F, G, S, U, V : random$, Set overlap numbers for users and items
for i = 1 to T **do**
  update F
  update F_c, F_d
  update G
  update S
  update U, V
end for
Output: F, G, S, U, V

3.4 Experiment

3.4.1 Datasets

We select real data from an online shopping site: yixun.com [1]. We collect data for 6 months, the entire dataset consists 5,324,231 users and 643,123 items. The sparsity in click matrix is 0.06%, in purchase matrix is 0.0003%.

To check the effectiveness of TRIMF in short time and long time, we construct two smaller datasets.

- **Yixun short term data**: we select data from two weeks, 20130801-20130814. we sample a fraction of user by random, and remove those whose action frequency is too low(e.g only one click during this period). In the click matrix we have 16240 users and 1932 items. In the purchase matrix we have 2520 users and 1791 items. There are 2029 overlapped users and 1642 overlapped items. We train our model using data from first week and data from second week is used for testing.
- **Yixun long term data**: we select 1012 active users through 6 months. In their long term actions, there are 6021 items which have been clicked and 1973 items boughted. We select the five latest purchasing items per user as test data, others as training data. There

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

non-transfer methods	transfer methods
Most Popular, SVD, NMF, PMF, BPRMF, WRMF	CMF, TCF, TRIMF

Table 3.2: Baseline methods.

are 1503 overlapped items.

3.4.2 Metrics

We use $prec@5$ and $prec@10$ as our evaluation metrics. $prec@n$ is the precision of top-n results. Our main goal is to optimize for conversion rate(future purchase matrix), so the test is mainly done in the purchase matrix. However, since TRIMF can also optimize for source domain(click matrix), some test in click matrix is also conducted.

3.4.3 Baseline methods

We divide baseline methods into non-transfer methods and transfer methods. All baseline methods are shown in Table 3.2.

non-transfer methods

For all non-transfer methods, we use 3 combinations of matrix as our training matrix:deal, click, deal+click, and report their **best** performance. We choose parameters by cross validation.

- Most Popular: Most popular selects top-n items in global, and provide same recommendation results for each user.
- SVD [29]: Singular Value Decomposition(SVD) is a typical method used in recommender system, here PureSVD from Matlab is used.
  - rank = {5,10,20,30,40,50}
- NMF [15]: Non-negative Matrix Factorization(NMF) is also a typical method used in recommender system, here NMF from Matlab is used.

- rank = $\{10,20,40,60,100\}$

- PMF [33]:Probabilistic Matrix Factorization(PMF) 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.
  - rank = $\{10,20,30,40,50\}$

- BPRMF [32]: BPR is a generic optimization criterion for personalized ranking that is the maximum posterior estimator derived from a Bayesian analysis of the problem. Unlike traditional methods whose object function is point-wise, BPR is a pair-wise object function. BPRMF implements BPR using matrix factorization.
  - We initialized BPR with most popular results.
  - We set $iteration = \#n * 100$, ($\#n$ in the number of observations)

- WRMF [23]: One-class collaborative filtering(WRMF) is a weighted low rank approximation method optimized for implicit dataset.
  - rank = $\{5,10,15,20,25\}$

transfer methods

- CMF [37]:Collective Matrix Factorization 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. For each training and testing pairs, we make two matrix the same dimension(in order to share a latent factor) by padding rows & columns.
  - Shared latent space dimension = $\{5,10,15,20,25\}$

- TCF [26]: TCF is a transfer learning method to predict missing ratings via heterogeneous feedbacks. It's originally designed for rating prediction, so we set the deal matrix with

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 3.3: Performance of TRIMF and other baseline methods on short-term users who have deal data.

random sampled zeros as the rating matrix, click matrix as the implicit feed back matrix. Zero rows and colomns are also padded to make the two matrix in same dimension.

- TRIMF: our method.
  - We set latent factor = 30, iteration = 200.

3.4.4 Results

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.

Results are showed in Table 3.3 and Table 3.4.

Yixun long term data

Since the user are manually selected, we only test X_d. The result is showed in Table 3.5.

3.5 Performance comparison & analysis

First, we observed that TRIMF out-perform all other baseline non-transfer methods in three tests. In short-term deal test, we can see traditional CF method which aims at rating prediction

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 3.4: Performance of TRIMF and other baseline methods on short-term users who have deal 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 3.5: Performance of TRIMF and other baseline methods on long-term users.

(e.g SVD, NMF) can't achieve compatible performance than others. Because these methods is designed for matrix with multiple values, not for our binary matrix. while CF method designed for binary matrix(BPRMF, WRMF) can achieve significantly greater result. In long-term test the difference is not so significant, because the data here is less sparse than short-term data, every method has enough data to train a good model.

Second, TRIMF also out-perform other transfer methods. Since CMF, TCF are also designed for rating prediction problems. The information in our training set is limited, so both methods can't transfer enough knowledge from their framework. TRIMF is designed for one-class transfer learning, TRIMF combines one-class and transfer methods, so it inherits advantages from both side.

The effects of cluster-level tranformation

In our assumption, U, V are two mapping matrix that describe the difference in user-cluster and item-category. To see whether U, V really reflects the phenomenon, we manually check entries in U, V with high and low values.

We found that high value in V reflects item clusters that people tends to buy after clicking, e.g toothbrush, snacks. While low value of V more reflects that items are popular but people may not buy it immediately, e.g cell phones, laptops. High value in U reflects user-cluster who tends to buy after clicking, while users belong to low value user-clusters are all window-shopping fans.

In Table 3.4, we can see if we want to predict future purchasing items on users who have click data, we can map UV back. Thus the learned cluster-pattern S is transformed from click pattern to purchase pattern.

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

Table 3.6: The effect of sharing.

The effects of latent vector sharing

In our method, for the same user the latent vectors are unique. Our intuition is that by making some vector alignment, we can leverage this information to factorize or co-cluster two matrix in the same latent dimension. Making them the same dimension is the foundation of knowledge transfer which happens during the alignment. To see that sharing F, G really works, we select another 6000 users and 1500 items, make two new matrix X'_c, X'_d to perform another test. We tried three sharing scheme:

- shareFG : TRIMF
- not share : We update F_c, G_c, F_d, G_d separately, pretending there is no overlap users or items.
- random share : We randomly choose 3000 users and 800 items, marked them as overlap,

The $prec@n$ here are not comparable with the first experiment in Table 3.3. Result(Table 3.6) shows that we must share latent factor carefully, random share may do harm to performance. But sharing latent factor for overlapped users/items can achieve a significantly greater result.

CHAPTER 4

CLUSTERING-BASED MATRIX FACTORIZATION FOR LEOLINE SHOPPING PREDICTION

4.1 Limitation of TRIMF

In Chapter 3, we introduced TRIMF. It's a matrix tri-factorization method for cross-domain recommendation. However, it has some limitations which restricts its scalability and extensibility.

First, when data are coming from multiple sources(e.g click, pageview, add cart), TRIMF treats every source equally and put each of them into a matrix which is very sparse. When solving the object function, more matrix will increase the time complexity and space complexity. If we try to update S, every matrix is included so it will be very time-consuming.

What's more, in reality we can't ignore users with fewer actions. Thus the matrix will be much more sparse than the ones in our experiment, so we can't guarantee to acheive equal performance.

To solve these problems, we developed a framework based on clustering and scoring scheme (CBMF, Figure 4.1). CBMF firstly cluster users according to their behavior and demographic features, then automatically convert different types of actions into one matrix call action matrix, finally a matrix factorization method is applied in the action matrix. For users with enough actions, personalized recommendation is provided. Otherwise we provide a recommendation based on his/her cluster.

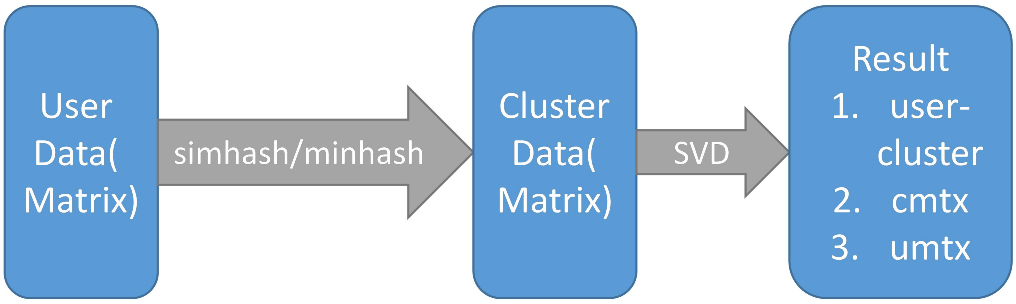

Figure 4.1: Framework of CBMF.

4.2 Clustering method in CBMF

Usually users' actions are unique and sparse, it'll be time-consuming if we want to cluster users using raw data. In Tencent, we have 800,000,000 users in total, and their feature vector size can be as large as 1,000,000. So if we want to speed up the phase, we must first convert large sparse user vector into a low dimension dense vector.

4.2.1 Simhash

Simhash is a kind of locality sensitive hashing(LSH). LSH is a hashing method that if we got two points A, B which are close in their original space, after hashing we got A', B', then A', B' is still close in the new space. Thus we keep the relationship of distance among two spaces.

The input of Simhash per user is $(feature_1, weight_1), ..(feature_n, weight_n)$. The procedure of Simhash is in Algorithm 2.

Assume that Simhash convert a vector x into a 32-dimension binary vector x'. Actually i_{th} bit of x' is the sign of inner product of x and $H_i = [h_i^1, h_i^2, ...h_i^n]$, H_i can be regarded as a hyperplane in original space. If two vector x, y are in the same direction of H_i, then x', y' is equal on i_{th} bit. Thus we can use hamming distance in the low dimension to represent their similarity in original space.

Algorithm 2: Simhash Algorithm for one instance.

Input: $\mathbf{X}_U, h$
$\mathbf{X}_U$: $(feature_1, weight_1), ..(feature_n, weight_n)$
h: a smooth hash function with k bits after hashing
Initialize: r(result vector) : $[0, 0, 0..0] \in \{0, 1\}^k$
for i = 1 to n **do**
  calculate $h(feature_i)$
  $r = r + weight_i * h(feature_i)$
end for
for i = 1 to k **do**
  if $r_i > 0$ **then**
    $r_i = 1$
  else
    $r_i = 0$
  end if
end for
Output: r

4.2.2 Minhash

The similarity between two users u_i, u_j is defined as the overlap between their item sets. $S(u_i, u_j) = \frac{C_{u_i} \cup C_{u_j}}{C_{u_i} \cap C_{u_j}}$, also known as the Jaccard coefficient. But doing this in real-time is clearly not scalable. However, we can achieve a provably sublinear time near-neighbor search technique by applying minhash.

MinHash is a probabilistic clustering method that assigns a pair of users to the same cluster with probability proportional to the overlap between the set of items that these users have voted for (clicked-on). In CBMF, minhashing is applied after simhashing, so every user has 32 items(bits).

The basic idea in the Min-Hashing scheme is to randomly permute the set of items (S) and for each user u_i compute its hash value $h(u_i)$ as the index of the first item under the permutation that belongs to the users item set C_{u_i}. And it's easy to show that the probability that two users will have the same hash function is exactly equal to their similarity or Jaccard coefficient.

Similar to [13], we can always concatenate p hash-keys for users, where $p \geq 1$, so the probability that any two users u_i, u_j will agree on the concatenated hash-key is equal to $S(u_i, u_j)^p$. p can be a parameter to control the number of clusters. If p is large, then the clusters will be more refined thus the number of cluster will increase.

4.2.3 Simhash & Minhash using MapReduce

MapReduce is a very simple model of computation over large clusters of machines that can handle processing of large amounts of data in relatively short periods of time and scales well with the number of machines. Our method Simhash and Minhash can be easily implemented using hadoop.

Map phase

In the map phase, we read the input records independently, in parallel, on different machines and map each input to a set of zero or more key- value pairs. In our case, hadoop streaming is applied and each input instance is a user's vector(in sparse representation).

We first iterate the user's vector u_i, using Simhash to convert the vector to a 32-bit binary vector, the hashing function used in Simhash is FNV-32. Then Minhash is applied for p times per user. We concatenate the p Minhash values $Mnhs_i$ to obtain the cluster id of the user. Finally, the output is ($Mnhs_i$, u_i), key is $Mnhs_i$, value is u_i. For users with enough actions, we output another pairs ($user - id$, u_i).

Reduce phase

In the reduce phase, our input has two form: ($Mnhs_i$, u_i) represents cluster-id and uservector. ($user - id$, u_i) represents an experienced user and its vector.

- For the cluster case: we obtain for each cluster-id the list of user-ids that belong to this cluster and prune away clusters with members less than 10. For each cluster-id, a joint vector is needed to represents all users in it. Thus we simply add scores from users to the joint vector, then we do a normalization to make the range of the vector between 0 and 1. The output has two parts:
  - user and the cluster he belongs to (userid, clusterid).

– cluster-id and its vector (cluster-id, cluster-vector).

- For the user case: we simply output the normalized vector.

After the reduce phase, we have three tables(matrix). 1, user and his cluster-id. 2, cluster-id and its vector. 3, user and his vector.

4.3 Feature construction in CBMF

After clustering, we have many clusters and their corresponding actions, including click, purchase, pageview on different(overlap) items.

The naive way to handle those actions is to create a matrix X_i for each action i. In matrix X_i, a row represents a cluster while a column represents an item, X_{mn} represents the frequency of action that users in cluster m had on item n. But simply creating such matrix may lead to data sparsity problems. Especially in the matrix standing for purchasing actions, even though we had already clustered similar users together. The data is still very sparse(0.01%) which may constrain our model from providing reasonable recommendation.

In CBMF, a scoring scheme is applied for each kind of action to put every action into a single matrix with proper score.For a specific item, a user may have four kinds of actions(click, purchase, pageview and uninterested). The idea behind the construction is that for a specific goal(e.g predict future purchase), what score should be given to an action depends on how much impact the action can have.

For example, if we want to improve conversion rate, let U_n denote the users who bought item n, U denote the entire user set. Then the average conversion rate for a given item n can be approximated by $Cvr(n_{all}) \approx \frac{|U_n|}{|U|}$. For a given action(e.g click), let U_n^{click} denote the users who clicked item n. Then the conversion rate for users who had clicked this items can be approximated by $Cvr(n_{click}) \approx \frac{|U_n \cap U_n^{click}|}{|U_n^{click}|}$we compare the conversion rate of users who had this action with average, their log ratio $log(\frac{Cvr(n_{click})}{Cvr(n_{all})})$ is our initial score.

For each action with an item we calculate a score, in CBMF we used weighted scores and add them together. That is, for cluster m and item n, if we have four scores:s_1, s_2, s_3, s_4 and their corresponding weights:w_1, w_2, w_3, w_4. We have $X_{mn} = \frac{\sum_{i=1}^{4} w_i * s_i}{\sum_{i=1}^{4} w_i}$. w_i is the percentage of users who have this action compared to all users.

4.4 Matrix factorization in CBMF

Once the matrix are generated, we use Singular Value Decomposition(SVD) from mahout [1]. The number of eigenvalues is set to 20 according to online test.

After SVD, we can provide recommendations to every user, for experienced users direct results are provides, otherwise we find his cluster-id and provide results for this cluster as our recommendation. We output our results to a online key-value dataset called TDE. Once a user comes, we search his id is TDE, the return value is our recommendation.

4.5 Experiments and results analysis

Our algorithm has gone online for about one months from 2014-04-01 to 2014-05-01 with 10% 20% users, in the chatting scenerio. That is, when you are chatting using QQ, an ad full of items will pop out in front of you. Most of the users will close it immediately, so 90% of the users are new. And we need to provide results for every user, about 80,000,000.

After some days of parameter selection, CBMF's performance becomes stable since 2014-04-10.

There are three evaluation metrics:

1 click-through-rate(CTR).

2 order amount per impression(OAPI)

[1] https://mahout.apache.org/users/dim-reduction/dimensional-reduction.html

3 pay amount per impression(PAPI).

An impression is a measure of the number of times an ad is seen. The difference between OAPI and PAPI is that a user may place an order but didn't pay for it(he can cancel at any time).

4.5.1 Baselines

There are some others algorithms competing with CBMF, we split users into different disjoint sets. Each algorithm impact a set of users, and an ID is given to every algorithms, the ID of CBMF is 4312. The detail of other algorithms is shown below:

- 4312: CBMF, updates per hour.
- 4313: Locally popular algorithm. Each user belongs to a cluster according to their age and gender, then the most popular items in this cluster are recommended. Real time algorithm.
- 4314: Logistic regression algorithm, the feature consists users' demographic data, items' category data and their conjunctions. The algorithm updates per hour.
- 4315: Globally popular algorithm. For every user, items with most-clicked are provided. Real time algorithm.
- cpsf: Redirecting based algorithm, if a user had some positive(click, pageview, purchase) actions before, recommend popular items in the same category. If the user is new, recommend the most-clicked items in different category(avoid recommending items all from one category). Real time algorithm.

4.5.2 Click through rate

In Figure 4.2 and Figure 4.3, CTR of different online algorithms are shown. The id of CBMF is 4312, the purple line. It can be seen that CBMF didn't rank first in CTR, but rather among

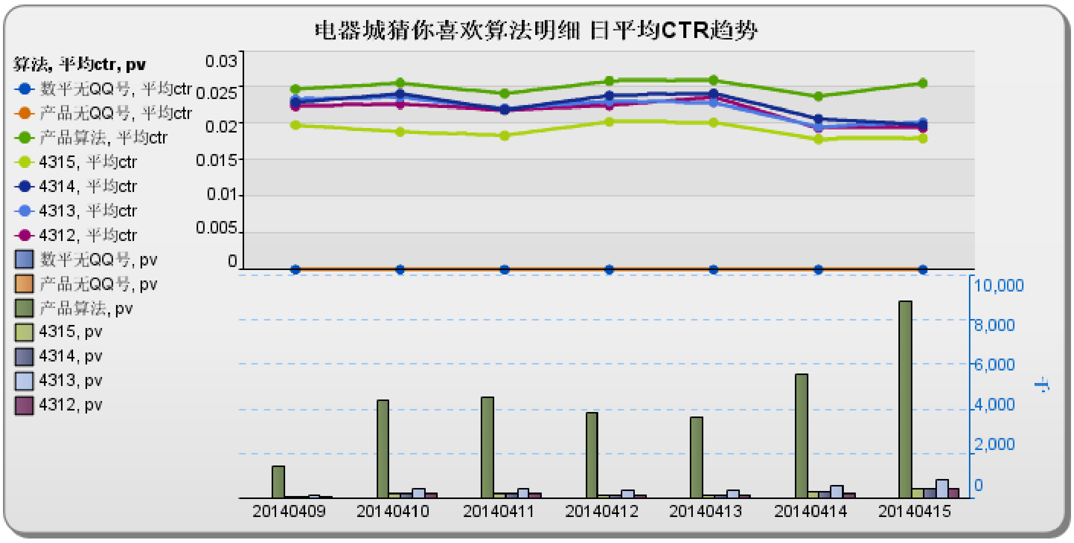

Figure 4.2: CTR of online algorithms from 0409 to 0415.

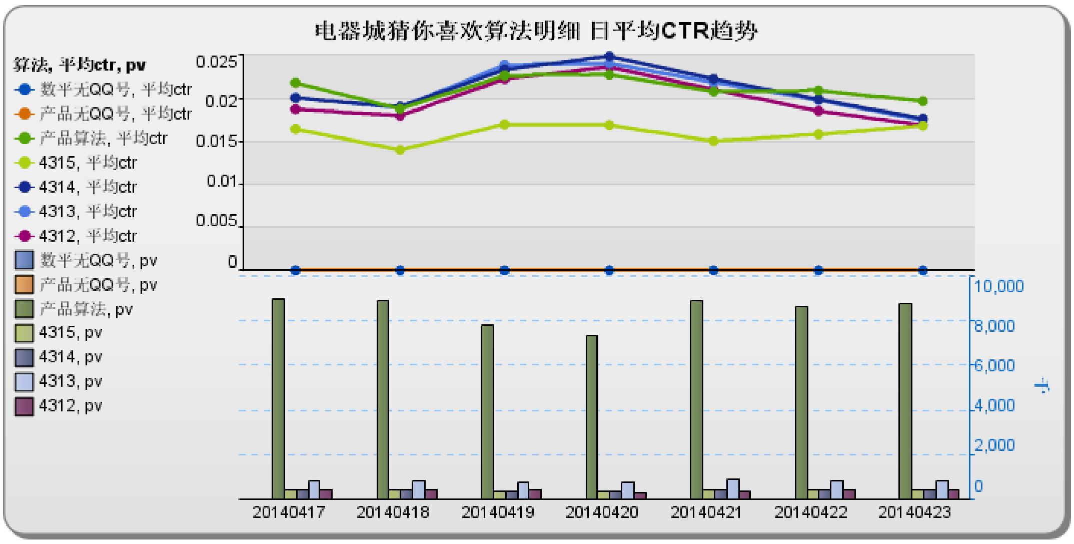

Figure 4.3: CTR of online algorithms from 0417 to 0424.

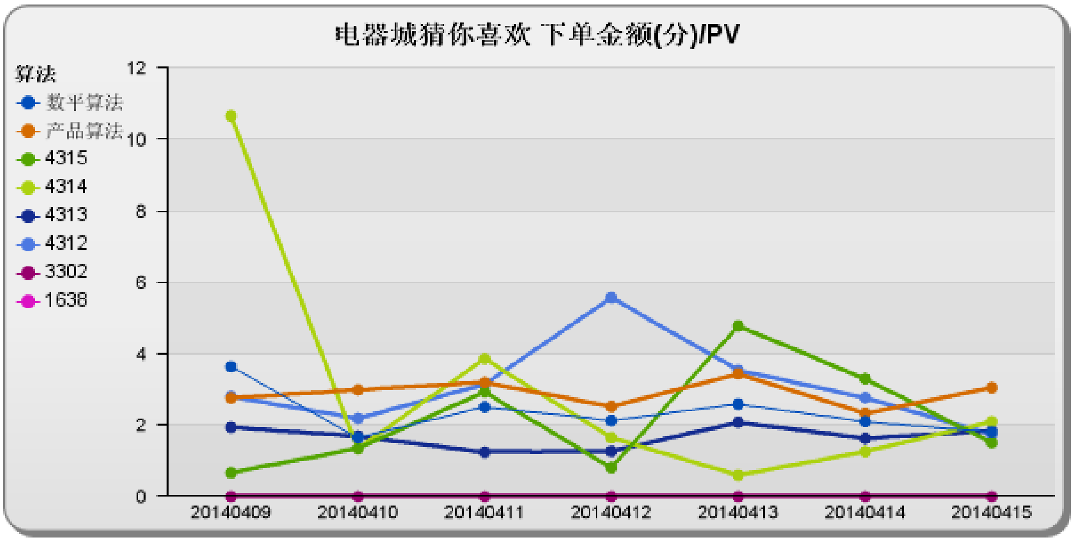

Figure 4.4: OAPI of online algorithms from 0409 to 0415.

the mid levels. This is because CBMF didn't optimize for CTR, it is designed for improving purchase rate. But because there is some similarity between click and purchase, so CBMF's CTR performance could not be too bad.

For other algorithms, we can see personalized algorithms perform better than global algorithm. 4315 has the lowest CTR because it just provide same recommendations to every user. Redirecting seems to work very well in CTR, since users may like to click items they are familiar with. Especially in our scenerio, the users are chatting while a panel full of viewed item poped out, he/she may not close it immediately. Instead, he/she may just review the items they clicked before.

4.5.3 Order amount per impression

Algorithm ID	Average OAPI in Figure 4.2	Average OAPI in Figure 4.3	Average total OAPI
4315	2.21	2.87	2.64
4314	**3.33**	2.65	2.86
4312	3.25	**3.97**	**3.65**
4313	1.93	2.31	2.12
cpsf	3.04	2.98	3.01

Table 4.1: Average OAPI of online algorithms.

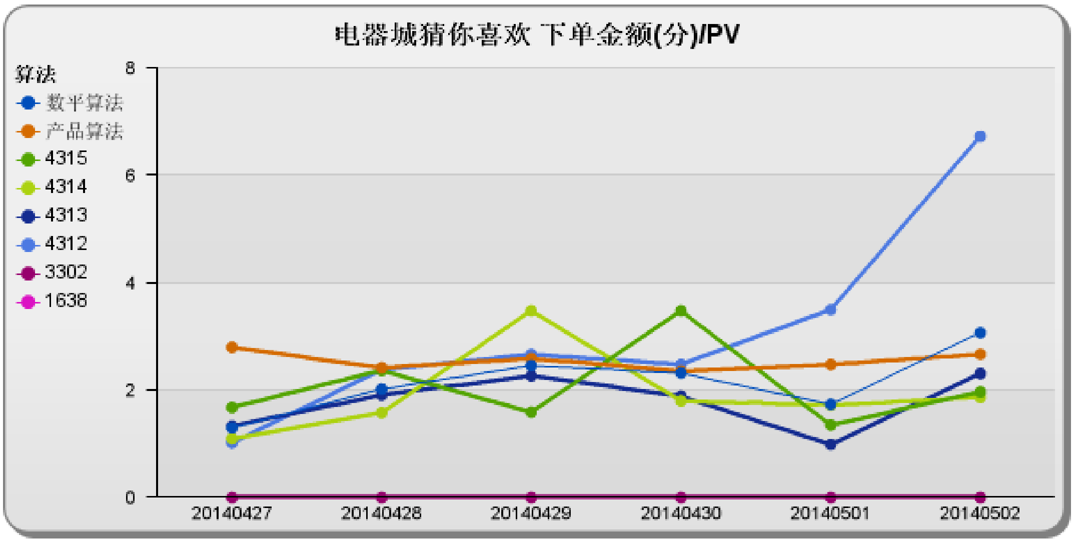

Figure 4.5: OAPI of online algorithms from 0427 to 0502.

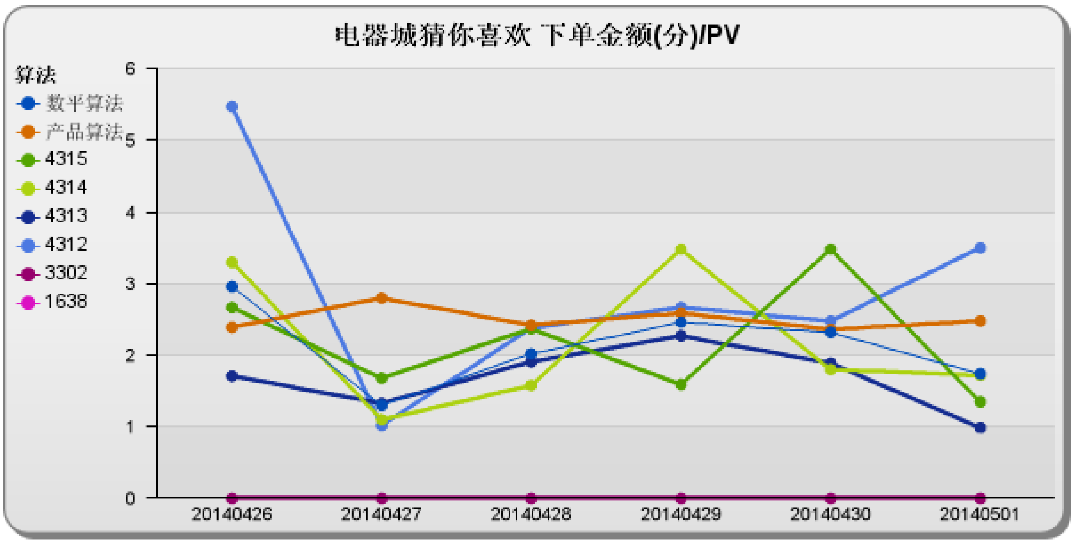

Figure 4.6: OAPI of online algorithms from 0426 to 0501.

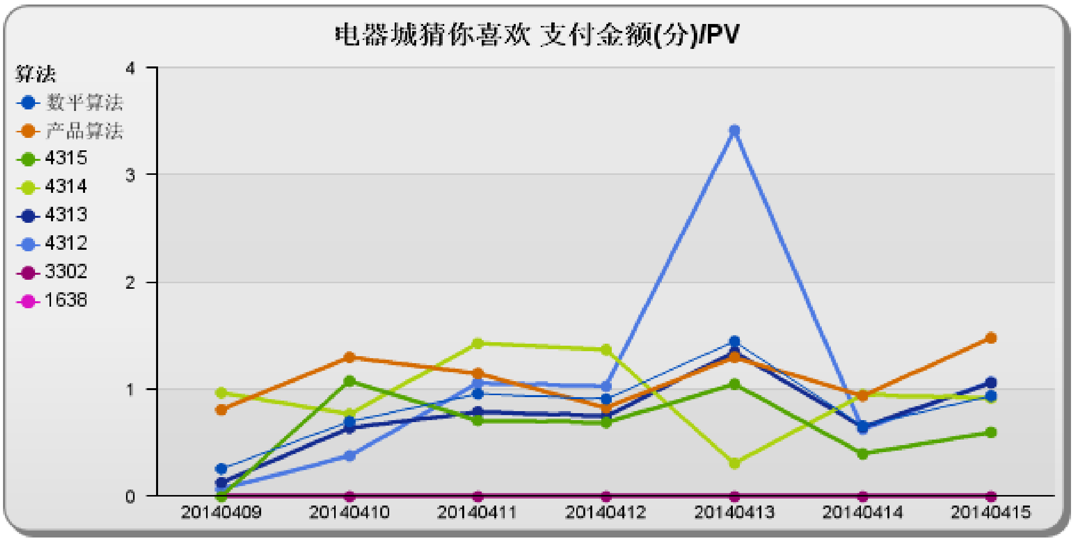

Figure 4.7: PAPI of online algorithms from 0409 to 0415.

In Figure 4.4 and Figure 4.5, OAPI of different online algorithms are shown. The id of CBMF is 4312, the blue line. In Figure 4.4, we can see that CBMF is relatively stable and its performance are among the best. In Figure 4.5 , CBMF outperforms other algorithms significantly. In Table 4.1, we can see that although CBMF may not be the best in one week's performance due to insufficient data. But its overall OAPI is the highest.

If we look at the Figure carefully we can find that, CBMF tends to achieve better performance in holidays(4-11, 4-12, 5-1, 5-2 are all Chinese holidays). This may because in holidays, users tends to buy something which they are long for while in ordinary days users may buy some necessary first. CBMF captures users' desire to buy something they like instead of they need, so CBMF will achieve better performance in holidays.

4.5.4 Pay amount per impression

In Figure 4.7 and Figure 4.8, OAPI of different online algorithms are shown. The id of CBMF is 4312, the blue line. We can see that CBMF outperforms other algorithms in these two periods. Different from the observations in OAPI, users may have high OAPI in Mondays and Sundays, that may because users may not pay right after they place their order. After thinking carefully,

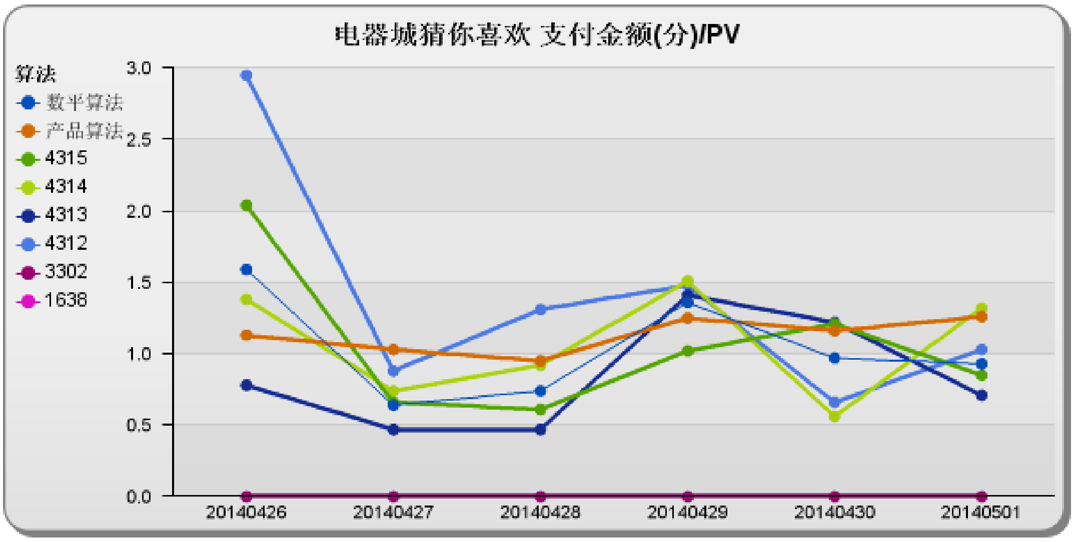

Figure 4.8: PAPI of online algorithms from 0426 to 0501.

they may finally pay which leads to the delay effect in OAPI.

Overall evaluation

From the three evluation metrics, we can see CBMF can acheive better conversion rate, moderate click through rate comparing to others.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this thesis, we proposed to perform knowledge transfer for one-class CF problems using matrix factorization and came up with a matrix tri-factorization method and a online framework to systematically study on the factors that will affect selection. We found although there exists some methods which tackle the one-class CF problem and there are also some transfer learning methods for CF. But no one has deal with one-class CF in multiple domains. Simply apply former transfer learning methods will fail due to data sparsity. We found under matrix tri-factorization framework(TRIMF), we can transfer as much knowledge as we can while ignore the noise. By leveraging overlapped users and items, we can transfer knowledge from different domains. While applying the linear control factor to pattern matrix, we can avoid direct transfer which can bring noise while capture the similarity between different domains. To put our method in reality, we developed a clustering based matrix factorization framework(CBMF) which automatically integrate all data together then perform matrix factorization. The experimental results for TRIMF in real-word data sets showed that our method performs better than several state-of-the-art methods in conversion rate comparison. The experimental results for CBMF in real-word showed that our method has the best conversion rate and moderate click through rate among others.

However, we notice that there are limitations in the work. First, in TRIMF, the computational cost is expensive since multiplicative rules will affect all matrix in update time. Second, we only support non-negative matrix factorization in TRIMF, because we need to constrain non-negative to fulfill optimization conditions. If matrix can be negative, it'll be more flexible and can carry more information. Third, both TRIMF and CBMF are point-wise methods which optimize for each entry of the matrix, actually we only need to rank those items not to calculate their score. That is, we only need their relative relationship [32]. Fourth, TRIMF and CBMF are batch updated algorithms, but in online test almost all algorithms whose performance are good are real-time.

We believe that Transfer Learning for One-Class Recommendation has practical applications in the real world and would be a promising research topic. TRIMF/CBMF is our initial attempt on this topic. In the future to make it more robust, we propose the following approaches:

- **Pair-wise Transfer Learning in CF.** Instead of point-wise transfer in CF, pair-wise CF is becoming more and more popular because they can almost achieve better results. In [16, 1] pair-wise CF is applied in implicit feedback. In transfer learning, integrating matrix factorization and pair-wise CF can be the future work.

- **Online Transfer Learning in CF.** There are little work on large scale transfer learning, but it is badly desirable. In real-world, online recommendation algorithms often dominant off-line ones. Our method CBMF is a batch-updating algorithm which updates per hour, but not real-time. It would be our future work to make a real online transfer learning algorithm in CF.

- **Transfer Learning in CF with multiple matrix.** In CBMF, data from different sources are integrated into one unify matrix. Although very carefully, we can still lose or misuse the data. If we can run our algorithm fast on their original data, then we don't need to integrate.

- **Time Complexity Optimization in CF.** In [36], an interesting relationship is shown: more data can faster training speed while getting the same performance on test data. In CF there are many data that we need plenty of time to deal with. If we could leverage all of them without increasing our training time or model complexity, we could use as much data as possible.

REFERENCES

[1] Fabio Aiolli. Convex auc optimization for top-n recommendation with implicit feedback. In *RecSys*, pages 293–296, 2014.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

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

[8] 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.

[9] Hoda Eldardiry and Jennifer Neville. Across-model collective ensemble classification. In Burgard and Roth [2].

[10] 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.

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

[12] 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.

[13] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.

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

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

[16] Lukas Lerche and Dietmar Jannach. Using graded implicit feedback for bayesian personalized ranking. In *RecSys*, pages 353–356, 2014.

[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] Christopher H. Lin, Ece Kamar, and Eric Horvitz. Signals in the silence: Models of implicit feedback in a recommendation system for crowdsourcing. In *AAAI*, pages 908–915, 2014.

[21] 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.

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

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

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

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

[30] 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.

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

[32] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press.

[33] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

[34] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey E. Hinton. Restricted boltzmann machines for collaborative filtering. In Ghahramani [10], pages 791–798.

[35] 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.

[36] 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.

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

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

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

[40] 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.

[41] 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.