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 have been much harder without the guidance of my supervisor, assistance from my friends and support of my parents.

Firstly, I would like to express my deepest gratitude to my supervisor Prof. Qiang Yang for his instruction on both my research and my life. Without his guidance, I would never have the chance to exploit so many possibilities, in particular my interest in recommender system.

I would also like to express my greatest appreciation to Lili Zhao and Liya Ji for their companionship and valuable suggestions on my study, also for their kindness and tolerance as my roommates.

In addition, I would like to thank Ben Tan, Bo Liu, Bin Wu, Kaixiang Mo, Zhongqi Lv, Lianghao Li for their ideas, suggestion and company. Also thanks to Dr. Wei Xiang and Dr. Qian Xu for their guidance on my future career. 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 on my thesis examination committee. Many thanks to the CSE staffs, Mr. Isaac Ma and Ms. Connie Lau, for their administration work.

Finally, I am thankful to my mother for her support and belief in me.

I 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

The One Class Recommender System aims at predicting users future behaviors according to their historical actions. In these problems, the training data usually only contains binary data which reflects behavior that has or has not happened. Thus, the data is sparser than traditional rating prediction problems. There are two current ways to tackle the problem: first, using knowledge transferred from other domains to mitigate the data sparsity problem and second, providing methods to distinguish negative data and unlabeled data. However, it is not easy to transfer knowledge simply from a source domain to target domain since their observations may be inconsistent. In addition, without data from an external source, distinguishing negative and unlabeled data is sometimes infeasible.

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

CHAPTER 1

INTRODUCTION

1.1 Motivation

Recommendation systems have become extremely popular in recent years. Typical recommendation systems recommend items (movies, music, books, etc.) that users may be interested in. Collaborative filtering approaches build a model from users past behavior to predict items they may have an interest in. In real-world recommendation systems, the number of users or items is huge. Users can only rate a small fraction of items, thus the user-item matrix is extremely sparse. What is more, we sometimes cannot observe explicit ratings as only implicit behavior is provided (e.g. click, pageview and purchase). Such a problem may lead to poor performance in CF models.

Different transfer learning methods have been developed recently to improve the performance of the model.In [19, 20], they use a rating-pattern sharing scheme to share user-item rating patterns across different domains. In [31, 28], an implicit dataset is available and knowledge is transferred via latent-feature sharing. In [42, 10] the authors 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 and the rating is usually trustworthy and informative. It can thus be used to recommend books to the same user. However, in a website where only implicit feedback is available (e.g. advertisement clicks), the behavior can be much more noisy and with less information. To achieve better performance, we must transfer more knowledge from source domain while being very careful about the noise.

Some works have been done on solving one-class recommendation problem [14, 26, 22]. They all model the frequency of actions by a confidence matrix. For example, if you clicked on item A 10 times, item B one time, there can be certainty that you like A, but uncertainty of whether you like B. On the other side, if you are an experienced user and you did not click a popular item A, then it is highly probable that you did not like A. However, these works only explore the original user-item matrix, in the real-world there are enormous amounts of other useful information which can be used to improve performance.

We collect several users clicking and purchasing behaviors from an online shopping site. After careful analysis, we find that users clicking and purchasing behaviors may be 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 the click-through-rate/conversion-rate).

Further, to make the method scalable and able to put online, we develop a clustering-based matrix factorization method (CBMF) using Hadoop. CBMF collects all kinds of user data and convert them into a single matrix per task. For cold-start users, a weighted recommendation from their neighbors is 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 the user side and item side (TRIMF).

- Second, implicit datasets consist too much noise. To transfer trustful knowledge, we develop a clustering-pattern transfer function. For each task, we provide a clustering pattern mapping function, which only does 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, and it's performance is among the best of 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 of the proposed matrix tri-factorization method (TRIMF) in detail in Chapter 3. We present details of our proposed CBMF framework in Chapter 4 . We perform two experiments, offline and online, in Chapter 5. Finally, we share our thoughts on possible future work and conclude the thesis in Chapter 6.

CHAPTER 2

BACKGROUND

In this chapter, we give a brief review of the related literatures. We classify our work as 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, or use 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.

<table>
<tr><th></th><th>Rating-Pattern Sharing</th><th>Latent-Feature Sharing</th><th>Other</th></tr>
<tr><td>$Rating \to Rating$</td><td>RMGM [19]</td><td>CMF [40]</td><td></td></tr>
<tr><td>$Implicit \to Rating$</td><td></td><td>CST [30], TCF [29]</td><td>TIF [31]</td></tr>
<tr><td>$Implicit \to Implicit$</td><td colspan="2">TRIMF</td><td></td></tr>
</table>

2.1 Collaborative Filtering

Collaborative filtering ([17], [34]) as an intelligent component in recommender systems ([43], [24]) has gained extensive interest in both academia and industry.

Collaborative filtering(CF) methods are based on collecting and analyzing information on users behaviors, activities or preferences and predicting what users will like in the future based on similar users. The underlying assumption of the CF approach is that people who agree in the past will agree in the future too and they will like similar items to those they liked in the past. For example, a CF recommendation system for television could predict which show a user should like given a partial list of this users tastes (likes, dislikes or ratings, etc.).

There are three types of CFs: 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, is effective given plenty of records and doesnt produce a model. Typical examples of this mechanism are the neighborhood based CF and item-based/user-based top-N recommendations [41].

The advantages of this approach include:

- The interpretability of the results, which is an important aspect of recommendation systems.
- Ease of setup and use.
- New data can be added easily and incrementally.
- Content of items being recommended need not be considered.
- Mechanism scales well with co-rated items.

However, there are several disadvantages with this approach:

- Large numbers of human ratings are required.
- 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 the representation usually relies on a specific vector space. This would require including a new item and re-inserting all the elements in the structure, preventing the scalability of the 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 [17, 31, 32, 34], probabilistic mixture models [13, 16], Bayesian networks [33] and restricted Boltzman machines [37].

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 has petabytes of records as input; on the other hand, the convergence of most models requires intensive computation. There needs to be a tradeoff between prediction performance and scalability. Given the accuracy of model-based CFs, how to overcome the scalability issue has attracted much attention. With the rapid development of computation, researchers have been exploring the use of parallel systems to speed up complex model building. For example in [5], 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 [7] paradigm. In [39], 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 increase your training speed.

The update step is hard to parallelize in our method TRIMF. To get it online, we develop a varied version of TRIMF(CBMF).

2.1.3 Hybrid models

A number of applications combine memory-based and model-based CF algorithms. These overcome the limits of native CF approaches and improve the prediction performance. Importantly, they overcome CF problems such as sparsity and loss of information. However, they have increased complexity and are expensive to implement. Most commercial recommender systems are usually hybrid, for example, Google news recommender system [6].

2.2 Matrix Factorization

In the mathematical discipline of linear algebra, a matrix factorization is a factorization of a matrix into a product of matrices. There are many different matrix factorizations; each finds use among a particular class of problems. Matrix factorization models map both users and items to a joint latent factor space of dimensionality f, such that user-item interactions are modeled as inner products in that space. There are two main methods of matrix factorization 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 [38]. $\sum$ consists of singular values of M, and we can select the k-biggest values and set other entries of $\sum$ to zero. Then put $M' = U \sum V$, since M' is our recommendation result.

2.2.2 Non-negative Matrix Factorization

Non-negative matrix factorization (NMF) is a group of algorithms in multivariate analysis and linear algebra where a matrix V is usually factorized into two matrices W and H, with the property that all three matrices have no negative elements. It can be regarded as a latent factor model [17].

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 a 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 decomposes 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, and H is a $d * n$ matrix. Usually $d \ll m, n$, is the dimension of the 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 the inner product.

2.2.3 Non-negative Matrix Tri-factorization

As a transformation of NMF, Non-negative matrix tri-factorization(NMTF) decomposes a matrix X into three non-negative parts : $X = USV$. Instead of mapping users and items to a common 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 [9], the authors proved that NMF is equivalent to k-means clustering. In [8] the authors also proved that NMTF can be regarded as a way of clustering. NMTF is well known in document processing, [21] uses prior knowledge in lexical and NMTF to tackle the sentiment analysis problem, [44]and exploits the relationship between word clusters and document classes in text classification problem.

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

2.3 Transfer Learning

Pan and Yang [27] 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 recommendation, users and items form a joint distribution in training data. However, in test data, users may be different as might the items; their relationship may vary as well. 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 [25] 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 a percentage of the users are aligned, which does not distinguish the consistency of users' preferences among the aligned users. The authors in [40] designed a collective matrix factorization framework, where two matrices 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). Based on which, are some follow-up works in cross-domain collaborative filtering using matrix factorization techniques. Li *et al.* [20] designed a regularization framework to transfer knowledge of cluster-level rating patterns, where they use matrix tri-factorization and cluster level rating patterns are shared. Pan *et al.* [29], [30] used a matrix factorization framework to transfer knowledge in a latent feature space. Knowledge is transferred from an implicit domain to an explicit domain, but this method cant handle knowledge transfer between implicit domains. Cao *et al.* [4] exploited correlations among different CF domains via learning. They factorize each matrix X_d by $X_d = F_d G_d^T$ where F_d and G_d are the user and item latent vectors, 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) carefully adopts rating patterns and latent feature sharing by designing a matrix tri-factorization framework. It can handle knowledge transfer between implicit domains and can be set to suit different tasks.

2.3.2 Large Scale Transfer Learning

Thus far, transfer learning has mostly been considered in the off-line learning settings, which do not emphasize scalability and computation speed. Due to the rapid development of storage techniques 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 the previous literature, though it is badly needed. To cope with the growing needs of todays recommendation system, we would like to discover the parallelizing possibility in our experiments. There are already some researchers working on large scale collaborative filtering, the authors in [6] designed a map-reduce framework for online news recommendation. In our approach, we have developed a parallel framework CBMF and put it 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, which yields 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 sorts all key-value pairs according to their key, thus in the reduce step, the 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, that is, the answer to the problem it was originally trying to solve.

We show that our method (CBMF) can be plugged into the Map-Reduce framework for parallelization in Chapter 4.

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 behaviors 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%). Therefore, if we only use the available information, we cannot achieve a reliable or even reasonable performance.

In an online shopping site, there are two main actions - click and purchase. Both can form a matrix which consists of 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 and although click matrix X_c is also sparse, it is much denser than X_d. To use more data, we develop a transfer learning algorithm(TRIMF) that leverages click data to predict purchasing. Compared with former methods which only share either rating patterns or latent features , our method shares both rating patterns and latent features through cluster-level transformation and overlapping matrices. Experiments in 5.1 show that our algorithm performs better than 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 partially overlap.

- 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 [14], [26] 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 [14], the authors consider actions that happen more than once(e.g. multiple clicks on an item). Negative entries are ignored; for each positive entry, its weight is proportional to its frequency, since higher frequency can mean that we are more confident about the entry. For example, user i viewed item j, n times, then $W_{ij} = 1 + log(n)$. In [26], positive entries all have same weight 1, while negative entries are considered differently. According to their experiments, the user-oriented weighting scheme can achieve the best performance. That is, for negative entries $W_{ij} \propto \sum_j R_{ij}$, the idea is that if a user has more positive examples, it is more likely that the user does not like the other items, that is, the missing data for this user is negative with higher probability.

In our method, we adopt these weighting schemes to give missing values proper weights, that is, for positive entries we use the weighting scheme in [14] and for 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. For instance, users will buy n_d items in one day while clicking n_c items which often results in $n_d \ll n_c$. Therefore, only using deal data is not sufficient. Traditional transfer learning methods use matrix factorization and share a certain part of low-rank matrices to achieve knowledge transfer(e.g. user-latent factor, rating pattern). However, none of them apply the selective-sharing scheme as ours does.

In TRIMF, rating matrices 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 the latent factor, while S stands for the clustering relationships between user clusters and item clusters. We want to learn a better cluster-level rating pattern S from users' deal data with the help of users' click data, not just use users' deal data. Therefore we factorize two matrices X_c, X_d together. In order to transfer knowledge, we must make sure that their latent spaces are the same. For a user who has click and deal actions, it would be particularly beneficial that his latent vectors factorized from X_c, X_d are the same.

Therefore, for overlap users and items, we want their clustering vector U, V to be the same. What is more, we want even more knowledge transfer from the matrix S which stands for cluster relationship or rating patterns. However, what a user likes to click is not always the item he wants to buy. In Yixun, there are only two common items in the top 10 click items and top 10 purchase items (Table 3.2.2). Therefore these rating patterns should somehow be related but not the same. We cannot simply make the pattern matrix S the same in the 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 belong to the same category with a higher conversion rate (user tends to buy after clicking), but not with other categories. There are also some users who like window-shopping while others buy an item straight after clicking. These are all cluster-level features. We design a mapping function to allow the learnt S better suit the data.

We design two mapping vectors: U, V, if we have learnt a rating pattern S_c from a 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 being multiplied by U and V we can capture the difference between them, at the 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 are the weights for X_C, X_d, every observed entry has weight $1+log(frequency)$. While others have weight $W_{ij} = \sum_j I(R_{ij})$.

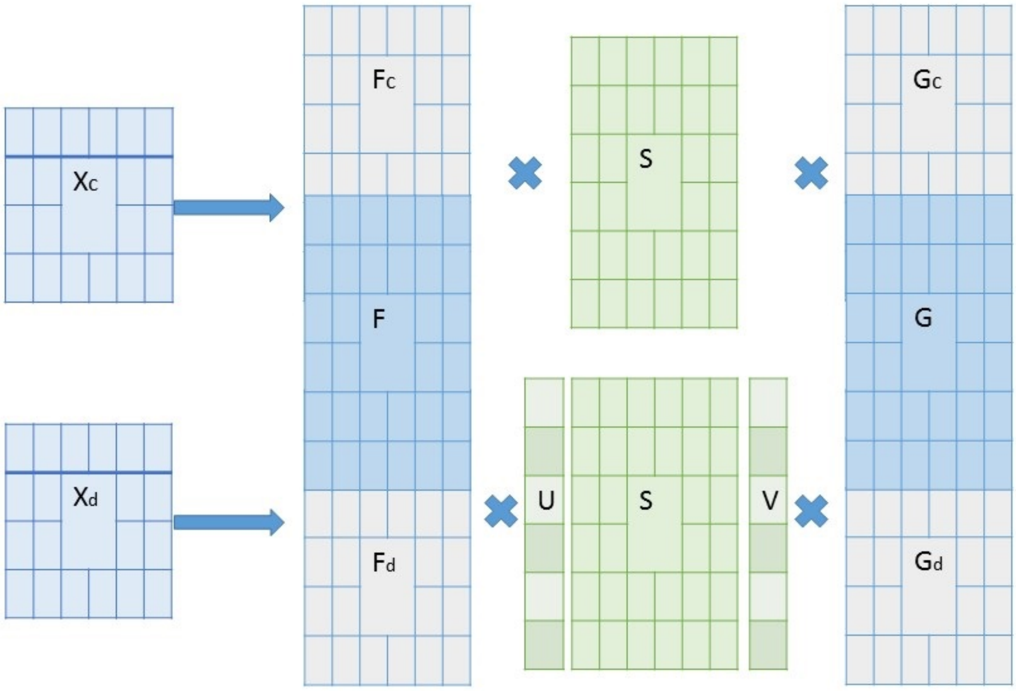

Figure 3.1: Graphical model of TRIMF.

- F, G are the soft clustering result matrices for overlapped users(items), they are forced to be the same. F_c, F_d, G_c, G_d are matrices for unique users(items).

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

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

3.3 Solution to TRIMF & Algorithm

Following the update rules in [44], we use an alternately iterative algorithm to solve the objective function.

Firstly, we declare some denotations:

- 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 iterations 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 also much smaller than n, m. Through using sparse matrix multiplications and avoiding dense intermediate matrices, the update steps 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 reaches 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

CHAPTER 4

CLUSTERING-BASED MATRIX FACTORIZATION FOR ONLINE SHOPPING PREDICTION

4.1 Limitation of TRIMF

In Chapter 3, we introduced TRIMF, which is a matrix tri-factorization method for cross-domain recommendation. However, it has some limitations which restrict its scalability and extensibility. First, when data are coming from multiple sources (e.g. click, pageview and add cart), TRIMF treats every source equally and puts each of them into a matrix which is very sparse. When solving the object function, increasing the matrixes will increase the time and space complexity. If we try to update S, every matrix is included so it will be quite time-consuming. What is more, in reality we cannot ignore users with fewer actions. Thus, the matrix will be much more sparse than the ones in our experiment, so we cannot guarantee achieving equal performance.

To solve these problems, we have developed a framework based on a clustering and scoring scheme (CBMF, Figure 4.1). CBMF first clusters users according to their behaviors and demographic features, then automatically converts different types of actions into one matrix, called action matrix. Finally a matrix factorization method is applied to the action matrix. For users with adequate actions, a personalized recommendation is provided. Otherwise we provide a recommendation based on their clusters.

We conduct two experiments:

- offline experiment: we select datasets used in TRIMF, and compare the run-time and precision of the two methods.

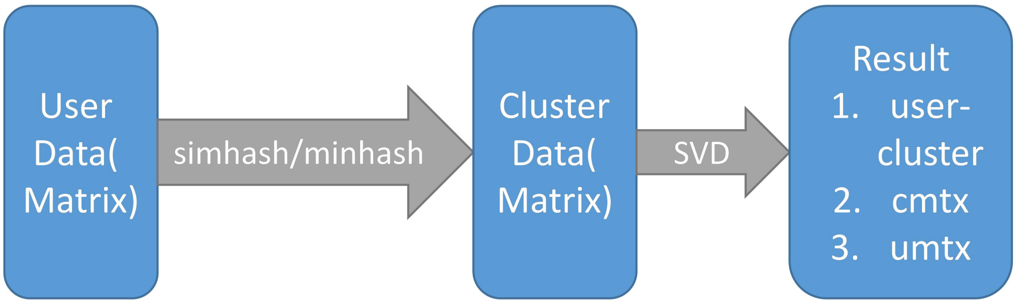

Figure 4.1: Framework of CBMF.

- online experiment: we do A/B test with CBMF in an online shopping site.

Results show that in offline datasets, CBMF runs much faster then TRIMF while achieving equal performance. In online test, CBMF outperforms other methods in conversion rate.

4.2 Clustering method in CBMF

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

4.2.1 Simhash

Simhash is a kind of locality sensitive hashing(LSH). LSH is a hashing method where if we got two points A, B which are close in their original space, after hashing we would get A', B', with A', B' still close in the new space. Thus we keep the relationship of distance among the 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 converts a vector x into a 32-dimension binary vector x'. Actually i_{th}

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

bit of x' is the sign of the 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 vectors 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 new space to represent their similarity in original space.

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. Regardless, calculating all similarities in real-time is nearly impossible. 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 a probability proportional to the overlap between the set of items that these users have voted for (clicked-on). In CBMF, Minhash is applied after Simhash and every user vector consists of 32 bits.

The basic idea in the Minhash 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}. It is also easy to prove that the probability of two users having the same hash function is exactly equal to their similarity or Jaccard coefficient.Similar to [15], 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 the concatenated hash-key is equal to $S(u_i, u_j)^p$. In that case, 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 clusters will increase.

4.2.3 Simhash & Minhash using MapReduce

MapReduce is a 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 easily be 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 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 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 pair ($user - id$, u_i).

Reduce phase

In the reduce phase, our input takes two forms: ($Mnhs_i$, u_i) represents cluster-id and user vector. ($user - id$, u_i) represents an experienced user and his vector.

- In the cluster case: for each cluster-id we obtain 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 represent all users in it. Thus we simply add scores from users to the joint vector. We, then do a normalization sothe range of the vector is between 0 and 1. The output has two parts:

  - users and the cluster they belong to (userid, clusterid).
  - cluster-ids and their vectors (cluster-id, cluster-vector).

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

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

4.3 Feature construction in CBMF

After clustering, we have many clusters and their corresponding actions, including click, purchase and pageview on different (overlapping) 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 used on item n. However, simply creating such a matrix may lead to data sparsity problems. This is especially true in the matrix standing for purchasing actions, even though similar users are alredy clustered together.The data is still very sparse(0.01%) which may constrain our model from providing a reasonable recommendation.

In CBMF, a scoring scheme is applied for each kind of action to put every action into a single matrix with a 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), the score that 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, while U denotes the entire user set. 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 these items can then be approximated by $Cvr(n_{click}) \approx \frac{|U_n \cap U_n^{click}|}{|U_n^{click}|}$we compare the conversion rate of users with this action with the average, and 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 use 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. w_i is the percentage of users who have this action compared to all users. The result, $X_{mn} = \frac{\sum_{i=1}^{4} w_i * s_i}{\sum_{i=1}^{4} w_i}$.

4.4 Matrix factorization in CBMF

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

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

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

Datasets	Yixun	Tmall
users(click)	16,240	884
items(click)	1,932	9,531
sparsity(click)	0.006	0.021
users(purchase)	2,520	840
item(purchase)	1,791	4,312
sparsity(purchase)	0.0003	0.001

Table 5.1: Dataset characteristics.

CHAPTER 5

EXPERIMENTS

5.1 Offline Experiments

5.1.1 Datasets

- **Yixun**: A dataset from Yixun[1], a large online retailer for electronic products, that has been sampled from log data from two weeks(20130801 to 20130814). For click and purchase behaviors, we created a subset for which there are at least 10 interactions per user and item.

- **Tmall**: A similar but smaller anonymized dataset from a shopping portal[2] containing user interactions of different types(click and purchase). For each of the actions, a time stamp is available. It is a relatively long-term data, actions are recorded from 20140103 to 20140704.

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

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

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

Table 5.2: Baseline methods.

5.1.2 Metrics

We use $prec@5$ and $prec@10$ as our evaluation metrics. Precision is the fraction of clicked items that are shown to the user.

$$Precision = \frac{\|clickeditems\| \cap \|shownitems\|}{\|shownitems\|}$$

Precision takes all retrieved documents into account, but it can also be evaluated at a given cut-off rank, considering only the topmost results returned by the system. This measure is called **precision at n** or $prec@n$. $prec@n$ is widely used in information retrieval for evaluating the ranked documents over a set of queries. We use it to assess the overall performance based on precisions at different recall levels on a test set. It computes the mean of precision over all users in the test set.

In the Yixun dataset, data from the first week is used as training data while we perform testing on data from the second week. For Tmall dataset, we select the last five purchased items from each user as their test data while training on their previous data.

Our main goal is to optimize for the conversion rate (future purchase matrix), so the test is mainly done in the purchase matrix. However, since TRIMF can also optimize for the source domain (click matrix), some tests in the click matrix are also conducted.

5.1.3 Baseline methods

We divide baseline methods into non-transfer methods and transfer methods. All baseline methods are shown in Table 5.2.

non-transfer methods

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

- Most Popular: selects top-n items globally, and provides the same recommendation results for every user.
- Singular Value Decomposition(SVD) [32]: is a typical method used in recommender system, here PureSVD from Matlab is used.
  - rank = $\{5{,}10{,}20{,}30{,}40{,}50\}$
- Non-negative Matrix Factorization(NMF) [17]: is also a typical method used in recommender system, here NMF from Matlab is used.
  - rank = $\{10{,}20{,}40{,}60{,}100\}$
- Probabilistic Matrix Factorization(PMF) [36]: is a recently proposed method for missing value prediction. Previous work showed that this method worked well on the large, sparse and imbalanced dataset.
  - rank = $\{10{,}20{,}30{,}40{,}50\}$
- BPRMF [35]: BPR is a generic optimization criterion for personalized ranking. It is a maximum posterior estimator derived from a Bayesian analysis of the problem. Unlike traditional methods whose objective function is point-wise, BPR is a pair-wise object function. BPRMF implements BPR using matrix factorization.
  - We initialized BPR with the most popular results.
  - We set $iteration = \#n * 100$, ($\#n$ in the number of observations)
- WRMF [26]: One-class collaborative filtering(WRMF) is a weighted low rank approximation method optimized for an implicit dataset.

- rank = {5,10,15,20,25}

transfer methods

- Collective Matrix Factorization(CMF) [40]: is proposed for jointly factorizing two matrices. 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 matrices of the same dimensions(in order to share a latent factor) by padding zero rows & columns.
  - Shared latent space dimension = {5,10,15,20,25}
- TCF [29]: is a transfer learning method used to predict missing ratings via heterogeneous feedback. It is originally designed for rating prediction, so we set the deal matrix with randomly sampled zeros as the rating matrix, and click matrix as the implicit feedback matrix. Zero rows and columns are also padded to make the two matrices of the same dimensions.

our methods

- TRIMF
  - We set latent factor = 30, iteration = 200.
- CBMF
  - We set latent factor = 15, iteration = 200, and we run Minhash for five times and concatenate them as our cluster id.

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**
CBMF	**0.512**	**0.403**

Table 5.3: Performance comparision on Yixun users who have deal data.

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
CBMF	**0.0181**	**0.0144**

Table 5.4: Performance comparision on Yixun users who have click data.

5.1.4 Results

Yixun

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

Results are shown in Tables 5.3 and 5.4.

Tmall

Since the users are manually selected, click matrix and deal matrix have the same number of rows. We only need to conduct a test on X_d. The result is shown in Table 5.5.

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**
CBMF	**0.00612**	**0.00503**

Table 5.5: Performance comparision on Tmall users.

5.1.5 Performance comparison & analysis

First, we observed that TRIMF out-performs all other baseline non-transfer methods in three tests. In the Yixun test, we can see traditional CF methods which aim at rating prediction (e.g. SVD, NMF) cannot achieve a more compatible performance than others. This is because these methods are designed for matrices with multiple values, not for our binary matrices, while the CF method designed for binary matrices (BPRMF, WRMF) can achieve significantly greater results. In the Tmall test, the difference is not so significant, because the data here is less sparse than the Yixun data. In other words, every method has enough data to train a good model.

Second, TRIMF also out-performs other transfer methods. Since CMF, TCF are also designed for rating prediction problems. The information in our training set is limited, so neither method can transfer enough knowledge based on their framework. TRIMF is designed for one-class transfer learning, which combines one-class and transfer methods, so it inherits advantages from both. What is more, we observed that CBMF can acheive the second best performance among all methods, very close to TRIMF.

The effects of cluster-level transformation

In our assumption, U, V are two mapping matrices that describe the difference in user clusters and item categories. To see whether U, V really reflect the phenomenon, we manually check entries in U, V with high and low values.

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

In Table 5.4, 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 the click pattern to the purchase pattern.

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 alignments, we can leverage this information to factorize or co-cluster the two matrices in a joint latent space. Making them the same space is the foundation of knowledge transfer which happens during the alignment. To see that sharing F, G really works, we randomly select another 6000 users and 1500 items from Yixun, make two new matrices X'_c, X'_d to perform another test. We try three sharing schemes:

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

The $prec@n$ here are not comparable with the first experiment in Table 5.3. Result(Table 5.6) shows that we must share latent factors carefully as randomly sharing them may harm the performance. However, sharing latent factors for overlapping users/items can achieve a significantly greater result.

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

Table 5.6: The effect of sharing.

5.1.6 Scalability

We select the Yixun dataset. We implement TRIMF and CBMF using Python 2.7 with Numpy and Scipy, and run them using a laptop with Intel Core i5-4200 with 4 cores at 2.4GHz and 12GB memory.

For each method, the update algorithm stops while training loss is less than n. To compare the training time and precision@n for each method, we conduct four different experiments based on different criteria.

- we set different threshold for n, and test the convergence speed for each method. We select 2520 users who has click and purchase data and 1791 items.
- we reduce the size of click matrix, and test the training speed regarding to dataset size.
- we set different number of parameters(latent dimension k) for each method, and test the training speed with regard to parameter size. User size and item size are 2520 and 1791.

In Figure 5.1, we can see that CBMF runs significantly faster then TRIMF in every comparision. Which shows consistency with the time complexity of TRIMF and CBMF. In Table 5.3 and 5.5, we see that the perfomance of CBMF and TRIMF are very close. Although TRIMF is slightly better in performance, but CBMF runs much faster and has good scalability. In online test, the data contains much more user and the latent dimension is larger. Therefore, we put CBMF online to perform tests.

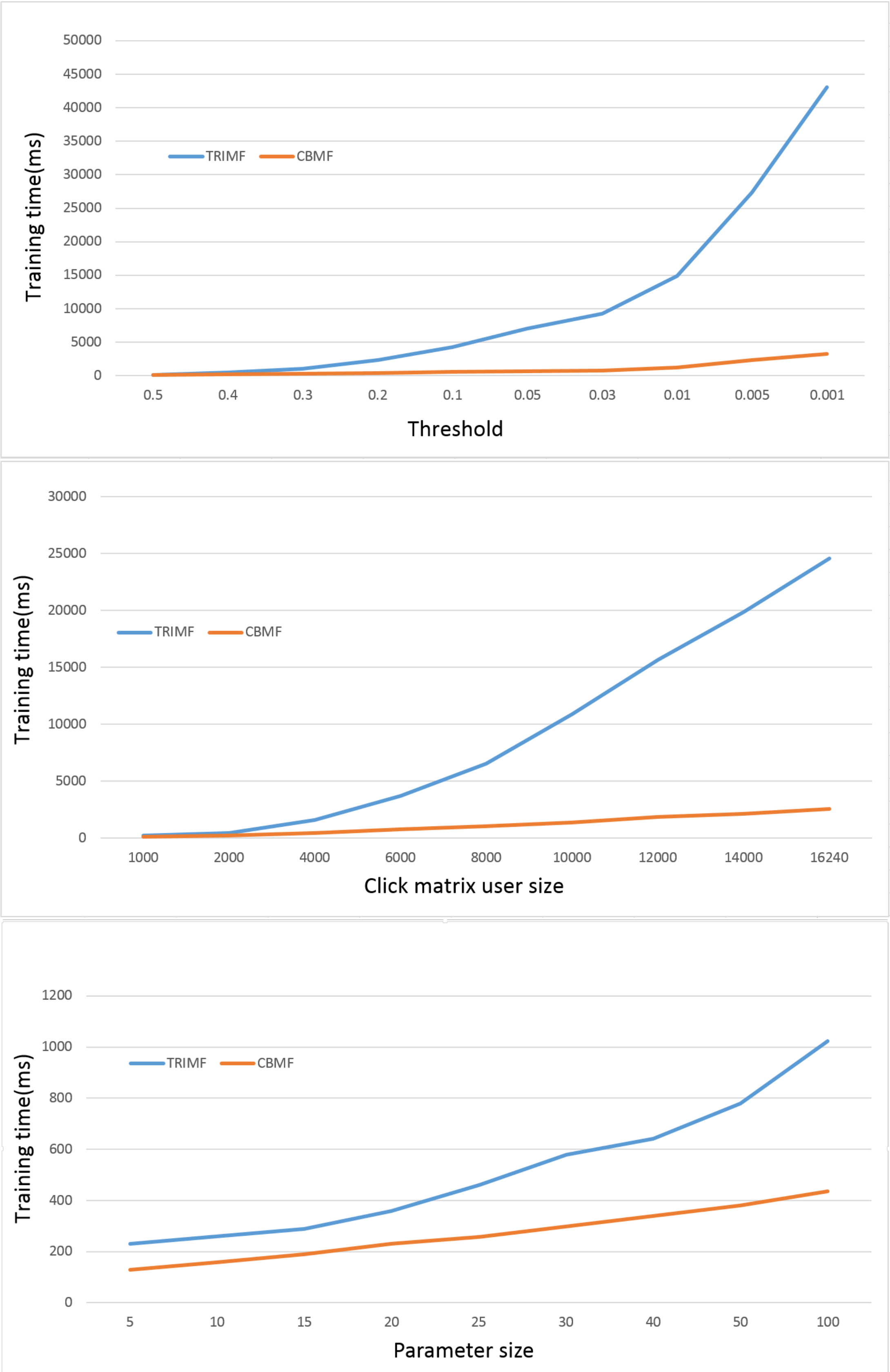

Figure 5.1: scalability comparision between TRIMF and CBMF.

5.2 Online Experiments

CBMF had been online for about one month from 2014-04-01 to 2014-05-01 with 20% users, in the chatting scenario. 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. We also need to provide results for every user;approximately 80,000,000.

After some days of parameter selection, CBMF's performance from 2014-04-10 stabilized.

There are three evaluation metrics, all based on reactions to impressions. An impression is a measure of the number of times an advertisement is seen.

- click-through-rate(CTR). The click-through rate is the number of times a click is made on the advertisement divided by the total impressions (the number of times an advertisement was served). For example, if a banner ad is delivered 100 times (100 impressions) and receives one click, then the click-through rate for the advertisement would be 1%.

$$CTR = \frac{Clicks}{Impressions} * 100\%$$

- order amount per impression(OAPI). The order amount per impression is the total price of orders divided by the total impressions.

$$OAPI = \frac{Order\ Prices}{Impressions}$$

- pay amount per impression(PAPI). The pay amount per impression is the total price paid divided by the total impressions.

$$PAPI = \frac{Paid\ Money}{Impressions}$$

These are three typical evaluation metrics in online test. CTR measures the success of an impression. While OAPI and PAPI directly reflect how much money is brought by recommendation. The difference between OAPI and PAPI is that a user may place an order but not pay (a cancellation can be made at any time).

5.2.1 Baselines

There are some others algorithms competing with CBMF, each covers $10\% - 20\%$ percentage of the users. An ID is given to every algorithm, the ID of CBMF is 4312. The details of other algorithms are shown below:

- **Co-clustering CF** [11](4313) is a scalable collaborative filtering framework based on co-clustering. Previous work showed that this method had a fast training time while providing decent accuracy.
- **Factorization Machine** [34](4314) is a generic approach that allows to mimic most factorization models by feature engineering. It provides high accuracy in several important prediction problems including recommender systems.
- **Item-based CF** [23](4315) is a typical recommending method proposed by Amazon. Cosine distance is used in calculating item similarities.
- **Efficient top-n recommendation** [1](cpsf) is a recommendation pipeline, which is the winner of the Million Songs Dataset (MSD) challenge [3]
- **CBMF**(4312): Our method.

5.2.2 Results

Click through rate

In Figure 5.2 and Figure 5.3, 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

[3]http://www.kaggle.com/c/msdchallenge

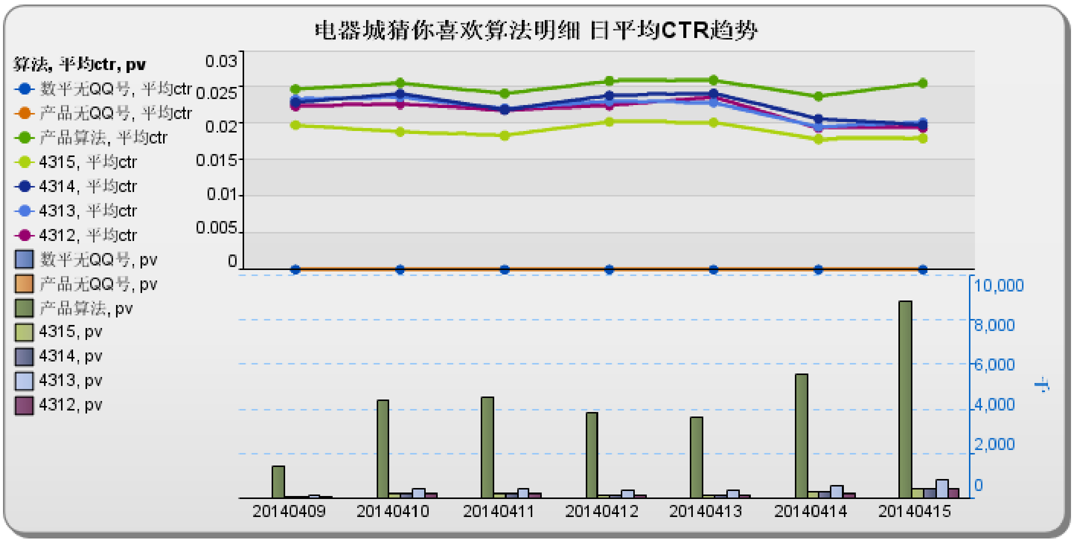

Figure 5.2: CTR of online algorithms from 0409 to 0415.

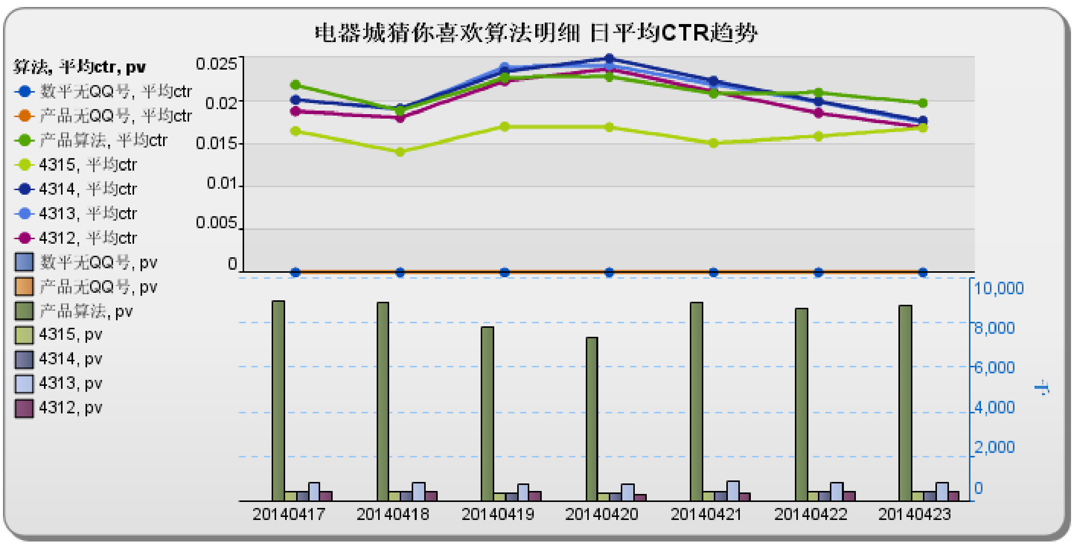

Figure 5.3: CTR of online algorithms from 0417 to 0424.

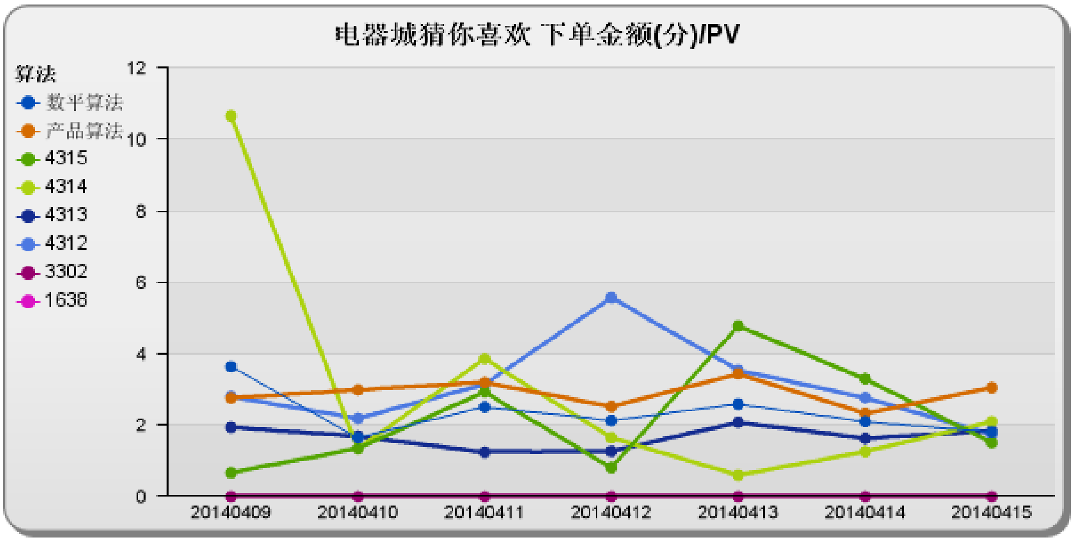

Figure 5.4: OAPI of online algorithms from 0409 to 0415.

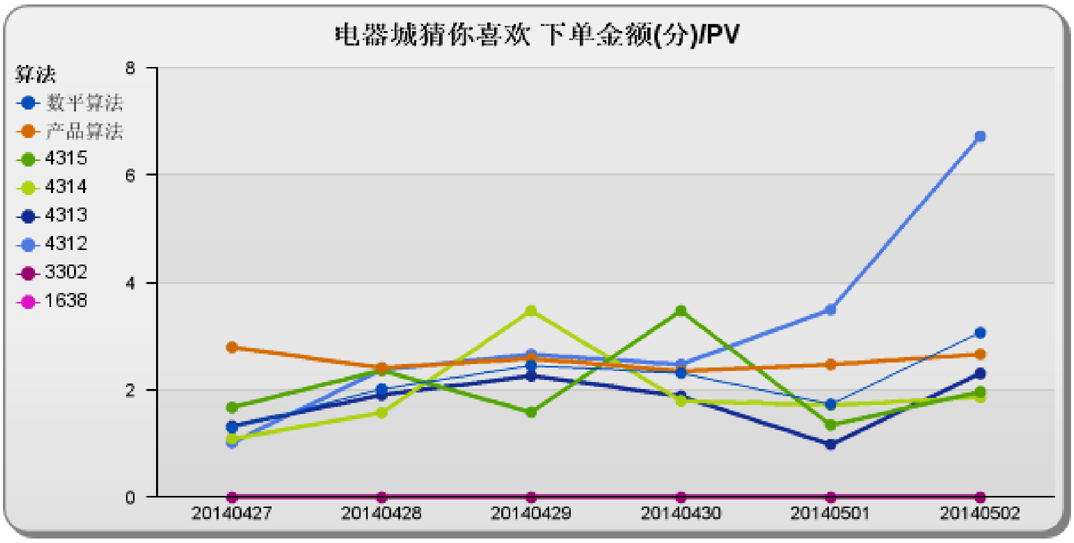

Figure 5.5: OAPI of online algorithms from 0427 to 0502.

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

Order amount per impression

In Figure 5.4 and Figure 5.5, OAPI of different online algorithms are shown. The id of CBMF is 4312, the blue line. In Figure 5.4, we can see that CBMF is relatively stable and its performance

Algorithm ID	Average OAPI in Figure 5.2	Average OAPI in Figure 5.3	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 5.7: Average OAPI of online algorithms.

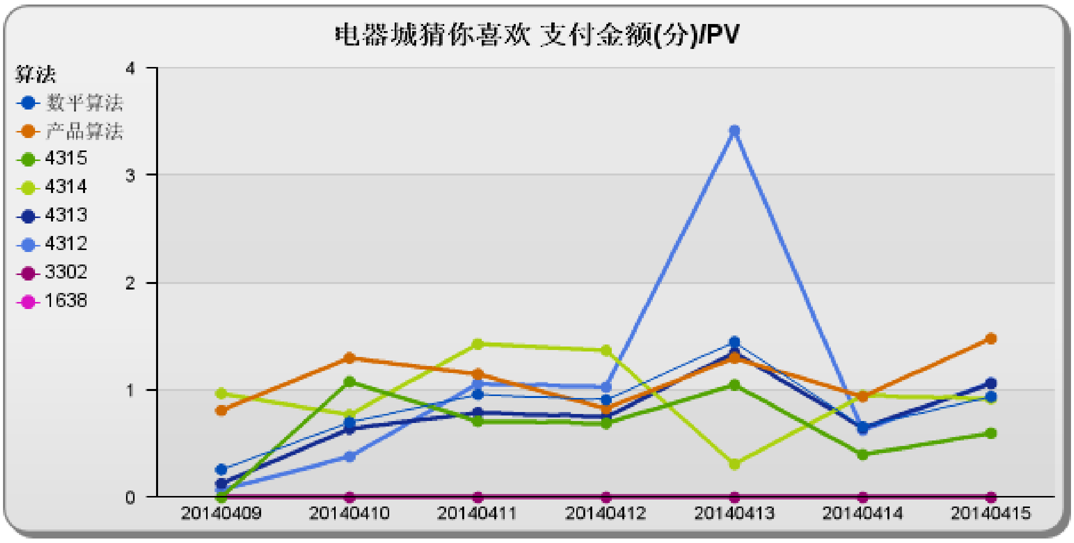

Figure 5.6: PAPI of online algorithms from 0409 to 0415.

is among the best. In Figure 5.5 , CBMF outperforms other algorithms significantly. In Table 5.7, we can see that although CBMF may not be the best in one week's performance due to insufficient data, its overall OAPI is the highest.

If we look at the figure carefully we can see that, CBMF tends to achieve better performance during holidays(4-11, 4-12, 5-1, 5-2 are all Chinese holidays). This is because during holidays, users tend to buy some things they have wanted for a long time. However, on ordinary days users may be inclined to buy only necessities. CBMF captures the users desire to buy something they like instead of what they need, so CBMF will achieve a better performance during holidays.

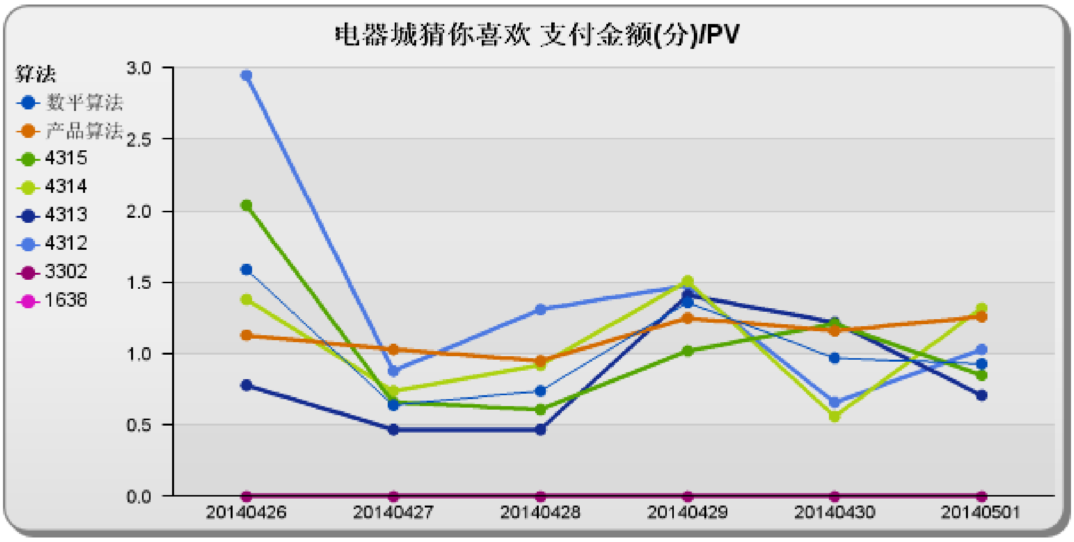

Figure 5.7: PAPI of online algorithms from 0426 to 0501.

Pay amount per impression

In Figure 5.6 and Figure 5.7, the OAPI of different online algorithms are demonstrated. The id of CBMF is 4312, the blue line. We can see that CBMF outperforms other algorithms during these two periods. Different from the observations in OAPI, users may have a high OAPI on Mondays and Sundays, which might be because users may not have to pay right after they place their order. After thinking more carefully, they might eventually pay which would explain the delayed effect in the OAPI.

Overall evaluation

From the three evaluation metrics, we can see CBMF can achieve better conversion rates of the moderate click through rate compared to others.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis, we proposed to perform knowledge transfer for one-class of CF problems using matrix factorization and came up with a matrix tri-factorization method and an online framework to systematically study 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, however, no one has dealt with one-class CF in multiple domains. Simply applying former transfer learning methods will fail due to data sparsity. We found under the matrix tri-factorization framework (TRIMF), we can transfer as much knowledge as we can while ignoring any noise. Through leveraging overlapping users and items, we can transfer knowledge from different domains. While applying the linear control factor to a pattern matrix, we can avoid a direct transfer, which can bring noise, while capturing the similarity between different domains. To bring our method into reality, we developed a clustering based matrix factorization framework (CBMF) which automatically integrates all data together then performs matrix factorization. The experimental results for TRIMF in real-world 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-world situations showed that our method has the best conversion rate and moderate click-through rate than the others.

However, we notice that there are limitations in the work. First, in TRIMF, the computational cost is high since multiplicative rules will affect all matrices in update time. Second, we only support non-negative matrix factorization in TRIMF, because we need non-negative constraints to fulfill optimization conditions. If a matrix can be negative, it will be more flexible and carry more information. Third, both TRIMF and CBMF are point-wise methods which optimize each entry of the matrix, and we actually only need to rank those items not calculate their score. That is, we only need their relative relationship [35]. Fourth, TRIMF and CBMF are batch updated algorithms, but in online tests almost all algorithms whose performances 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 represents 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 it almost achieves better results. In [18, 2] pair-wise CF is applied in implicit feedback. In transfer learning, integrating matrix factorization and pair-wise CF can be future work.

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

- **Transfer Learning in CF with multiple matrix.** In CBMF, data from different sources are integrated into one unify matrix, although it must be done carefully as we could still lose or misuse the data. If we can run our algorithm quickly on the original data, then we do not need to integrate it.

- **Time Complexity Optimization in CF.** In [39], an interesting relationship is shown: more data can train at a faster speed while getting the same performance on the test data. 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. Efficient top-n recommendation for very large scale binary rated datasets. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 273–280, New York, NY, USA, 2013. ACM.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

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

[9] 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.

[10] Hoda Eldardiry and Jennifer Neville. Across-model collective ensemble classification. In Burgard and Roth [3].

[11] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *Data Mining, Fifth IEEE International Conference on*, pages 4 pp.–, Nov 2005.

[12] 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.

[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] 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.

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

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

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, Jan 2003.

[24] 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.

[25] 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.

[26] 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.

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

[28] 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.

[29] 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.

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

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

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

[33] 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.

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

[35] 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.

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

[37] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey E. Hinton. Restricted boltzmann machines for collaborative filtering. In Ghahramani [12], pages 791–798.

[38] 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.

[39] 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.

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

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

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

[43] 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.

[44] 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.