Improving One-Class Collaborative Filtering via Ranking-Based Implicit Regularizer

Jin Chen,[1] **Defu Lian,**[*2,3] **Kai Zheng**[1]
[1]School of Computer Science and Engineering, University of Electronic Science and Technology of China
[2]School of Computer Science and Technology, University of Science and Technology of China
[3]School of Data Science, University of Science and Technology of China
{chenjin.uestc,dove.ustc}@gmail.com, zhengkai@uestc.edu.cn

Abstract

One-class collaborative filtering (OCCF) problems are vital in many applications of recommender systems, such as news and music recommendation, but suffers from sparsity issues and lacks negative examples. To address this problem, the state-of-the-arts assigned smaller weights to unobserved samples and performed low-rank approximation. However, the ground-truth ratings of unobserved samples are usually set to zero but ill-defined. In this paper, we propose a ranking-based implicit regularizer and provide a new general framework for OCCF, to avert the ground-truth ratings of unobserved samples. We then exploit it to regularize a ranking-based loss function and design efficient optimization algorithms to learn model parameters. Finally, we evaluate them on three real-world datasets. The results show that the proposed regularizer significantly improves ranking-based algorithms and that the proposed framework outperforms the state-of-the-art OCCF algorithms.

Introduction

Recommender system has become more and more popular to satisfy the user and increase the revenue for providers. With the development of digital media, the form of information is becoming various and lots of research has shifted attention from explicit feedback to implicit feedback. Implicit feedback is everywhere, such as click, purchase or browsing history which covers the user's interest and preference. Recommender system aims to infer from the abundant implicit feedback to provide the user a rank list of the items. One-class collaborative filtering (OCCF) problem refers to a strategy which relies on the implicit feedback and OCCF problem is common and vital in many application scenarios.

However, learning from implicit feedback is expensive. There is some hidden information behind the ratings both for observed ratings and unobserved ratings. As a result, recommender system should consider the interaction between each user-item pair and the unobserved data, which is time consuming. Moreover, it is difficult to minimize the objective function in linear time due to the sparsity of the data. Some methods sacrifice the accuracy by sampling for efficiency (Rendle et al. 2009).

The absence of negative feedback (Hu, Koren, and Volinsky 2008) is another problem for the case of implicit feedback. In unobserved data, there are dislike items and the items not exposed but potentially preferred. In most cases, the unobserved data is treated as missing data. Some other interpretations are put on the missing data (Devooght, Kourtellis, and Mantrach 2015; Steck 2010). The state-of-the-art algorithm(Devooght, Kourtellis, and Mantrach 2015) sets a prior on the unobserved data, resulting to a smaller weight of unobserved data. The WRMF method sets the approximate rating 0 for the missing data but in the real word the ratings are not single.

In this paper, we focus on the unobserved data depending on the real-world data. There is no doubt that the items with 1 ratings are more likely to be preferred than the items with 0 ratings. However, the difference between the unobserved data is hard to formalize to help rank the items. In most cases, the interaction between the unobserved data is omitted. The 0 ratings are not unique and the unobserved items contains both dislike items and potentially preferred items. There are certain similarities between the unobserved data. As a result, we aim to catch the link between these data to improve the performance of ranking. We propose a ranking-based implicit regularizer which penalizes the large difference between the unobserved data. In this way, we avert the estimation for unobserved data of ground-truth dataset and we can deal with the origin data without approximation.

Besides proposing the regularizer, we design a ranking-based framework for one-class collaborative filtering problems based on matrix factorization. The framework captures the comparison for two cases: the comparison between the observed and the unobserved data, and the inner comparison between the unobserved data which corresponds with the regular form of objective function.

Our main contributions are:

- We propose a ranking-based regularizer to penalize the discrepancy of estimated preferences for different unobserved items, and thus avert the use of ill-defined ground-truth ratings of unobserved items.
- We propose a general ranking-based framework for one class collaborative filtering, and design a highly effective optimization algorithm to dramatically improve scalability of the framework .

*Corresponding author

Our paper is organized as follows: First, we briefly introduce the recommendation problem for one-class collaborative filtering and the notations we referred. Second, we introduce our proposed ranking-based implicit regularizer with its simplification forms and provide the general framework. Next, we apply square loss function and introduce the optimization of our method. At last, we experiment our method on three datasets with large number and compare it with other matrix factorization methods.

Related Work

Matrix factorization (MF) model is regarded as the most effective recommendation system model (Koren and Bell 2015; Koren 2009). For OCCF problem, there are two widely used methods to rank the items for the user. The sampling method, such as BPR (Rendle et al. 2009) (Pan et al. 2008), has an efficient updating strategy based on the negative sampling. BPR using a stochastic gradient descent (SGD) to make comparisons on the pairs of the positive and negative samples. However, negative sampling for efficiency leaves out some information in the ranking-based system. Another method is based on the entire data, such as coordinate decent (CD), SGD (Shi et al. 2012; Ning and Karypis 2011; Sedhain et al. 2016)and etc. These methods take the whole data into consideration but suffer from the inefficiency due to the huge number of unobserved data. For our work, we try to utilize the full data and design efficient algorithm.

Several methods have been proposed to optimize the MF model. Such as coordinate descent (CD) (Bayer et al. 2017; Hu, Koren, and Volinsky 2008) and stochastic gradient descent (SGD). The method of alternating least square (ALS) (Hu, Koren, and Volinsky 2008; Takács and Tikk 2012; Pilászy, Zibriczky, and Tikk 2010) is an instance of coordinate descent. Pan's work(Pan and Scholz 2009) introduces an approximation method for ALS to solve the problem of inefficiency. Furthermore, Tikk's work(Takács and Tikk 2012) describes another method for ALS by simplifying the equations. In our work, we adopt this method to optimize the objective function without sampling, which utilize all the data.

Regardless of the optimization for matrix factorization, the treating on unobserved data is another important problem in one-class collaborative filtering problem. The unobserved data contains much noise(Hu, Koren, and Volinsky 2008) and the interpretation on the unobserved data varies (Hu, Koren, and Volinsky 2008; Pan et al. 2008; Pan and Scholz 2009). The state-of-the-art algorithm WRMF (Devooght, Kourtellis, and Mantrach 2015) sets small weights on the unobserved data depending on the *not missing at random* assumption for the ranking based recommendation model and apply it into an online model because of the low time complexity. However, the estimate on the unobserved data means a deviation over the real data and the suggested value 0 leaves out some significant information for ranking. In our work, we consider the comparison between the unobserved data in a new way. And we exploit it to regularize the ranking-based loss function to satisfy a recommendation model.

Table 1: Notations

Notation	Representations
E_u	$E_u = \{i \in E \vert r_{ui} > 0\}$
U_i	$U_i = \{u \in U \vert r_{ui} > 0\}$
$\mathbf{p}_u$	the latent vector for the u-th user
$\mathbf{q}_i$	the latent vector for the i-th item
$\mathbf{Q}_u$	$\mathbf{Q}_{[E_u]}$, the submatrix of $\mathbf{Q}$
$\mathbf{P}_l$	$\mathbf{P}_{[U_l]}$, the submatrix of $\mathbf{P}$
$\tilde{\mathbf{q}}$	$\sum_i \mathbf{q}_i$, the sum of the item latent vectors
$\tilde{\mathbf{q}}_u$	$\sum_{i \in E_u} \mathbf{q}_i$
r_{ui}	the real rating for user u and item i
$\hat{r}_{ui}$	$\mathbf{p}_u^T \mathbf{q}_i$, the estimate rating
N_u	the number of the items the u-th user rated
$\vec{\mathbf{N}}_u$	the vector of N_u
$\tilde{\mathbf{Q}}$	each row of $\tilde{\mathbf{Q}}$ is $\tilde{\mathbf{q}}_u$
$\vert R \vert$	the number of the observed ratings($r_{ui} = 1$)
T	the number of iterations

Preliminaries

A recommendation model aims to a assign score to each user-item pair to provide the user with a ranking list of items. The rating r_{ui} denotes the rating of the u-th user on the i-th item. The set $U = \{u_1, u_2, ..., u_M\}$ is the set of all users and $E = \{e_1, e_2, ..., e_N\}$ of all items. The set $E_u = \{i \in E \vert r_{ui} > 0\}$ denotes the items the u-th user rated. Similarly the set $U_i = \{u \in U \vert r_{ui} > 0\}$ denotes the users who rated the i-th item. The matrix $\mathbf{R}$ denotes the rating matrix. For a one-class collaborative filtering problem, $r_{ui} \in \{0, 1\}$ where r_{ui} is an element of $\mathbf{R}$.

Matrix factorization methods produce a latent feature vector of k dimensions for each user and each item. Vector $\mathbf{p}_u$ denotes the latent feature vector for the u-th user and vector $\mathbf{q}_i$ for i-th item. Matrices $\mathbf{P} \in \mathbb{R}^{M \times K}$ and $\mathbf{Q} \in \mathbb{R}^{N \times K}$ denote the latent factor matrix for users and items. The predicted rating $\hat{r}_{ui}$ is the inner product of $\mathbf{p}_u$ and $\mathbf{q}_i$. Matrix factorization is described as an optimization problem, whose regular form is:

$$\min_{\mathbf{P},\mathbf{Q}} L(r_{ui}, \hat{r}_{ui}) + \lambda R(\mathbf{P}, \mathbf{Q})$$

The objective function has two parts: the loss function and the regularizer. The loss function L measures the error the model makes on the observed data. The loss function can be square loss, logit loss, hinge loss, etc. R is the regularization term. λ is the parameter of the regularizer. For the sake of convenience and simplicity, we summarize the notations referred in Table(1).

Learning From Implicit Feedback

Depending on the matrix factorization model, we introduce our proposed implicit regularizer and a general framework for one-class collaborative filtering. Then we exploit it to the square loss function and design the optimization algorithm.

Ranking-based Implicit Regularizer

For the one-class collaborative filtering (OCCF) problem, the lack of the negative samples prevents the delicate classification for the items. In the implicit feedback data set, the ratings are only marked as 0 and 1. However, the items with the rating 0 can be either disliked or not exposed but potentially preferred.

According to Robin Devooght's work(Devooght, Kourtellis, and Mantrach 2015), a prior is set on the unknown values. The form of the regularizer on the unobserved ratings is:

$$R_0(\mathbf{P}, \mathbf{Q}) = \sum_u \sum_{i \notin E_u} (r_0 - \hat{r}_{ui})^2$$

where $\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i$. A rating $r_0 = 0$ is suggested as a prior estimate for the unobserved data. The regularizer becomes:

$$R_0(\mathbf{P}, \mathbf{Q}) = \sum_u \sum_{i \notin E_u} (\hat{r}_{ui})^2 = \sum_u \sum_{i \notin E_u} (\mathbf{p}_u^T \mathbf{q}_i)^2 \quad (1)$$

Although the method has a good performance in matrix factorization models and a prior estimate is set on the unobserved samples, the value of the prior is hard to explain and lack of verification. The suggested value 0 for the prior tends to treat all the unobserved data as negative samples which leads to a deviation from common awareness. In addition, it is hard to estimate a unique rating for the unobserved items.

According to Steffen Rendle's recent work in WWW 2017 (Bayer et al. 2017), an implicit regularizer is actually introduced in WRMF, to penalize the deviation of each user's estimated preference for all unobserved item from zero. This means each user's estimated preference for unobserved items should be close to zero, implying some "similarity" among unobserved items. However, the zero-ratings for unobserved items should be uncertain. From this perspective, we propose a ranking based regularizer defined as:

$$\begin{aligned} R_r &= \frac{1}{2} \sum_u \sum_{i<j \notin E_u} (\hat{r}_{ui} - \hat{r}_{uj})^2 \\ &= \frac{1}{4} \sum_u \sum_{i,j \notin E_u} (\mathbf{p}_u^T \mathbf{q}_i - \mathbf{p}_u^T \mathbf{q}_j)^2 \end{aligned} \quad (2)$$

From the formula, we can measure the difference between the unobserved data and intend to smooth it. The regularizer averts the usage of the rating zeros of unobserved items but utilizes the ground-truth data.

Compared with the regularizer Eq(1), the computation of our regularizer is more complicated. We expand Eq(2) into three parts according to the principle of inclusion and exclusion and we obtain a form of R_r as:

$$R_r = \frac{1}{4}(R_0 - 2R_1 + R_2)$$

where

$$\begin{aligned} R_0 &= \sum_u \sum_{i,j} (\mathbf{p}_u^T \mathbf{q}_i - \mathbf{p}_u^T \mathbf{q}_j)^2 \\ R_1 &= \sum_u \sum_{i \in E_u} \sum_j (\mathbf{p}_u^T \mathbf{q}_i - \mathbf{p}_u^T \mathbf{q}_j)^2 \\ R_2 &= \sum_u \sum_{i,j \in E_u} (\mathbf{p}_u^T \mathbf{q}_i - \mathbf{p}_u^T \mathbf{q}_j)^2 \end{aligned}$$

The regularizer includes the computation of set E and the subset E_u for each user. To some extent, the avoidance of computing of non-set improves the efficiency of computation and solves the some problems caused by data sparsity. Besides, the derivative of R_i is similar. Let us briefly derive R_1 with respect to $\mathbf{q}_l$ as:

$$\begin{aligned} \frac{\partial R_1}{\partial \mathbf{q}_l} &= 2 \sum_{u \in U_l} \sum_{j \neq l} \mathbf{p}_u \mathbf{p}_u^T (\mathbf{q}_l - \mathbf{q}_j) + 2 \sum_{i \neq l} \sum_{u \in U_i} \mathbf{p}_u \mathbf{p}_u^T (\mathbf{q}_l - \mathbf{q}_i) \\ &= 2 \sum_{u \in U_l} \mathbf{p}_u \mathbf{p}_u^T (N \mathbf{q}_l - \tilde{\mathbf{q}}) + 2 \sum_u \mathbf{p}_u \mathbf{p}_u^T (N_u \mathbf{q}_l - \tilde{\mathbf{q}}_u) \end{aligned}$$

where $\tilde{\mathbf{q}} = \sum_i \mathbf{q}_i$ and $\tilde{\mathbf{q}}_u = \sum_{j \in E_u} \mathbf{q}_j$ which is the sum over the item latent vector corresponding to the u-th user. N_u is the number of the items the u-th user rated.

Similarly, we derive R_1 with respect to $\mathbf{p}_u$ as:

$$\begin{aligned} \frac{\partial R_1}{\partial \mathbf{p}_u} &= 2 \sum_{i \in E_u} \sum_j (\mathbf{q}_i - \mathbf{q}_j)(\mathbf{q}_i - \mathbf{q}_j)^T \mathbf{p}_u \\ &= \left(2N \mathbf{Q}_u^T \mathbf{Q}_u + 2N_u \mathbf{Q}^T \mathbf{Q} - 2\tilde{\mathbf{q}} \tilde{\mathbf{q}}_u^T - 2\tilde{\mathbf{q}}_u \tilde{\mathbf{q}}^T\right) \mathbf{p}_u \end{aligned}$$

Matrix $\mathbf{Q}_u \in \mathbb{R}^{N_u \times K}$ is the submatrix of $\mathbf{Q}$ composed of the latent vectors of the items which the u-th user rated. Extracting $\mathbf{Q}_u$ from $\mathbf{Q}$ can be quickly operated. We transform the accumulation into the matrix multiplication to avert the time-consuming loops. Finally, the derivative of $\mathbf{q}_l$ is as:

$$\frac{\partial R_r}{\partial \mathbf{q}_l} = \sum_{u \notin U_l} (N - N_u) \mathbf{p}_u \mathbf{p}_u^T \mathbf{q}_l - \sum_{u \notin U_l} \mathbf{p}_u \mathbf{p}_u^T (\tilde{\mathbf{q}} - \tilde{\mathbf{q}}_u) \quad (3)$$

Similarly, the derivative of $\mathbf{p}_u$ is as:

$$\begin{aligned} \frac{\partial R_r}{\partial \mathbf{p}_u} =& (N - N_u)(\mathbf{Q}\mathbf{Q}^T - \mathbf{Q}_u \mathbf{Q}_u^T) \mathbf{p}_u \\ & - (\tilde{\mathbf{q}} - \tilde{\mathbf{q}}_u)(\tilde{\mathbf{q}} - \tilde{\mathbf{q}}_u)^T \mathbf{p}_u \end{aligned} \quad (4)$$

Form Eq.(3) and Eq.(4), the hessian matrix can be obtained both for the user and the item which contributes to efficient update depending on Newton's method. Moreover, there's no external parameter or estimate in the regularizer but the computation based on the actual data. We can change the value of λ to control the influence of the ranking based regularizer.

A General Framework for One-Class Collaborative Filtering

The objective function of a matrix factorization model of ranking based system can be defined as:

$$\hat{y} = \sum_u \sum_{i \notin E_u} \sum_j L((r_{ui} - r_{uj}) - (\hat{r}_{ui} - \hat{r}_{uj})) \quad (5)$$

This function was introduced by Jahrer (Jahrer and Töscher 2011) in the recommender system literature. The methods BPR (Rendle et al. 2009), WRMF(Hu, Koren, and Volinsky 2008) also provide similar frameworks for ranking-based system. There are two circumstances in this objective function: (1)The comparison between the observed and unobserved data which implies that the items with 1 ratings are preferred than zero rating items. (2)The comparison between

the unobserved data which corresponds with our proposed regularizer. We integrate our ranking-based regularizer into the function and obtain the objective function as:

$$\hat{y} = \sum_{u} \sum_{i \in E_u} \sum_{j \notin E_u} L(1, \hat{r}_{uij}) + \lambda R_r \tag{6}$$

where $\hat{r}_{uij} = \hat{r}_{ui} - \hat{r}_{uj}$, and L is the loss function, such as square loss $L(y,x) = (y-x)^2$, logit loss $L(y,x) = \log(1+e^{-yx})$, or hinge loss $L(y,x) = \max(0, y(1-x))$. λ is the coefficient of the regularizer. For the one-class collaborative filtering system, our objective function is consistent with Eq.(5).

Applications and Optimizations

We apply square loss function as the loss function in our framework. Square loss function is widely used in ranking-based system. Square loss is differentiable which helps efficient calculation during the optimization. We can obtain a simple form depending on the alternating least square method. Furthermore, analytic solution can be applied under the square loss for optimization. However, a new difficulty raised is the increase of time complexity due to the increasing number of comparison between unobserved items for each user. To improve efficiency, we design optimization algorithms to learn model parameters.

By considering L as the squared loss function, the objective function becomes:

$$\begin{aligned}\hat{y}_s =& \frac{1}{2} \sum_{u} \sum_{i \in E_u} \sum_{j \notin E_u} (1 - (\hat{r}_{ui} - \hat{r}_{uj}))^2 \\ &+ \lambda \sum_{u} \sum_{i,j \notin E_u} (\hat{r}_{ui} - \hat{r}_{uj})^2\end{aligned} \tag{7}$$

The objective function contains $M \times N \times N$ terms. In addition to the large numbers of the dataset, the sparsity of the data set prevents the quick calculation. For the sake of efficiency, we will simplify the loss function in a similar way of simplifying the regularizer. The derivative of loss function with respect to user latent vector $\mathbf{p}_u$ is as:

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{p}_u} =& \frac{\partial L_0}{\partial \mathbf{p}_u} - \frac{\partial L_1}{\partial \mathbf{p}_u} \\ =& - \sum_{i \in E_u} \sum_{j} (1 - (\mathbf{p}_u^T \mathbf{q}_i - \mathbf{p}_u^T \mathbf{q}_j))(\mathbf{q}_i - \mathbf{q}_j) \\ &+ \sum_{i \in E_u} \sum_{j \in E_u} (1 - (\mathbf{p}_u^T \mathbf{q}_i - \mathbf{p}_u^T \mathbf{q}_j))(\mathbf{q}_i - \mathbf{q}_j)\end{aligned}$$

where

$$L_0 = \frac{1}{2} \sum_{u} \sum_{i \in E_u} \sum_{j} (1 - (\hat{r}_{ui} - \hat{r}_{uj}))^2$$

$$L_1 = \frac{1}{2} \sum_{u} \sum_{i \in E_u} \sum_{j \in E_u} (1 - (\hat{r}_{ui} - \hat{r}_{uj}))^2$$

We expand the loss function into two parts to avert the computation over the unobserved data which consumes an enormous amount of time due to the huge sparse matrix. After simplification, we obtain the derivative of the loss function as:

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{p}_u} =& N_u (\mathbf{Q}^T \mathbf{Q} - \mathbf{Q}_u^T \mathbf{Q}_u) \mathbf{p}_u - \tilde{\mathbf{q}}_u (\tilde{\mathbf{q}} - \tilde{\mathbf{q}}_u)^T \mathbf{p}_u \\ &+ (N - N_u) \mathbf{Q}_u^T \mathbf{Q}_u \mathbf{p}_u - (\tilde{\mathbf{q}} - \tilde{\mathbf{q}}_u) \tilde{\mathbf{q}}_u^T \mathbf{p}_u \\ &- (N \tilde{\mathbf{q}}_u - N_u \tilde{\mathbf{q}})\end{aligned} \tag{8}$$

$\mathbf{Q}^T \mathbf{Q}$ and $\tilde{\mathbf{q}}$ can be computed before the loop of updating each user to save time.

The derivative of item latent vector $\mathbf{q}_l$ is as:

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{q}_l} =& (\sum_{u \notin U_l} N_u \mathbf{p}_u \mathbf{p}_u^T + \sum_{u \in U_l} (N - N_u) \mathbf{p}_u \mathbf{p}_u^T) \mathbf{q}_l \\ &- \sum_{u \notin U_l} \mathbf{p}_u \mathbf{p}_u^T \tilde{\mathbf{q}}_u - \sum_{u \in U_l} \mathbf{p}_u \mathbf{p}_u^T (\tilde{\mathbf{q}} - \tilde{\mathbf{q}}_u) \\ &+ \sum_{u} N_u \mathbf{p}_u - \sum_{u \in U_l} N \mathbf{p}_u \\ =& (\sum_{u} N_u \mathbf{p}_u \mathbf{p}_u^T + \sum_{u \in U_l} (N - 2N_u) \mathbf{p}_u \mathbf{p}_u^T) \mathbf{q}_l \\ &- \sum_{u} \mathbf{p}_u \mathbf{p}_u^T \tilde{\mathbf{q}}_u - \sum_{u \in U_l} \mathbf{p}_u \mathbf{p}_u^T (\tilde{\mathbf{q}} - 2\tilde{\mathbf{q}}_u) \\ &+ \sum_{u} N_u \mathbf{p}_u - \sum_{u \in U_l} N \mathbf{p}_u\end{aligned} \tag{9}$$

The form of the derivative for the item latent vector is more complicated. To speed up the update, apart from restoring some variables, we cache some values and update a part of them after each updating.

We denote $\sum_u N_u \mathbf{p}_u \mathbf{p}_u^T$ as S_l, $\sum_u N_u \mathbf{p}_u$ as N_p and these variables can be calculated before the loops of updating the latent vector. To compute $P_q = \sum_u \mathbf{p}_u \mathbf{p}_u^T \tilde{\mathbf{q}}_u$ and update P_q, we first calculate the sum of the ratings over the related items $\sum_{i \in E_u} \hat{r}_{ui}$ for each user and restore these sums into matrix $\tilde{\mathbf{Q}}$. We define an operation $\mathrm{tr}(A, B)$ as computing the sum of each row of the matrix AB where AB is the dot product of A and B. After computing $\mathbf{P}^T tr(\mathbf{P}, \tilde{\mathbf{Q}})$, We obtain the value of P_q. After each update for the item latent vector, we compute the values of $\sum_{i \in E_u} \hat{r}_{ui}$ of the related users and change the related multiplier while keeping other fixed. As there are similar components in Eq(3) , we can integrate the two parts.

Setting $\frac{\partial L}{\partial \mathbf{p}_u} + \lambda \frac{\partial R_r}{\partial \mathbf{p}_u} = 0$ and $\frac{\partial L}{\partial \mathbf{q}_l} + \lambda \frac{\partial R_r}{\partial \mathbf{q}_l} = 0$, we can update the user latent vector $\mathbf{p}_u$ and item latent vector $\mathbf{q}_l$ with the form of $x = A^{-1} b$. The Algorithm(1) shows the procedure of the optimization with ALS. After the optimization, we learn the parameter of the latent matrix $\mathbf{P} \in \mathbb{R}^{M \times K}$ and $\mathbf{Q} \in \mathbb{R}^{N \times K}$ which can predict the ratings for each user-item pair.

TIME COMPLEXITY

From the formula(7), the trivial method takes $O(MN^2K^2 + (M+N)K^3)$ time in each iteration. In one-class collaborative filter problem, the dimension of the latent vector K

is much smaller than the number of the users M and the number of the items N. And the number of the observed ratings $|R|$ is also smaller than the whole number of the ratings. Taking some methods, our learning algorithm takes only $O(|R|K^2 + (M+N)K^3)$. The calculations of $\mathbf{Q}_u^T\mathbf{Q}_u$ and the inversion of matrix occupy most time.

Algorithm 1: Square Loss for Ranking based implicit Regularizer

Input: The rating matrix $\mathbf{R} \in \mathbb{R}^{M\times N}$
Output: Latent matrix $\mathbf{P} \in \mathbb{R}^{M\times K}$, $\mathbf{Q} \in \mathbb{R}^{N\times K}$
Randomly initialize $\mathbf{P}$, $\mathbf{Q}$;
for $t \leftarrow 1, 2, ..., T$ **do**
  // the update of $\mathbf{P}$;
  $\mathrm{QtQ} \leftarrow \mathbf{Q}^T\mathbf{Q}$ $\tilde{\mathbf{q}} \leftarrow \mathbf{Q}^T 1$;
  for $u = 1 : M$ **do**
    Calculate A^{-1} and b based on Eq(4)(8) ;
    $\mathbf{p}_u \leftarrow A^{-1}b$;
  end
  // the update of $\mathbf{Q}$;
  $\tilde{\mathbf{q}} \leftarrow \mathbf{Q}^T 1$;
  for $u = 1 : M$ **do**
    $\tilde{\mathbf{Q}}_{[u]} \leftarrow \mathbf{Q}^T_{[E_u]} 1$;
  end
  $r_- \leftarrow tr(\mathbf{P}, \tilde{\mathbf{Q}})$;
  for $l = 1 : N$ **do**
    Calculate A^{-1} and b based on Eq(3)(9);
    $\mathbf{q}_l \leftarrow A^{-1}b$;
    // update the relevant variable;
    $\tilde{\mathbf{q}} \leftarrow \tilde{\mathbf{q}} - \mathbf{q} + \mathbf{q}_l$;
    $\tilde{\mathbf{Q}}'_{[U_l]} \leftarrow \tilde{\mathbf{Q}}_{[U_l]} - \mathbf{q} + \mathbf{q}_l$;
    $r'_- \leftarrow r_- + tr(\mathbf{P}_{[U_l]}, \tilde{\mathbf{Q}}'_{[U_l]} - \tilde{\mathbf{Q}}_{[U_l]})$;
    $S'_l \leftarrow S_l + \mathbf{P}^T_{[U_l]}(r'_- - r_-)$;
  end
end
return $\mathbf{P}$, $\mathbf{Q}$

To sum up, our method provides a simpler and faster way depending on the alternating least square (ALS) method to minimize the objective function and design effective strategy to accelerate the update.

Experiments

In this part, we perform several experiments to demonstrate the following key points:

- The general framework based on the square loss function has a better performance than the state-of-the-art algorithm.
- Our proposed method still has a better performance as the dimension of the latent vector increases.
- Using the ranking-based implicit regularizer leads to significant improvement of ranking.

Table 2: Statistics of Datasets

Datasets	Ratings	Users	Items	Sparsity
MovieLens	9,983,739	69,838	8,939	83.669%
Amazon	4,701,968	158,650	128,939	99.978%
Netflix	100,396,329	463,770	17,764	98.781%

Experimental Setup

Datasets and Preprocessing We perform experiments on three real-world datasets: MovieLens, AmazonMovies and Netflix. We convert the datasets into implicit data, where each entry is marked as 0/1 indicating whether the user reviewed the item. Table (2) summarizes the characteristics of the data.

- **MovieLens**: The MovieLens dataset is the famous movie ratings dataset. We use the dataset from of MovieLens10M dataset, including 10,000,054 ratings from 71,567 users for 10,681 items. We filter out users and items with less than 20 ratings.
- **Amazon**: This is a larger collection of ratings for Amazon books, including 8,898,041 ratings. This datasets is sparser than MovieLens. The number of items is larger than the other datasets. We filter out users and items with less than 10 ratings.
- **Netflix**: The Netflix dataset contains 463,770 users and 17,764 items, which is larger than MovieLens and Amazon dataset. We filter out the users and items with less than 10 rating interactions.

Baseline We compare our proposed method with the following methods:

- **WRMF**(Hu, Koren, and Volinsky 2008) :WRMF sets a prior on unobserved values. It learns parameters by alternating least square method in the case of square loss. We choose the square loss as the loss function and experiment on static datasets. We tuned the parameter ρ of the regularizer on the unobserved data.
- **BPR**(Rendle et al. 2009):BPR optimizes the pair-wise ranking loss between positive samples and negative samples based on bootstrap sampling. It learns parameters by Stochastic Gradient Descent (SGD). We tuned the parameter of regularizer β_u, β_i for users and items and the learning rate ϵ.
- **SQL_RANK**(Wu, Hsieh, and Sharpnack 2018): SQL_RANK is a listwise approach in ranking-based system. It accommodates ties and missing data and can run in linear time. It cast listwise collaborative ranking as maximum likelihood under a permutation model which applies probability mass to permutations. We keep the parameter λ of the regularizer fixed and tune the initial learning rate ϵ.
- **WARP** (Kula 2015): WARP treats user's rating as a binary value and fits the ratings on the observed and unobserved items with different confidential values.

Table 3: The Tuning Parameters Sets

Method	Key	Tested Values
SRRMF	K	8, 16, 32, 64, 128
	λ	1e-5, 1e-4, 0.001, 0.01, 0.1, 0.5
WRMF	ρ	1e-4, 1e-3, 0.01, 0.1, 1.0
BPR	β_u	0.001, 0.01, 0.05, 0.1, 0.5, 1.0
	β_i	0.001, 0.01, 0.05, 0.1, 0.5, 1
	ϵ	0.001, 0.01, 0.1, 0.5
SQL_RANK	ϵ	0.2,0.1

- **HPF**(Gopalan, Hofman, and Blei 2013): HPF is a hierarchical poisson matrix factorization for recommendation system. It handles explicit data and implicit data. We tune the parameter of the number in the hierarchical system.

Evaluation Metrics We use two standard metrics of ranking evaluation: Normalized Discounted Cumulative Gain (NDCG) and recall.

NDCG is widely used in the recommendation system. It considers both the prediction of the rankings and the position of ratings. We return the top-200 items for all users and compute the average NDCG as our final metric.

The metric recall is another evaluating indicator we used. It shows the performance of the correct prediction over the observed samples in the testing set. We also return the top-200 items for all users.

Parameter Tuning Table (3) shows the parameters for different methods during tuning. After 10 iterations of training, we choose the parameter depending on the best performance of NDCG for our method.

Results and Analysis

(1) The general framework. We do cross validation experiment of 5 times on each dataset for our method and baselines. Table (4) shows the comparison among the algorithms. Our square loss with ranking based regularizer matrix factorization (SRRMF)[1] shows a better performance over all baseline methods. We can conclude that averting the zero-ratings for the unobserved items contributes to the ranking of the items compared with WRMF. As for BPR, the negative sample based method, our algorithm shows a significant improvement probably due to the utilization of full data. However, for the SQL_rank method, it takes us more than two days to run on the dataset MovieLens10M and Netflix in one iteration and so we don't report the result for the two datasets. Furthermore, our method SRRMF has an edge over the stability because of the avoidance of negative sampling and the linear form of the gradient for both the user and the item latent vector.

As a result, we choose the methods which have superior performances in Table (4) to figure out the change over

[1]https://github.com/HERECJ/recsys/tree/master/alg/discrete/SRRMF

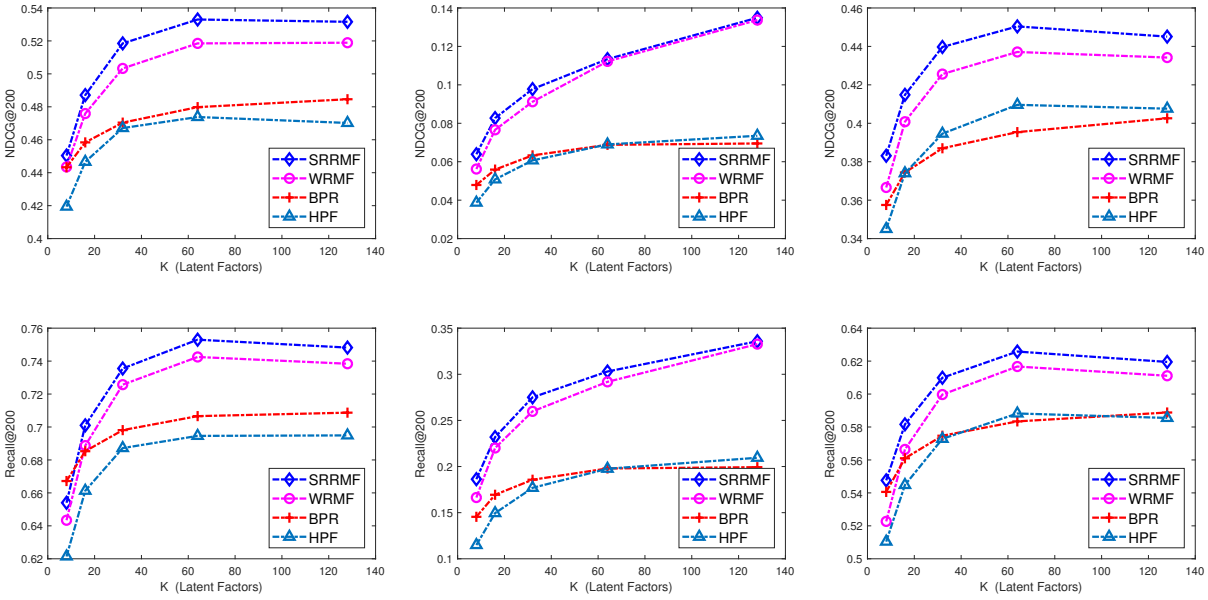
Table 4: Comparison of our general framework. The dimension of the latent factor K is 32.

	MovieLens NDCG@200	MovieLens Recall
SRRMF	0.5177 ± 0.0003	0.7165 ± 0.0006
WRMF	0.4974 ± 0.0005	0.6915 ± 0.0007
BPR	0.4710 ± 0.0009	0.6991 ± 0.0003
WARP	0.2809 ± 0.0015	0.4613 ± 0.0007
HPF	0.4673 ± 0.0015	0.6885 ± 0.0017
SQL_rank	/	/
	Amazon NDCG@200	**Amazon** Recall
SRRMF	0.0977 ± 0.0003	0.2748 ± 0.0006
WRMF	0.0914 ± 0.0002	0.2452 ± 0.0006
BPR	0.0634 ± 0.0009	0.1859 ± 0.0007
WARP	0.0002 ± 0.0000	0.0008 ± 0.0001
HPF	0.0609 ± 0.0007	0.1768 ± 0.0022
SQL_rank	0.0004 ± 0.0000	0.0014 ± 0.0000
	Netflix NDCG@200	**Netflix** Recall
SRRMF	0.4397 ± 0.0001	0.5991 ± 0.0003
WRMF	0.4224 ± 0.0001	0.5765 ± 0.0003
BPR	0.3292 ± 0.0012	0.5179 ± 0.0006
WARP	0.2257 ± 0.0083	0.3794 ± 0.0079
HPF	0.3946 ± 0.0008	0.5728 ± 0.0010
SQL_rank	/	/

the values of K which is the dimension of the latent vector. Fig.(1) shows the comparison between our method and the state-of-the-art algorithms. Our proposed general framework outperforms on the all three datasets. Compared with WRMF, on the MovieLens and Netflix datasets, our method performs better as the dimension increases. The sparsity of the Amazon dataset leads to a slightly worse performance when the dimension of features increases. As the Amazon dataset has the biggest number of items, the best parameter of the regularizer is smaller than the other dataset.

According to the comparisons, our method still has an improvement over the larger value of K, while the performance BPR and HPF have worse performances due to overfitting. That is to say, our method can capture more features to improve the performance for ranking which is particularly important for big dataset such as Netflix. In general, our method takes into account all the data and fully explore the implied information under the unobserved data which leads to a better performance over the state-of-the-art algorithm.

(2) The Influence of the Regularizer. In this section, we analyze the impact of our proposed ranking-based implication regularizer. Setting $\lambda = 0$, we obtain the prediction



(a) (b) (c)

Figure 1: NDCG@200 and Recall@200 on the dataset (a):MovieLens10M (b):Amazon (c):Netflix

Table 5: The Comparison between With/Without Regularizer (K = 64).

DataSets	Regularizer	NDCG@200	Improvement	RECALL@200	Improvement
MovieLens	√	0.5330	21.33%	0.7530	4.08%
	×	0.4393		0.7268	
Amazon	√	0.1134	38.46%	0.2940	19.90%
	×	0.0819		0.2452	
Netflix	√	0.4504	31.73%	0.6067	3.60%
	×	0.3419		0.5829	

ratings without the regularizer after minimizing the objective function. After tuning the parameter of the regularizer, we obtain the final result of optimization. We compared the cases for the best parameter and 0 to witness the effect of regularization. Table (5) shows the change of exploiting our regularizer depending on the square loss function. For the all three data sets, our proposed regularizer leads to huge improvement. Especially for Amazon data set, the regularizer results in the most significant improvement both on NDCG@200 and recall, which is 38.46% and 19.90% respectively. As Amazon dataset is sparser than the other two datasets, the proposed regularizer still shows good performance in spite of the data sparsity. We can infer that the inner interaction between the unobserved samples has something in common and contains plenty of useful information for ranking.

In addition, depending on the same ranking-based loss function, the metric NDCG@200 grows more than recall, which indicates our regularizer brings a more appropriate recommendation ranking for the users. To summarize, our proposed ranking-based implicit regularizer utilizes more information and has a much better performance on the sparser dataset.

Conclusion

In this work, we propose a new ranking-based regularizer which depends on the real-world data. And we apply it into a general framework for recommendation system. According to our experiments on the three datasets with large number, we can conclude that our proposed ranking-based implicit regularizer can efficiently capture the comparison between the unobserved samples with which we can have a better recommendation for the user.

Considering the comparison with other methods, our framework with square loss has a better performance over

the large ground-truth dataset. Furthermore, our framework outperforms the state-of-the-art algorithm.

In the future, we will apply more loss functions such as logit loss, hinge loss, to our framework to solve the nonlinear problems. Secondly, we will normalize the loss and the regularizer to obtain a standard form for computing which can balance the influence of the loss and the regularizer. Next, we will consider the case of explicit feedback to build a more general framework for both implicit feedback and explicit feedback.

Acknowledgments

Defu Lian is supported by the National Natural Science Foundation of China (Grant No. 61502077 and 61631005), Kai Zheng is supported by the National Natural Science Foundation of China (Grant No. 61532018, 61836007 and 61832017).

References

Bayer, I.; He, X.; Kanagal, B.; and Rendle, S. 2017. A generic coordinate descent framework for learning from implicit feedback. In *Proceedings of the 26th International Conference on World Wide Web*, 1341–1350. International World Wide Web Conferences Steering Committee.

Devooght, R.; Kourtellis, N.; and Mantrach, A. 2015. Dynamic matrix factorization with priors on unknown values. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 189–198. ACM.

Gopalan, P.; Hofman, J. M.; and Blei, D. M. 2013. Scalable recommendation with poisson factorization. *arXiv preprint arXiv:1311.1704*.

Hu, Y.; Koren, Y.; and Volinsky, C. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, 263–272. Ieee.

Jahrer, M., and Töscher, A. 2011. Collaborative filtering ensemble. In *Proceedings of the 2011 International Conference on KDD Cup 2011-Volume 18*, 61–74. JMLR. org.

Koren, Y., and Bell, R. 2015. Advances in collaborative filtering. In *Recommender systems handbook*. Springer. 77–118.

Koren, Y. 2009. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 447–456. ACM.

Kula, M. 2015. Metadata embeddings for user and item cold-start recommendations. In Bogers, T., and Koolen, M., eds., *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015.*, volume 1448 of *CEUR Workshop Proceedings*, 14–21. CEUR-WS.org.

Ning, X., and Karypis, G. 2011. Slim: Sparse linear methods for top-n recommender systems. In *2011 11th IEEE International Conference on Data Mining*, 497–506. IEEE.

Pan, R., and Scholz, M. 2009. Mind the gaps: weighting the unknown in large-scale one-class collaborative filtering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 667–676. ACM.

Pan, R.; Zhou, Y.; Cao, B.; Liu, N. N.; Lukose, R.; Scholz, M.; and Yang, Q. 2008. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, 502–511. IEEE.

Pilászy, I.; Zibriczky, D.; and Tikk, D. 2010. Fast als-based matrix factorization for explicit and implicit feedback datasets. In *Proceedings of the fourth ACM conference on Recommender systems*, 71–78. ACM.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, 452–461. AUAI Press.

Sedhain, S.; Menon, A. K.; Sanner, S.; and Braziunas, D. 2016. On the effectiveness of linear models for one-class collaborative filtering. In *AAAI*, 229–235.

Shi, Y.; Karatzoglou, A.; Baltrunas, L.; Larson, M.; Oliver, N.; and Hanjalic, A. 2012. Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the sixth ACM conference on Recommender systems*, 139–146. ACM.

Steck, H. 2010. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 713–722. ACM.

Takács, G., and Tikk, D. 2012. Alternating least squares for personalized ranking. In *Proceedings of the sixth ACM conference on Recommender systems*, 83–90. ACM.

Wu, L.; Hsieh, C.-J.; and Sharpnack, J. 2018. Sql-rank: A listwise approach to collaborative ranking. *arXiv preprint arXiv:1803.00114*.