Representation Learning with Pair-wise Constraints for Collaborative Ranking

Fuzhen Zhuang
Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), ICT, CAS, Beijing
zhuangfz@ics.ict.ac.cn

Dan Luo
Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), ICT, CAS, Beijing
luod@ics.ict.ac.cn

Nicholas Jing Yuan
Microsoft Research Asia (MSRA)
nicholas.yuan@microsoft.com

Xing Xie
Microsoft Research Asia (MSRA)
xing.xie@microsoft.com

Ping Luo
Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), ICT, CAS, Beijing
luop@ict.ac.cn

Qing He
Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), ICT, CAS, Beijing
heq@ics.ict.ac.cn

ABSTRACT

Last decades have witnessed a vast amount of interest and research in recommendation systems. Collaborative filtering, which uses the known preferences of a group of users to make recommendations or predictions of the unknown preferences for other users, is one of the most successful approaches to build recommendation systems. As we know, most previous collaborative filtering approaches explore the matrix factorization techniques to learn latent user feature profiles and item feature profiles. Also many subsequent works are proposed to incorporate users' social network information and items' attributions to further improve recommendation performance under the matrix factorization framework. However, the matrix factorization based methods may not make full use of the rating information, leading to unsatisfying performance. Recently deep learning has been approved to be able to find good representations in nature language processing, image classification, and so on. Along this line, we propose a collaborative ranking framework via *RE*present*A*tion learning with *P*air-wise constraints (REAP for short), in which autoencoder is used to simultaneously learn the latent factors of both users and items and pair-wise ranked loss defined by (user, item) pairs is considered. Extensive experiments are conducted on five data sets to demonstrate the effectiveness of the proposed framework.

Keywords

Collaborative Ranking; Autoencoder; Representation Learning; Pair-wise Constraints.

1. INTRODUCTION

In order to tackle the information overload problem in Web 2.0, people often rely on recommendations to find objects of interest from other people by spoken words, reference letters, video show, travel guides, and so forth. Recommendation systems provide suggestions of items to purchase or examine in a large space of possible options using user preferences, which have become fundamental applications in electronic commerce and information access. Therefore in recent decades, recommender systems have attracted much attention from multiple disciplines, and many techniques have been proposed to build recommendation systems [1, 8].

As one of the successful approaches to build recommendation systems, previous works of collaborative filtering can be grouped into two types, i.e., memory based techniques and model based ones. Memory based methods use the rating data to calculate the similarity or weight between users and items, and make predictions or recommendations according to those calculated similarity values. However, memory based methods usually perform well only when the data is dense, which are unreliable when data are sparse due to the lack of overlapped items. The other type is model based methods, and its representative technique is matrix factorization. Single Value Decomposition (SVD) is a popular factorization model to build recommendation systems, which uses squared loss to construct the optimization problem,

$$(\hat{\boldsymbol{U}}, \hat{\boldsymbol{V}}) = \arg\min_{\boldsymbol{U},\boldsymbol{V}} ||\boldsymbol{I} \circ (\boldsymbol{M} - \boldsymbol{U}\boldsymbol{V}^\top)|| + \lambda(||\boldsymbol{U}||^2 + ||\boldsymbol{V}||^2), \quad (1)$$

where symbol $\circ$ denotes the element-wise product of two matrices, and $\boldsymbol{M} \in \mathbb{R}^{m \times n}$ is the rating score matrix with m users and n items. $\boldsymbol{U} \in \mathbb{R}^{m \times k}$ and $\boldsymbol{V} \in \mathbb{R}^{n \times k}$ ($k \ll \min(m, n)$ is the number of latent dimensions.) are respectively the latent factors of users and items, and $\boldsymbol{I}$ is the indicator function, i.e., $\boldsymbol{I}_{i,j} = 1$ if user i gave a rating score to item j, else $\boldsymbol{I}_{i,j} = 0$. After solving the optimization problem in Eq.(1), we can obtain the latent factors and make predictions according to the approximated matrix $\hat{\boldsymbol{M}} = \hat{\boldsymbol{U}}\hat{\boldsymbol{V}}^\top$. Actually, we think SVD might not make full use of the rating data, which leads to unsatisfying performance. Therefore, many previous works are proposed to improve the performance of SVD by incorporating external information, e.g., users' social network and item attributions [13, 14, 2, 26].

In recent years, deep learning techniques have been approved to be able to learn good representations in natural language pro-

ACM SIGKDD '16 San Francisco, California, USA

DOI: 10.475/123_4

cessing, computer vision, image processing and speech recognition. There have already some effort on devoting deep learning techniques to recommendation systems. Salakhutdinov et al. [19] proposed Restricted Boltzmann Machine (RBM) for collaborative filtering, which can perform better than the score of Netix's own system. To incorporate external information, Wang et al. [23] proposed a hierarchical Bayesian model called collaborative deep learning (CDL), which jointly performed deep representation learning for the content information and collaborative filtering for the rating matrices.

As we know, there is little work that adopts deep learning techniques on the sole source of rating matrix. In this work, we try to adopt autoencoder to find well representations for building recommendation systems only on rating data. Given the input $\boldsymbol{x}$, autoencoder aims to find the latent representation $\boldsymbol{\xi}$ by minimizing the reconstruction error $||\boldsymbol{x} - \hat{\boldsymbol{x}}||$. Furthermore, to make full use of the rating data, we incorporate pair-wise ranked loss defined by (user, item) pairs into the representation learning framework. To intuitively understand our idea, the proposed framework is shown in Figure 1. From this figure, the latent representations of users and items are simultaneously learned by the same autoencoder, which are used to restructure the approximated matrix $\hat{\boldsymbol{M}}$. Also the ranked losses are considered between the rating matrix $\boldsymbol{M}$ and the approximated matrix $\hat{\boldsymbol{M}}$, which enforces information preservation in $\hat{\boldsymbol{M}}$.

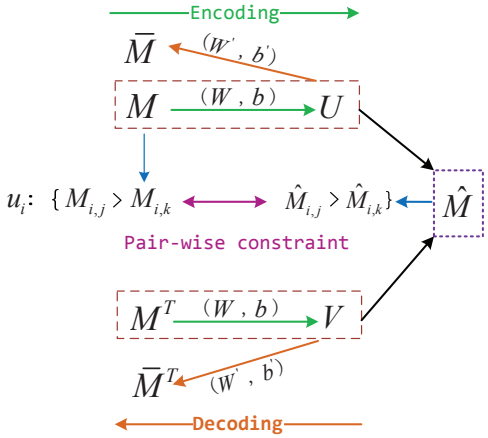


Figure 1: Representation Learning

Our contributions in this paper can be summarized as follows,

- Different from previous works based on matrix factorization technique, we try to use autoencoder to simultaneously learn the latent user factors and item factors.
- To further more full use the rating data, the pair-wise ranked loss defined by (user, item) pair is incorporated in our framework.
- We conduct systemic experiments on five data sets to show the effectiveness of the proposed model. REAP can not only achieve better performance than all the baselines, but also obtain high performance even when the observed training data is sparse.

The rest of this paper is organized as follows. Section 2 briefly introduces the related work, and preliminary knowledge is given in Section 3. Section 4 details the problem formulation and solution derivation of REAP. The experimental results are shown in Section 5. Finally, Section 6 concludes this paper.

2. RELATED WORK

In this section, we will survey some related work to our model, including the matrix factorization based approaches to build recommendation systems and some effort of applying deep learning for recommendations.

The matrix factorization based approaches can be roughly grouped into two types, i.e., one type works only over the sole source of rating data and the other one needs some additional external information. Non-negative matrix factorization (NMF) has previously been shown to be a useful decomposition for multivariate data, and it is used to factorize the rating matrix into user and item profiles [9]. Paterek proposed to improve regularized singular value decomposition for predicting users' preferences for items in KDDCup'07 [16]. Salakhutdinov and Mnih [15, 18] further proposed Probabilistic Matrix Factorization (PMF) and Bayesian Probabilistic Matrix Factorization (BPMF). PMF adopts a probabilistic linear model with Gaussian observation noise for representing latent features both for users and items, which can perform well on the large and sparse rating data. In BPMF, the model capacity is controlled automatically by integrating over all model parameters and hyper parameters. To address the highly challenging to scale nonparametric matrix factorization models to large-scale problems, Yu et al. [25] introduced the novel optimization algorithms, which allow learning both singular value decomposition model and probabilistic principal component analysis to be highly efficient on large-scale problems. Sun et al. [21] modeled the ranked data, which are very important in the applications of recommendation systems and web search, to improve the recommendation performance. Lee et al. [10] also proposed a local collaborative ranking to consider ranked data, which is indeed an ensemble version of matrix factorization model. We also incorporate the ranked data into our framework.

To consider the external information, i.e., users' social relationships or item attributions, Ma et al. [13] employed both users' social network information and rating records, to solve the data sparsity and poor prediction accuracy problems. They also coined the term social trust ensemble to represent the formulation of the social trust restrictions, and proposed a novel probabilistic factor analysis framework to utilize the users' tastes and their trusted friends' favors together [12]. In [14], the additional social regularization term was proposed to ensure the distance of latent feature vectors of two friends with similar tastes to be closer. Yang et al. [24] inferred category-specific social trust circles from available rating data combined with friend relations. Recently, many studies have begun to utilize other types of information. For example, Cantador et al. [5] made use of user and item profiles defined in terms of weighted lists of social tags for top N recommendation. Furthermore, they presented a comparative study on the influence that different types of information available in social systems have on item recommendation [2]. Recently the heterogeneous information network, in which objects are of different types and links among objects represent different relations, are also considered to enhance the recommendation performance. Yu et al. [26] proposed an implicit feedback recommendation model with systematically extracted latent features from heterogeneous network. Wang et al. [6] proposed the OptRank method to alleviate the cold start problem by utilizing heterogeneous information contained in social tagging system. Shi et al. [20] proposed the concept of weighted heterogeneous information network and designed a meta-path based recommendation model called SemRec. The additional external information sometimes is not easy to acquire, therefore our model focuses on only the sole source of rating data.

There are already some effort based on deep learning techniques

Table 1: The Notation and Denotation

$\boldsymbol{M}$	The rating matrix
$\boldsymbol{I}$	The indicator matrix for observed user-item pairs
$\boldsymbol{U}$	The latent user feature matrix
$\boldsymbol{V}$	The latent item feature matrix
$\boldsymbol{R}$	Extension of $\boldsymbol{M}$
$\hat{\boldsymbol{R}}$	Reconstruction of $\boldsymbol{R}$
$\widetilde{\boldsymbol{I}}$	Extension of $\boldsymbol{I}$
$\boldsymbol{\xi}$	The hidden representation of $\boldsymbol{R}$
$\boldsymbol{W}$, $\boldsymbol{b}$	Encoding weight matrix and bias vector
$\boldsymbol{W}'$, $\boldsymbol{b}'$	Decoding weight matrix and bias vector
m	The number of users
n	The number of items
k	The number of nodes in hidden layer of autoencoder
s_u	The number of ordered item pairs of user u
$\top$	The transposition of a matrix
$\circ$	The element-wise product of vectors or matrices

devoted to recommendation systems. Salakhutdinov et al. [19] first used Restricted Boltzmann Machines (RBM's) to model tabular data, such as user's ratings of movies. Phung et al. [17] also explored and extended Boltzmann Machine for collaborative filtering tasks, which seamlessly integrates both the similarity and co-occurrence in a principled manner. Van et al. [22] proposed to use a latent factor model for recommendation, and predicted the latent factors from music audio when they cannot be obtained from usage data. Wang et al. [23] proposed the collaborative deep learning algorithm to address the sparsity problem, and considered auxiliary information, such as item content information. Their model jointly performed deep representation learning for the content information and collaborative filtering for the ratings matrix. Most of these approaches need the additional information, and do not make full use of the rating data, e.g., ranked loss. Our model imposes the latent representation learning and ranked loss regularization into a uniform framework, which leads to the improved performance for recommendation systems.

3. PRELIMINARY KNOWLEDGE

Some frequently used notations are summarized in Table 1, and without specified definition, all the vectors are column ones. In this section, we introduce detailed information about autoencoder and ranking learning which are the two main components in our framework.

3.1 Representation Learning with Autoencoder

The basic framework of antoencoder [3] is a neural network that consists two parts, the encoder and decoder. In the simplest case, there is one hidden layer and it can be defined as transitions ϕ (from input to hidden layer) and ψ (from hidden layer to output) such as,

$$\begin{aligned} &\phi : x \to f, \\ &\psi : f \to x', \\ &\arg\min_{\phi,\psi} ||x - (\psi \circ \phi)x||^2. \end{aligned} \tag{2}$$

Autoencoder minimises reconstruction errors as Eq. (2), and the latent variables of the hidden layer can be a feature representation of the input. When input the rating matrix $\boldsymbol{M}$ into autoencoder, the hidden layer forms the low-dimension feature $\boldsymbol{V}$ ($\boldsymbol{V} \in \mathbb{R}^{k\times n}$ and $k < m$) of items.

Low rank matrix approximation assumes that the rating matrix $\boldsymbol{M}$ is of low rank, and there is an approximate $\hat{\boldsymbol{M}}$ ($\hat{\boldsymbol{M}} = \boldsymbol{U}^\top \boldsymbol{V}$) can be used to estimate ratings and suggest recommendations. To get the best approximation $\hat{\boldsymbol{M}}$, a good representation $\boldsymbol{U}$ ($\boldsymbol{U} \in \mathbb{R}^{k\times m}$) and $\boldsymbol{V}$ should be learned first.

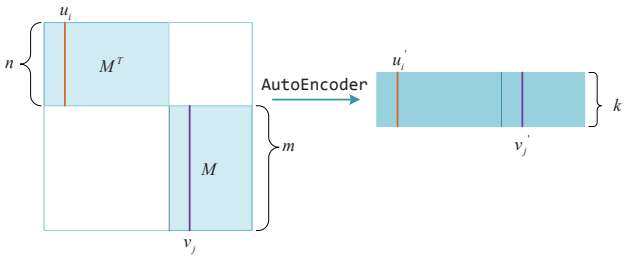


Figure 2: Representation Learning.

To learn the representation of users and items simultaneously, the rating matrix $\boldsymbol{M}$ is extended into $\boldsymbol{R}$ as showed in Figure 2 and $\boldsymbol{R} \in \mathbb{R}^{(m+n)\times(m+n)}$, which can be written as,

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{M}^\top & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{M} \end{bmatrix} \tag{3}$$

Through the encoding process of autoencoder, the latent presentations can be learned by,

$$\boldsymbol{\xi} = f(\boldsymbol{W}\boldsymbol{R} + \boldsymbol{b}) = \begin{bmatrix} \boldsymbol{U} & \boldsymbol{V} \end{bmatrix}, \tag{4}$$

where $\boldsymbol{W} \in \mathbb{R}^{k\times(m+n)}$ is the encoding weight matrix and sigmoid function $f(x) = 1/(1+e^{-x})$ is used as the activation function in our framework. f can be replaced by other radial-basis function here. In Figure 2, u_i is encoded into u_i' and v_j is encoded into v_j'. For an autoencoder, the dimension k of the latent layer can be any size, no matter bigger or smaller than that of the input. If k is smaller than the input, it can be regard as an information compressor. And k also can be set bigger than the input when it is needed. In this framework, k is set to be smaller than the number of users and items, that is $k < n$ and $k < m$, so users and items are projected into the same feature space. Then the latent representation is decoded into the same feature space as the input,

$$\hat{\boldsymbol{R}} = f(\boldsymbol{W}'\boldsymbol{\xi} + \boldsymbol{b}'), \tag{5}$$

where $\boldsymbol{W}' \in \mathbb{R}^{(m+n)\times k}$ is the decoding weight matrix. Autoencoder learns a satisfying representation of input via minimizing the reconstruction errors, which is formulated as,

$$\mathcal{J}_r = ||\widetilde{\boldsymbol{I}} \circ (\hat{\boldsymbol{R}} - \boldsymbol{R})||^2. \tag{6}$$

where matrix $\widetilde{\boldsymbol{I}}$ is the extension of matrix $\boldsymbol{I}$,

$$\widetilde{\boldsymbol{I}} = \begin{bmatrix} \boldsymbol{I}^\top & 0 \\ 0 & \boldsymbol{I} \end{bmatrix}. \tag{7}$$

In Eq. (6), just the observed user-item pairs are taken into consideration, and this is ensured by multiplying zero-one matrix $\widetilde{\boldsymbol{I}}$.

3.2 Ranking Learning

In recommendation system, the main target is to predict user's interest in some items which they have not noticed before. In other word, it aims at finding the most possible items which may be bought by users when these items are recommended to them. When training the observed data to predict the unobserved ones, it is necessary not only to keep the consistency of the observed values but

also the consistency of value ranking. Suppose item i and item j are both scored by user u, and have $\boldsymbol{I}_{u,i} = 1$ and $\boldsymbol{I}_{u,j} = 1$. When $\boldsymbol{M}_{u,i} > \boldsymbol{M}_{u,j}$, which means user u prefers item i than item j, then in the predicted result, the predicted value of user-item pair (u, i) should also be larger than that of (u, j). This is the consistency of value ranking in the observed data. The predicted score of items $\hat{\boldsymbol{M}}_{u,1}$, $\hat{\boldsymbol{M}}_{u,2}$,..., $\hat{\boldsymbol{M}}_{u,n}$ should have the same order as the observed one. It is obvious that a method considers both value and value ranking is more reasonable and should have higher accuracy than methods just focusing on one side.

As people have different criterion when marking items, and some people may give high score to an item even it is not so good in his eyes, while the others may give a low score even they are satisfied with it but has not reach an ideal level. Therefore when trying to keep the relative order of preferences, this constraint on different users should be made independently. The item sequence of each user is considered alone, so pair-wise method is a better choice than point-wise method.

Pair-wise methods care about the difference between variables rather than the values of them. For variable x and variable y, their difference $x - y$ is the practical variable in this method. Defining a transformation function τ, then $\tau(x) - \tau(y)$ is the new value of variable $x - y$. In this case, the change from $x - y$ to $\tau(x) - \tau(y)$ is the main point to be analyzed.

Pair-wise constraint of rank-based collaborative method prefers positive value of $\tau(x) - \tau(y)$ when $x > y$. When it is violated, there should be a loss. With this constraint, ranking learning can be optimized by minimizing the risk function with zero-one loss as

$$\begin{aligned}\varepsilon(f) &= \sum_{u} \sum_{\boldsymbol{M}_{u,i} > \boldsymbol{M}_{u,j}} \mathcal{L}(\boldsymbol{M}_{u,i} - \boldsymbol{M}_{u,j}, \tau(u,i) - \tau(u,j)), \\ \mathcal{L} &= 1\{(\boldsymbol{M}_{u,i} - \boldsymbol{M}_{u,j}) \cdot (\tau(u,i) - \tau(u,j)) < 0\}.\end{aligned} \tag{8}$$

Zero-one loss function gives an unit penalty to the pairs whose sign change after the transformation of function τ. For zero-one loss function is not differentiable, so it complicates the computation, and a corresponding smooth surrogate should be found. Margin-based loss function log-loss can be alternatively chosen as a substitute,

$$\mathcal{L} = \triangle \boldsymbol{M} \log(1 + \exp\{c - \triangle\tau\}), \tag{9}$$

where constant c is a free factor to control margin value. In [10], the similar zero-one loss function has been empirically proved to be valid on real-world data sets.

4. REPRESENTATION LEARNING WITH PAIR-WISE CONSTRAINTS FOR COLLABORATIVE RANKING

We are now ready to present the details of our REAP model. Firstly, we will formulate the optimization problem, and then introduce the optimization procedure of gradient descent and derive the solutions of the objective function. Finally, the algorithm is summarized in the pseudo-code, and how to make prediction is given.

4.1 Problem Formulation

To consider both value and ranking of the rating matrix, a combination framework is proposed in this paper. It learns user and item features with a single-hidden layer neural network autoencoder, and then gets the predicted matrix with the latent user and item feature matrix, finally an pair-wise constraint via log-loss function is enforced. The overall objective of REAP can be defined as,

$$\mathcal{J} = \varepsilon + \alpha \mathcal{J}_r + \gamma \Omega. \tag{10}$$

The first item ε is the loss function of pair-wise constraint to preserve the rating order of item sequence,

$$\varepsilon = \sum_{u \in U} \frac{1}{s_u} \sum_{k=1}^{s_u} \mathcal{L}(\Delta \boldsymbol{M}_k, \Delta f(x_k)), \tag{11}$$

where s_u is the number of ordered item-pairs rated by user u. In this item, the function f is the prediction result of autoencoder. The input $\boldsymbol{M}$ is compressed by autoencoder firstly, then predicted matrix derived from the multiplication of latent user feature matrix and item feature matrix. The second item $\mathcal{J}_r$ is the reconstruction error of autoencoder,

$$\mathcal{J}_r = ||\widetilde{\boldsymbol{I}} \circ (\hat{\boldsymbol{R}} - \boldsymbol{R})||^2. \tag{12}$$

User representations and item presentations are learned simultaneously by the same encoding and decoding weights, i.e., $\boldsymbol{W}$, $\boldsymbol{b}$, $\boldsymbol{W}'$ and $\boldsymbol{b}'$. The hidden layer has k nodes ($k < m, k < n$), and the weight matrixes $\boldsymbol{W} \in \mathbb{R}^{k \times (m+n)}$ and $\boldsymbol{b} \in \mathbb{R}^{k \times 1}$ connect the input layer and the hidden layer, while $\boldsymbol{W}' \in \mathbb{R}^{(m+n) \times k}$ and $\boldsymbol{b}' \in \mathbb{R}^{(m+n) \times 1}$ connect the hidden layer and the output. The rating matrix and its transposed one are both put into training data, so the reconstruction error of each instance of both user and item are minimized.

At last, to control the complexity of the autoencoder model, a weight decay item Ω is added to improve its generalization ability, which can be written as follows,

$$\Omega = ||\boldsymbol{W}||^2 + ||\boldsymbol{b}||^2 + ||\boldsymbol{W}'||^2 + ||\boldsymbol{b}'||^2. \tag{13}$$

The parameters α and γ balance each component. α confirms the importance of autoencoder, and γ is the trade-off parameter for the whole framework.

4.2 Solution Derivation of REAP

The optimization problem of our proposed framework is to minimize $\mathcal{J}$ (seen in Eq. (10)) as a function of $\boldsymbol{W}$, $\boldsymbol{b}$, $\boldsymbol{W}'$ and $\boldsymbol{b}'$. Obviously, it is an unconstrained optimization problem. To solve this problem, we adopt the gradient descent method to derive it. To simplify the math expressions, we first introduce the following intermediate variables,

$$\begin{aligned} A &= \boldsymbol{\xi} \circ (1 - \boldsymbol{\xi}), \\ B &= 2\widetilde{\boldsymbol{I}} \circ (\hat{\boldsymbol{R}} - \boldsymbol{R}) \circ \hat{\boldsymbol{R}} \circ (1 - \hat{\boldsymbol{R}}), \\ S(u,i,j) &= \frac{\partial \mathcal{L}(\Delta \boldsymbol{M}_{u,i,j}, g)}{\partial g(u,i,j)}. \end{aligned} \tag{14}$$

Firstly, we can get the partial derivatives of $g(u, i, j)$ with respect to the entries of $\boldsymbol{U}$ and $\boldsymbol{V}$. The partial derivatives of g are given by

$$\frac{\partial g(u,i,j)}{\partial \boldsymbol{U}_{k,u}} = \boldsymbol{V}_{k,i} - \boldsymbol{V}_{k,j}, \tag{15}$$

$$\frac{\partial g(u,i,j)}{\partial \boldsymbol{V}_{k,a}} = \begin{cases} \boldsymbol{U}_{k,u} & \text{if } a = i \\ -\boldsymbol{U}_{k,u} & \text{if } a = j \end{cases}, \tag{16}$$

$$\frac{\partial \mathcal{L}(\Delta \boldsymbol{M}_{u,i,j}, g)}{\partial g(u,i,j)} = \frac{-\Delta \boldsymbol{M} \exp^{-g}}{1 + \exp^{-g}}. \tag{17}$$

Then the partial derivatives of ε with respect to the entries of $\boldsymbol{U}$ and $\boldsymbol{V}$ can be computed as below.

$$\begin{aligned} \frac{\partial \varepsilon}{\partial \boldsymbol{U}_{k,u}} &= \frac{1}{s_u} \sum_{\Delta \boldsymbol{M}_{u,i,j} > 0} \frac{\partial \mathcal{L}(\Delta \boldsymbol{M}_{u,i,j}, g)}{\partial g(u,i,j)} \frac{\partial g(u,i,j)}{\partial \boldsymbol{U}_{k,u}} \\ &= \frac{1}{s_u} \sum_{\Delta \boldsymbol{M}_{u,i,j} > 0} S(u,i,j)\boldsymbol{V}_{k,i} - S(u,i,j)\boldsymbol{V}_{k,j}, \end{aligned} \tag{18}$$

$$\frac{\partial \varepsilon}{\partial \boldsymbol{V}_{k,v}} = \sum_{u \in U} \frac{1}{s_u} [\sum_{j:\boldsymbol{M}_{u,v} > \boldsymbol{M}_{u,j}} S(u,v,j)\boldsymbol{U}_{k,u} - \sum_{j:\boldsymbol{M}_{u,v} < \boldsymbol{M}_{u,j}} S(u,i,v)\boldsymbol{U}_{k,u}]. \tag{19}$$

Also, we can simplify the partial differential as,

$$E = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\xi}} = \begin{bmatrix} \frac{\partial \varepsilon}{\partial \boldsymbol{U}} & \frac{\partial \varepsilon}{\partial \boldsymbol{V}} \end{bmatrix} + \alpha(\boldsymbol{W}')^\top B \tag{20}$$

To minimize the objective $\mathcal{J}$, the partial differentials can be computed as below.

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{W}} = (E \circ A)\boldsymbol{R}^\top + 2\gamma \boldsymbol{W}, \tag{21}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{b}} = E \circ A + 2\gamma \boldsymbol{b}, \tag{22}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{W}'} = \alpha B \boldsymbol{\xi}^\top + 2\gamma \boldsymbol{W}', \tag{23}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{b}'} = \alpha B + 2\gamma \boldsymbol{b}'. \tag{24}$$

Based on the partial derivatives above, we develop an alternately iterating algorithm to derive the solutions with the following rules,

$$\begin{aligned} \boldsymbol{W} &\leftarrow \boldsymbol{W} - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{W}}, \quad & \boldsymbol{b} &\leftarrow \boldsymbol{b} - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{b}}, \\ \boldsymbol{W}' &\leftarrow \boldsymbol{W}' - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{W}'}, \quad & \boldsymbol{b}' &\leftarrow \boldsymbol{b}' - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{b}'}, \end{aligned} \tag{25}$$

where η is the step length, which determines the speed of convergence.

Though the objective of the whole framework is not convex, we can obtain relatively good results through appropriate initiation of the variables for the effective representation learning of autoencoder. Specially, we use Stacked AutoEncoder [4] to initialize the encoding and decoding weight matrices. The experimental results validate the effectiveness of the proposed solution. The details of our proposed algorithm are summarized in Algorithm 1.

Algorithm 1 Representation Learning with Pair-wise Constraints for Collaborative Ranking (REAP)

Input: rating matrix $\boldsymbol{M}$;
Parameters: rank k, balance coefficient α, regularization coefficient γ ;
Define: indicator matrix $\boldsymbol{I}$ to locate the observed data in rating matrix;
Output: user feature matrix $\boldsymbol{U}$ and item feature matrix $\boldsymbol{V}$.

1. Initialize $\boldsymbol{W}$, $\boldsymbol{b}$, $\boldsymbol{W}'$, $\boldsymbol{b}'$ by Stacked Autoencoders;
2. Compute the latent layer output of antoencoder by Eq. (4), and separate it into $\boldsymbol{U}$ and $\boldsymbol{V}$. Also get the predict matrix by $\hat{\boldsymbol{M}} = \boldsymbol{U}^\top \boldsymbol{V}$;
3. Compute the partial derivatives of all variables according to Eqs. (21) (22) (23) (24);
4. Iteratively update the variables using Eq. (25);
5. Continue Step 2), Step 3) and Step 4) until the algorithm converges;

4.3 Making Prediction

After the process showed by the pseudo-code in Algorithm 1, we can get user feature matrix and item feature matrix. Then we can predict an unseen test sample (u^*, i^*) as described in Algorithm 2.

Algorithm 2 The REAP prediction

1. **Input**: user feature matrix $\boldsymbol{U}$ and item feature matrix $\boldsymbol{V}$, test point (u^*, i^*);
2. **Return**: $\hat{\boldsymbol{M}}_{\boldsymbol{u^*,i^*}} = [\boldsymbol{U}^\top \boldsymbol{V}]_{u^*,i^*}$.

5. EXPERIMENTS

In this section, we demonstrate the effectiveness of the proposed collaborative learning framework by extensive experiments on five real-world rating datasets. We first compare our algorithm with various baselines and several state-of-the-art recommendation algorithms, and then investigate the parameter effect and computational complexity of our algorithm.

5.1 Data Preparation

We use five real-world datasets in the experiments for performance comparison among all the methods. One is MovieLens 100K[1] which contains 943 users and 1682 movies with $100,000$ ratings, as one of the most famous recommendation dataset. The second dataset is Douban book, which is crawled from the well known social media network in China[2]. This dataset contains 3399 users and 2394 books with 334788 ratings. The other three datasets are Yelp[3], Netflix[4] and MovieLens 1M[5]. The detailed statistical information of these five datasets are summarized in Table 2, including sparse dataset Yelp with a density of 0.88% only and much denser dataset MovieLens 100K with a density of 6.3%.

5.2 Baselines and Evaluation Metrics

5.2.1 Baselines

We compare our collaborative learning algorithms REAP with the following baseline algorithms,

- Non-negative Matrix Factorization (NMF) [9]: a robust matrix factorization algorithm which learns latent user and item profiles via factorizing the rating matrix.
- Probabilistic Matrix Factorization (PMF) [15]: it generalizes considerably nice result for users with very few ratings based on the assumption that users who have rated similar sets of items are likely to have similar preferences.
- Bayesian Probabilistic Matrix Factorization using Markov Chain Monte Carlo (BPMF) [18]: its model capacity is controlled automatically by integrating over all model parameters and hyper parameters to handle overfitting problems in PMF.

[1] http://www.grouplens.org/
[2] http://book.douban.com/
[3] http://recsys.acm.org/recsys13/recsys-2013-challenge-workshop/
[4] http://prea.gatech.edu/download/netflix_3m1k.zip
[5] http://www.grouplens.org/

Table 2: Statistical Information of Five DataSets

Dataset	#user	#item	#rating	Rating density (%)
MovieLens 100K	943	1682	100000	6.3
Douban book	3399	2394	334788	4.11
Yelp	4409	3419	132745	0.88
Netflix	4427	1000	56136	1.27
MovieLens 1M	6039	3883	1000209	4.27

- Rank-based Regularized SVD (RSVD)[6]: an improved implementation of Regularized SVD [16] [7] via rank-based collaborating.

- Rank-based Collaborative Filtering (CofiRank) [21]: a non-parametric model for rankings with missing items via mean-loss function (CofiRM) and asymmetric loss function (CofiRA).

Among all the baselines, NMF, PMF and BPMF are mainly based on matrix factorization to learn latent user and item profiles, and both RSVD and CofiRank take into account ranking learning in addition. Note that, our method not only learns latent user and item profiles by autoencoder, but also considers ranking learning in a uniform framework.

5.2.2 Implementation Details

We adopt the PREA toolkit [11] for the implementation of our compared algorithms, i.e., NMF, PMF, BPMF, RSVD, ColiRM and ColiRA. We use Stacked AutoEncoders to initiate the encoding and decoding weights of antoencoder in our framework.

As the rating score of all datasets vary from 1 to 5 and the input value range of autoencoder (when using sigmoid function) should be between 0 and 1, therefore we normalize the input rating matrix by dividing the max value in the rating matrix. The balance coefficient and regularization coefficient are determined by cross-validation. In the procedure of cross-validation, 40% of the training data is set aside for validation purpose. The max number of iterations of gradient descent method is set as 150. For each user, the number of training data N is set to be 5, 10, 15 and 20. The training procedure is stopped when the algorithm iterates exceeding the set max iteration times or the improvement in training error is smaller than the threshold, such as 0.0001.

5.2.3 Evaluation Metrics

We use two evaluation measures to evaluate our collaborative ranking algorithm. $NDCG@k$ and mean average precision are two widely used measures, which evaluate the performance of algorithms in preserving the relative order of items. $NDCG@k$ is computed by

$$\begin{aligned} NDCG@k &= \frac{DCG}{iDCG}, \\ DCG@k &= \sum_{i=1}^{k} \frac{2^{rel_i}-1}{\log_2(i+1)}, \end{aligned} \tag{26}$$

where i ranges over positions in the recommendation list and rel_i reflects the preference of the ith items by the user. Sort rel_i by descending order and compute like Eq. (26), then we can get $iDCG$. The result of dividing DCG by $iDCG$ indicates the difference of recommendation order and true order. In the experiments, we compute $NDCG@5$, $NDCG@10$ and$NDCG@15$ for all algorithms, so users in the training dataset should have at least rated $N + 15$ items since 15 items should be set aside for testing. Mean average precision is computed by

$$\begin{aligned} \text{AP} &= \frac{1}{r}\sum_{i=1}^{r} \frac{i}{the\ position\ of\ i-th\ relevant\ item}, \\ MAP &= \frac{1}{N}\sum_{i=1}^{N} AP_i. \end{aligned} \tag{27}$$

According to Eq. 27, we can find that MAP also pay close attention to the position of the recommendation items.

5.3 Experimental Results

To fairly compare REAP with RSVD, we show three group of comparison results of RSVD with the number of low-rank 5, 10 and 15 respectively in representation learning, denoted as RSVD5, RSVD10 and RSVD15. The detailed comparison results of $NDCG@5$, $NDCG@10$, $NDCG@15$ and MAE on MovieLens 100K are reported in Table 3 to 6. Figure 3 to 7 show the detailed results of $NDCG@5$, $NDCG@10$ and MAE on all five data sets (The results of $NDCG@15$ are similar with$NDCG@5$ and $NDCG@10$, so we omit it here due to the brevity). All the results recorded in the tables and figures are the means and variances of five independent trials.

From these results, we have the following insightful observations,

- With the increasing number of training data, the performance of all algorithms becomes better. Also it is worth mentioning that our model REAP can perform very well even there is only a small number of training data, i.e., $N = 5$, which indicates the effectiveness of REAP. This is very important since there is always scare of training data and not easy to acquire.

- Generally, the ranking leaning based methods, e.g., CofiRA and RSVD, perform better than the other baselines without consider ranking learning, e.g., NMF, PMF and BPMF, which shows the importance to consider ranking learning. Our method REAP is significantly better than the matrix-factorization based methods, i.e., NMF, PMF and BPMF, which indicates the superiority of applying autoencoder to simultaneously learn the latent representations of items and users.

- REAP can obtain significant improvement compared with the matrix-factorization based methods, e.g., NMF, RSVD, on different degrees of rating density of all data sets varying 0.88% to 6.3% (except that on Douban Book, RSVD is slightly better than REAP when there are sufficient training data, i.e., $N = 20$.), which again verifies the advantage of using autoencoder for recommendations and the effectiveness of REAP to deal with sparse data set, i.e., Yelp with rating density only 0.88%.

[6]http://prea.gatech.edu/download/prea_code_2_0.zip

Table 3: Performance Comparison on MovieLens 100K w.r.t $NDCG@5$

Method \ Metric	$NDCG@5$				
	$N = 5$	$N = 10$	$N = 15$	$N = 20$	Avg
PMF	0.839 ± 0.003	0.855 ± 0.004	0.861 ± 0.007	0.861 ± 0.004	0.854 ± 0.010
BPMF	0.836 ± 0.005	0.836 ± 0.005	0.849 ± 0.008	0.859 ± 0.012	0.845 ± 0.013
CofiRM	0.856 ± 0.000	0.850 ± 0.001	0.854 ± 0.001	0.856 ± 0.000	0.854 ± 0.002
CofiRA	0.862 ± 0.001	0.864 ± 0.001	0.860 ± 0.000	0.866 ± 0.000	0.863 ± 0.002
NMF	0.858 ± 0.005	0.856 ± 0.005	0.857 ± 0.003	0.858 ± 0.005	0.857 ± 0.005
RSVD5	0.873 ± 0.004	0.883 ± 0.003	0.890 ± 0.002	0.889 ± 0.003	0.884 ± 0.007
RSVD10	0.876 ± 0.004	0.882 ± 0.002	0.892 ± 0.004	0.887 ± 0.005	0.884 ± 0.007
RSVD15	0.876 ± 0.002	0.884 ± 0.003	0.890 ± 0.003	0.890 ± 0.002	0.885 ± 0.006
REAP	$\mathbf{0.902 \pm 0.000}$	$\mathbf{0.904 \pm 0.000}$	$\mathbf{0.899 \pm 0.000}$	$\mathbf{0.908 \pm 0.000}$	$\mathbf{0.903 \pm 0.004}$

Table 4: Performance Comparison on MovieLens 100K w.r.t $NDCG@10$

Method \ Metric	$NDCG@10$				
	$N = 5$	$N = 10$	$N = 15$	$N = 20$	Avg
PMF	0.806 ± 0.003	0.825 ± 0.003	0.833 ± 0.005	0.835 ± 0.003	0.825 ± 0.012
BPMF	0.801 ± 0.006	0.803 ± 0.007	0.821 ± 0.008	0.833 ± 0.015	0.814 ± 0.016
CofiRM	0.826 ± 0.000	0.818 ± 0.000	0.827 ± 0.000	0.832 ± 0.000	0.826 ± 0.005
CofiRA	0.835 ± 0.002	0.837 ± 0.002	0.833 ± 0.001	0.838 ± 0.002	0.836 ± 0.003
NMF	0.828 ± 0.005	0.831 ± 0.004	0.835 ± 0.002	0.835 ± 0.004	0.832 ± 0.005
RSVD5	0.849 ± 0.003	0.860 ± 0.002	0.869 ± 0.002	0.868 ± 0.002	0.862 ± 0.009
RSVD10	0.850 ± 0.002	0.858 ± 0.002	0.870 ± 0.003	0.867 ± 0.003	0.861 ± 0.008
RSVD15	0.849 ± 0.001	0.861 ± 0.002	0.869 ± 0.002	0.869 ± 0.002	0.862 ± 0.008
REAP	$\mathbf{0.883 \pm 0.000}$	$\mathbf{0.884 \pm 0.000}$	$\mathbf{0.885 \pm 0.000}$	$\mathbf{0.888 \pm 0.000}$	$\mathbf{0.885 \pm 0.002}$

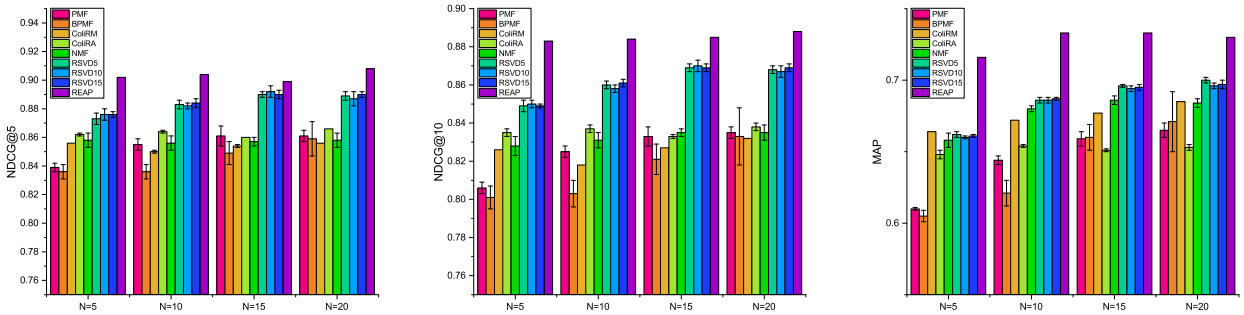
Table 5: Performance Comparison on MovieLens 100K w.r.t $NDCG@15$

Method \ Metric	$NDCG@15$				
	$N = 5$	$N = 10$	$N = 15$	$N = 20$	Avg
PMF	0.798 ± 0.002	0.818 ± 0.002	0.827 ± 0.004	0.830 ± 0.003	0.818 ± 0.013
BPMF	0.792 ± 0.005	0.795 ± 0.007	0.816 ± 0.007	0.828 ± 0.015	0.808 ± 0.018
CofiRM	0.818 ± 0.000	0.813 ± 0.000	0.818 ± 0.000	0.828 ± 0.000	0.819 ± 0.005
CofiRA	0.828 ± 0.002	0.829 ± 0.002	0.826 ± 0.002	0.831 ± 0.002	0.829 ± 0.003
NMF	0.819 ± 0.004	0.826 ± 0.003	0.831 ± 0.002	0.832 ± 0.004	0.827 ± 0.006
RSVD5	0.842 ± 0.002	0.854 ± 0.001	0.863 ± 0.002	0.864 ± 0.002	0.856 ± 0.009
RSVD10	0.842 ± 0.002	0.852 ± 0.001	0.863 ± 0.003	0.862 ± 0.003	0.855 ± 0.009
RSVD15	0.843 ± 0.001	0.855 ± 0.001	0.862 ± 0.002	0.862 ± 0.002	0.856 ± 0.008
REAP	$\mathbf{0.875 \pm 0.000}$	$\mathbf{0.880 \pm 0.000}$	$\mathbf{0.880 \pm 0.000}$	$\mathbf{0.881 \pm 0.000}$	$\mathbf{0.879 \pm 0.002}$

Table 6: Performance Comparison on MovieLens 100K w.r.t MAP

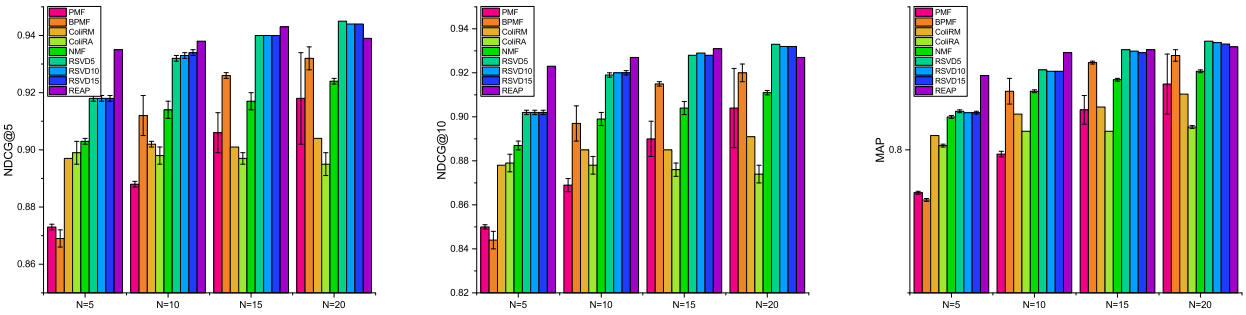
Method \ Metric	MAP				
	$N = 5$	$N = 10$	$N = 15$	$N = 20$	Avg
PMF	0.610 ± 0.001	0.644 ± 0.003	0.659 ± 0.005	0.665 ± 0.005	0.644 ± 0.022
BPMF	0.605 ± 0.004	0.621 ± 0.009	0.660 ± 0.009	0.671 ± 0.021	0.639 ± 0.030
CofiRM	0.664 ± 0.000	0.672 ± 0.000	0.677 ± 0.000	0.685 ± 0.000	0.674 ± 0.008
CofiRA	0.648 ± 0.003	0.654 ± 0.001	0.651 ± 0.001	0.653 ± 0.002	0.652 ± 0.003
NMF	0.658 ± 0.005	0.680 ± 0.002	0.686 ± 0.003	0.684 ± 0.003	0.677 ± 0.012
RSVD5	0.662 ± 0.002	0.686 ± 0.002	0.696 ± 0.001	0.700 ± 0.002	0.686 ± 0.015
RSVD10	0.660 ± 0.001	0.686 ± 0.002	0.694 ± 0.002	0.696 ± 0.002	0.684 ± 0.014
RSVD15	0.661 ± 0.001	0.687 ± 0.001	0.695 ± 0.002	0.697 ± 0.003	0.685 ± 0.015
REAP	$\mathbf{0.716 \pm 0.000}$	$\mathbf{0.733 \pm 0.000}$	$\mathbf{0.733 \pm 0.000}$	$\mathbf{0.730 \pm 0.000}$	$\mathbf{0.728 \pm 0.007}$

- Furthermore, the value of variance of our model REAP is much smaller than the baselines', which shows that REAP can achieve the most stable performance.
- Overall, REAP outperforms all the baselines, which validates the effectiveness of incorporating autoencoder and ranking learning into a uniform framework.



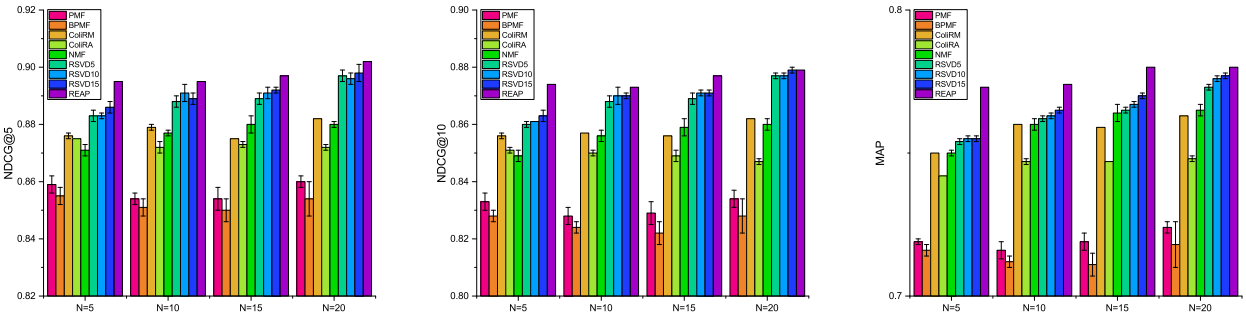
(a) Results of NDCG@5 (b) Results of NDCG@10 (c) Results of MAP

Figure 3: The Comparison Results of MovieLens 100K



(a) Results of NDCG@5 (b) Results of NDCG@10 (c) Results of MAP

Figure 4: The Comparison Results of Douban Book



(a) Results of NDCG@5 (b) Results of NDCG@10 (c) Results of MAP

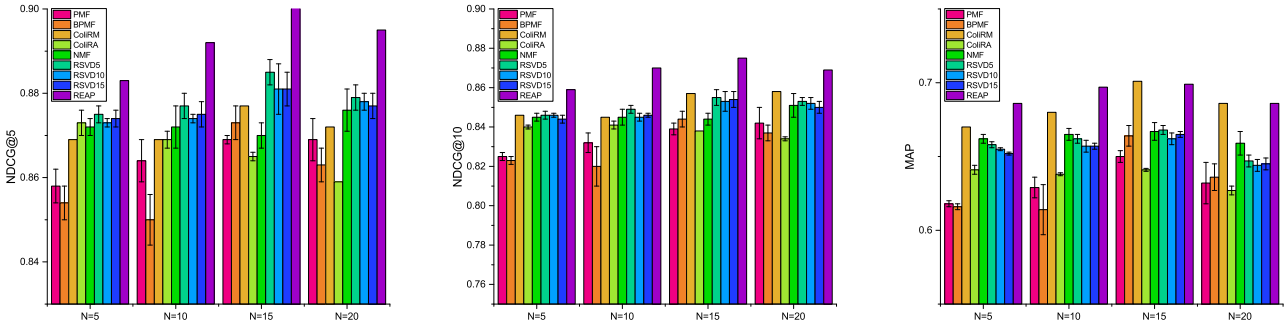
Figure 5: The Comparison Results of Yelp

5.4 Parameter Sensitivity

In this section, we introduce how to set the parameters by cross-validation, and show the results on the validation data set of MovieLens 100K to analyze the effect of the rank k and balance coefficient α in Figure 8(a) and Figure 8(b). The number of sampled training data is 5, 10, 15 and 20, respectively, and when tuning one parameter, the other is fixed. In this experiment, k is sampled from 15 to 300 with interval 15, and α is from 50 to 1000 with interval 50. From these results, we can find that REAP can perform very stable under different settings of k and α , and finally we set $k = 45$ and $\alpha = 500$ for MovieLens 100K to achieve good and stable results. The parameters of other data sets are similarly determined by the same way.

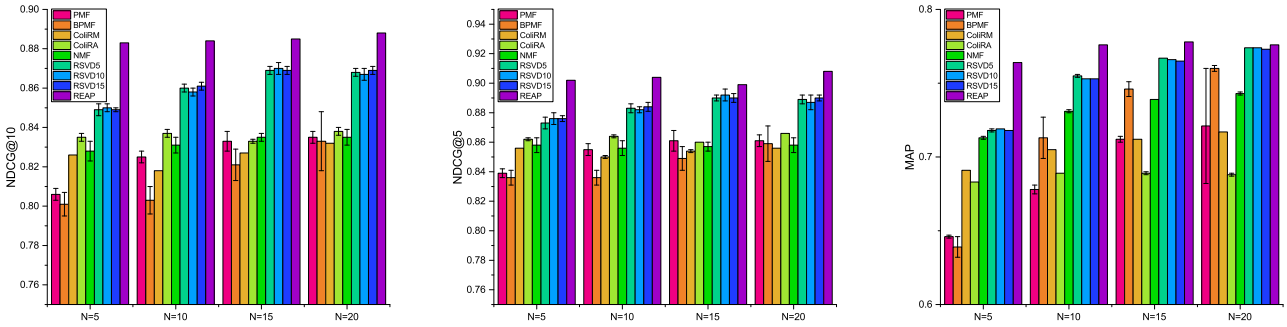
5.5 Computational Complexity

We also analyze the computational complexity of our algorithm. For each round of iteration in algorithm 1, the computational complexity of Eq. (4) to learn the user and item latent representations $\boldsymbol{\xi}$ is $O(k(m+n)^2)$, and the one of Eq. (11) to compute the pairwise loss ε is $O(cm)$. Since users usually rated very few items, so the coefficient c is far smaller than n. Moreover, the computational complexity of Eq. (13) is $O(k(m+n))$, so the computational complexity of Eq. (10) is $O(k(m+n)^2)$. Similarly, the computational complexities of Eqs. (21) (22) (23) (24) are respectively $O(k(m+n)^2)$, $O(k(m+n)^2)$, $O(k(m+n)^2)$ and $O((m+n)^2)$. Given the number of iterations T, the maximal computational complexity is $O(Tk(m+n)^2)$, which mainly depends on the number of users and items. Fortunately, the rating matrix is always very



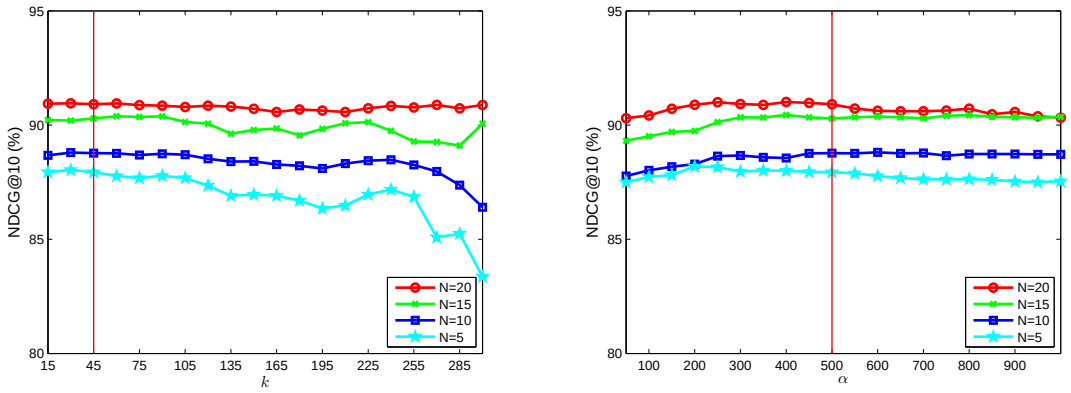
(a) Results of NDCG@5 (b) Results of NDCG@10 (c) Results of MAP

Figure 6: The Comparison Results of Netflix



(a) Results of NDCG@5 (b) Results of NDCG@10 (c) Results of MAP

Figure 7: The Comparison Results of MovieLens 1M



(a) The parameter influence of k (b) The parameter influence of α

Figure 8: Investigation of Parameter Influence

sparse, and sparse matrix computation in Matlab can dramatically reduce the computational intensity.

6. CONCLUSIONS

In this paper, we proposed a novel collaborative ranking framework via representation learning with pair-wise constraints. In this framework, the well known representation learning model autoencoder is used to learn the user and item profile, and we combine pair-wise constraint with it by an log-loss function. The loss of pair-wise constraint supervises the feature learning of antoencoder further. Extensive experiments on five real-world data sets demonstrate the effectiveness of the proposed framework.

7. ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (NO. 9154610306, 61573335, 61473273, 61473274), National High-tech R&D Program of China (863 Program) (NO. 2014AA015105), Guangdong provincial science and technology plan projects (NO. 2015 B010109005).

8. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. In *IEEE Transactions on Knowledge and Data Engineering*, volume 177, pages 734–749, 2005.

[2] A. Bellogín, I. Cantador, and P. Castells. A comparative study of heterogeneous item recommendations in social systems. *Information Sciences*, 221:142–169, 2013.

[3] Y. Bengio. Learning deep architectures for ai. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.

[4] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, U. D. MontrÃl'al, and M. QuÃl'bec. Greedy layer-wise training of deep networks. In *In NIPS*. MIT Press, 2007.

[5] I. Cantador, A. Bellogín, and D. Vallet. Content-based recommendation in social tagging systems. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 237–240. ACM, 2010.

[6] W. Feng and J. Wang. Incorporating heterogeneous information for personalized tag recommendation in social tagging systems. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1276–1284. ACM, 2012.

[7] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *In Proc. of the 14th ACM SIGKDD conference*, pages 426–434, 2008.

[8] Y. Koren and R. Bell. *Recommender Systems Handbook*, chapter Advances in collaborative filtering. Number 145–186. Springer, 2011.

[9] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[10] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer. Local collaborative ranking. In *Proceedings of the 23rd international conference on World wide web*, pages 85–96. ACM, 2014.

[11] J. Lee, M. Sun, and G. Lebanon. Prea: personalized recommendation algorithms toolkit. *Journal of Machine Learning Research*, 13(3):2699–2703, 2012.

[12] H. Ma, I. King, and M. R. Lyu. Learning to recommend with social trust ensemble. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 203–210. ACM, 2009.

[13] H. Ma, H. Yang, M. R. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 931–940. ACM, 2008.

[14] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 287–296. ACM, 2011.

[15] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2007.

[16] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

[17] D. Q. Phung, S. Venkatesh, et al. Ordinal boltzmann machines for collaborative filtering. In *Proceedings of the Twenty-fifth Conference on Uncertainty in Artificial Intelligence*, pages 548–556. AUAI Press, 2009.

[18] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM, 2008.

[19] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

[20] C. Shi, Z. Zhang, P. Luo, P. S. Yu, Y. Yue, and B. Wu. Semantic path based personalized recommendation on weighted heterogeneous information networks. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 453–462. ACM, 2015.

[21] M. Sun, G. Lebanon, and P. Kidwell. Estimating probabilities in recommendation systems. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 61(3):471–492, 2012.

[22] A. Van den Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, pages 2643–2651, 2013.

[23] H. Wang, N. Wang, and D.-Y. Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1235–1244, New York, NY, USA, 2015. ACM.

[24] X. Yang, H. Steck, and Y. Liu. Circle-based recommendation in online social networks. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1267–1275. ACM, 2012.

[25] K. Yu, S. Zhu, J. Lafferty, and Y. Gong. Fast nonparametric matrix factorization for large-scale collaborative filtering. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 211–218. ACM, 2009.

[26] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 283–292. ACM, 2014.