Neural Networks 90 (2017) 83–89

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Representation learning via Dual-Autoencoder for recommendation

Fuzhen Zhuang [a,*], Zhiqiang Zhang [b], Mingda Qian [a], Chuan Shi [b], Xing Xie [c], Qing He [a]

[a] Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[b] Beijing University of Posts and Telecommunications, Beijing, China
[c] Microsoft Research, China

ARTICLE INFO

Article history:
Received 10 August 2016
Received in revised form 19 January 2017
Accepted 17 March 2017
Available online 27 March 2017

Keywords:
Matrix factorization
Dual-Autoencoder
Recommendation
Representation learning

ABSTRACT

Recommendation has provoked vast amount of attention and research in recent decades. Most previous works employ matrix factorization techniques to learn the latent factors of users and items. And many subsequent works consider external information, e.g., social relationships of users and items' attributions, to improve the recommendation performance under the matrix factorization framework. However, matrix factorization methods may not make full use of the limited information from rating or check-in matrices, and achieve unsatisfying results. Recently, deep learning has proven able to learn good representation in natural language processing, image classification, and so on. Along this line, we propose a new representation learning framework called *Re*commendation via *D*ual-*A*utoencoder (ReDa). In this framework, we simultaneously learn the new hidden representations of users and items using autoencoders, and minimize the deviations of training data by the learnt representations of users and items. Based on this framework, we develop a gradient descent method to learn hidden representations. Extensive experiments conducted on several real-world data sets demonstrate the effectiveness of our proposed method compared with state-of-the-art matrix factorization based methods.

1. Introduction

In order to tackle the information overload problem, recommender systems are proposed to help users to find objects of interest by utilizing the user–item interaction information and/or content information associated with users and items. Recommender systems have attracted much attention from multiple disciplines, and many techniques have been proposed to build recommender systems (Adomavicius & Tuzhilin, 2005; Bell, 2011). It is also widely used in many E-commerce companies, such as for product sale on Amazon and movie rentals from Netflix (Srebro & Jaakkola, 2003).

Traditional recommender systems normally only utilize user–item rating feedback information for recommendations. Moreover, the matrix factorization technique is widely used in recommender systems, which factorizes a user–item rating matrix into two low rank user-specific and item-specific matrices, and then utilizes the factorized matrices to make further predictions (Koren, Bell, & Volinsky, 2009; Srebro & Jaakkola, 2003). In order to comprehensively utilize rich information in recommender system, there is a surge of hybrid recommendation, such as social recommendation (Ma, Yang, Lyu, & King, 2008; Ma, Zhou, Lyu, & King, 2011), location based recommendation (Lian et al., 2014; Liu, Fu, Yao, & Xiong, 2013), and heterogeneous network based recommendation (Shi et al., 2015; Yu et al., 2014). Most of these methods are based on a matrix factorization framework, in which the latent factors of users and items are usually obtained by directly factorizing the user–item rating matrix and additional information are usually used as a regularization constraint. Although these methods pay much attention to exploit additional information, we wonder they might not make full use of the user–item rating information. In other words, we might be able to obtain better latent factors of users and items through extensively exploiting rating information.

On the other hand, deep learning has shown its power in learning latent feature representation in many domains, such as image/video processing (Alex et al., 2009) and text data (Socher, Huang, Pennington, Ng, & Manning, 2011). Can we use deep learning techniques to learn latent representations for recommendation? Some researchers have pursued this goal. For example, the latent factor of music is extracted from audio signals with a deep convolutional neural network (Oord, Dieleman, & Schrauwen, 2013) and the tabular data is modeled through the adaption

* Corresponding author.
E-mail addresses: zhuangfz@ics.ict.ac.cn (F. Zhuang), zzqsmall@gmail.com (Z. Zhang), qianmd@ics.ict.ac.cn (M. Qian), shichuan@bupt.edu.cn (C. Shi), xing.xie@microsoft.com (X. Xie), heq@ics.ict.ac.cn (Q. He).

http://dx.doi.org/10.1016/j.neunet.2017.03.009
0893-6080/

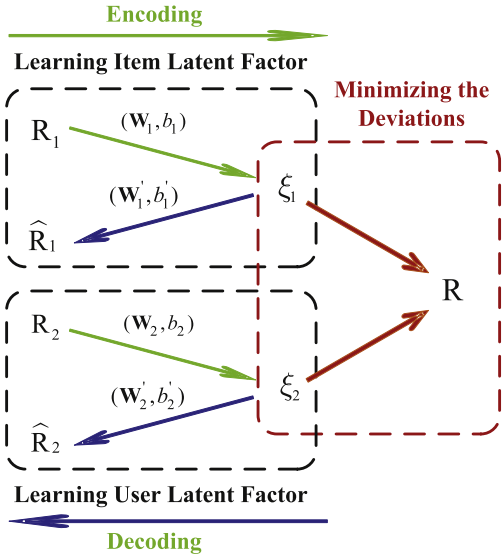


Fig. 1. Recommendation via Dual-Autoencoders ($\boldsymbol{R}$ is a rating matrix or check-in matrix, $\boldsymbol{R}_1 = \boldsymbol{R}, \boldsymbol{R}_2 = \boldsymbol{R}^\top$).

of Restricted Boltzmann Machines (Salakhutdinov, Mnih, & Hinton, 2007). Recently, Wang, Wang, and Yeung (2014) designed a collaborative deep learning method to utilize item content information. In essence, these methods utilize the powerful representation learning of deep learning to analyze the additional information (e.g., audio signal and text context), not rating information, for better recommendations. They did not directly learn the latent factors of users and items with deep learning. Moreover, the additional information sometimes is not easy to acquire and very sparse.

To the best of our knowledge, there has been little effort focused on employing deep learning for recommendations only on user–item rating information. Motivated by the success of the latent feature representation of deep learning on image and text data, we design a novel *Re*commendation framework via *D*ual-*A*utoencoders (ReDa), which is illustrated in Fig. 1. In this figure, ReDa simultaneously learns the new hidden representations of users and items using autoencoders, and minimizes the deviations of training data by the learnt representations of users and items. Moreover, a gradient descent method is derived to learn the hidden representations. Experiments on four real-world data sets demonstrate the effectiveness of our proposed model.

The remainder of this paper is organized as follows. We introduce the notations and preliminary knowledge in Section 2, and then propose the representation learning framework based on autoencoders for recommendation in Section 3. Extensive experiments conducted on several data sets are shown in Section 4, followed by the related work in Section 5 and conclusions in Section 6.

2. Notations and preliminaries

In this section, we first introduce some frequently used notations as presented in Table 1, and some preliminaries which will be used in our proposed framework.

2.1. Autoencoders

An autoencoder (Bengio, 2009) first maps an input instance $\boldsymbol{x} \in \mathbb{R}^{m\times 1}$ to a hidden representation $\boldsymbol{\xi}$ through an encoding mapping:

$$\boldsymbol{\xi} = h(\boldsymbol{W}\boldsymbol{x} + b),$$

where h is a nonlinear activation function, $\mathbf{W} \in \mathbb{R}^{k\times m}$ is a weight matrix, and b is a bias. The resulting latent representation $\boldsymbol{\xi}$ is then mapped back to a reconstruction $\hat{\boldsymbol{x}}$ through a decoding mapping:

$$\hat{\boldsymbol{x}} = g(\boldsymbol{W}'\boldsymbol{\xi} + b'),$$

where g is a nonlinear activation function, $\mathbf{W}' \in \mathbb{R}^{m\times k}$ is a weight matrix, and b' is a bias vector. Given a set of inputs $\{\boldsymbol{x}_i\}_{i=1}^n$, the parameters of an autoencoder are optimized by minimizing the reconstruction error as follows,

$$\min_{\mathbf{W},b,\mathbf{W}',b'} = \sum_{i=1}^{n} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2. \tag{1}$$

Note that, in this paper we adopt the sigmoid function $\sigma(a) = \frac{1}{1+e^{-a}}$, which is widely used in constructing autoencoders, as the nonlinear activation functions g and h for encoding and decoding, respectively. As has been proven previously, autoencoder can find a powerful feature representation $\boldsymbol{\xi}$ given the input instance $\boldsymbol{x}$.

Indeed, the essential idea of matrix factorization-based recommendation models is to find good latent factors of users and items. This insightful observation motivates us to apply an autoencoder to achieve better latent representations. Analogically, given the rating matrix $\boldsymbol{R} \in \mathbb{R}^{m\times n}$ (or check-in matrix), where m and n are respectively the numbers of users and items, each item can be seen as an input instance and the users can be regarded as its corresponding features. We can then use the autoencoder to learn the latent representations of items. Similarly, if we take items as the features of each user, we can also learn the latent representations of users.

Table 1
The notation and denotation.

$\boldsymbol{R}$	The rating matrix or check-in matrix
$\boldsymbol{I}$	The indicator matrix
m	The number of users
n	The number of items
k	The number of hidden factors/features
$\boldsymbol{x}$	An original instance
$\hat{\boldsymbol{x}}$	The reconstruction of $\boldsymbol{x}$
$\boldsymbol{\xi}$	An embedded instance
$\mathbf{W}, b$	A weight matrix and bias of encoding
$\mathbf{W}', b'$	A weight matrix and bias of decoding
$\top$	The transposition of a matrix
$\circ$	The element-wise product of vectors or matrices

3. Representation learning via Dual-Autoencoders

In this section, we first formulate the representation learning framework via dual-autoencoders for recommendation, and then derive the model solution using the gradient decent method.

3.1. Problem formalization

The recommendation problem is to predict the rating score (or check-in interest) of a user on an item (or business) based on the historical information. Given the rating matrix $\boldsymbol{R} \in \mathbb{R}^{m\times n}$, where m is the number of users and n is the number of items, and $\boldsymbol{I} \in \mathbb{R}^{m\times n}$ is the indicator matrix, where $I_{ij} = 1$ if $R_{ij} \neq 0$, or else $I_{ij} = 0$, then the objective function to learn the latent representations of items is formulated as,

$$\mathcal{J}_b = \|\boldsymbol{I}_1 \circ (\boldsymbol{R}_1 - \hat{\boldsymbol{R}}_1)\|^2, \tag{2}$$

where $\boldsymbol{R}_1 = \boldsymbol{R}$ and $\boldsymbol{I}_1 = \boldsymbol{I}$. The encoding and decoding processes are as follows,

$$\begin{aligned} \boldsymbol{\Xi}_1 &= f(\boldsymbol{W}_1\boldsymbol{R}_1 + \boldsymbol{e}_1\boldsymbol{b}_1^\top), \\ \hat{\boldsymbol{R}}_1 &= f(\boldsymbol{W}_1'\boldsymbol{\Xi}_1 + \boldsymbol{e}_1'\boldsymbol{b}_1'^\top), \end{aligned} \tag{3}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{k\times m}$ and $\boldsymbol{W}_1' \in \mathbb{R}^{m\times k}$ are the weight matrices, $\boldsymbol{\Xi}_1 \in \mathbb{R}^{k\times n}$ is the latent representation of items, $\boldsymbol{b}_1 \in \mathbb{R}^{n\times 1}$ and $\boldsymbol{b}_1' \in \mathbb{R}^{n\times 1}$

are bias vectors, $\boldsymbol{e}_1 \in \mathbb{R}^{k\times 1}$ and $\boldsymbol{e}'_1 \in \mathbb{R}^{m\times 1}$ are two constant vectors with each entry equal to 1, and f is the sigmoid function.

Similarly, we can formulate the objective function for learning the latent factors of users as,

$$\mathcal{J}_a = \|\boldsymbol{I}_2 \circ (\boldsymbol{R}_2 - \hat{\boldsymbol{R}}_2)\|^2, \tag{4}$$

where $\boldsymbol{R}_2 = \boldsymbol{R}^\top$, $\boldsymbol{I}_2 = \boldsymbol{I}^\top$ are the transposition matrices of $\boldsymbol{R}$, $\boldsymbol{I}$, respectively. The encoding and decoding processes are as follows,

$$\begin{aligned} \boldsymbol{\Xi}_2 &= f(\boldsymbol{W}_2\boldsymbol{R}_2 + \boldsymbol{e}_2\boldsymbol{b}_2^\top), \\ \hat{\boldsymbol{R}}_2 &= f(\boldsymbol{W}'_2\boldsymbol{\Xi}_2 + \boldsymbol{e}'_2\boldsymbol{b}_2'^\top), \end{aligned} \tag{5}$$

where $\boldsymbol{W}_2 \in \mathbb{R}^{k\times n}$ and $\boldsymbol{W}'_2 \in \mathbb{R}^{n\times k}$ are the weight matrices, $\boldsymbol{\Xi}_2 \in \mathbb{R}^{k\times m}$ is the latent representation of users, $\boldsymbol{b}_2 \in \mathbb{R}^{m\times 1}$ and $\boldsymbol{b}'_2 \in \mathbb{R}^{m\times 1}$ are bias vectors, $\boldsymbol{e}_2 \in \mathbb{R}^{k\times 1}$ and $\boldsymbol{e}'_2 \in \mathbb{R}^{n\times 1}$ are two constant vectors with each entry equal to 1.

For the last item, we also require the learnt latent representations of users and items from autoencoders to be able to minimize the deviations of training data, which is formulated as,

$$\mathcal{J}_c = \|\boldsymbol{I} \circ (\boldsymbol{R} - \boldsymbol{\Xi}_2^\top \boldsymbol{\Xi}_1)\|^2. \tag{6}$$

Finally, the total optimization problem of our proposed framework is,

$$\begin{aligned} \mathcal{J} = {} & \mathcal{J}_c + \alpha \cdot \mathcal{J}_b + \beta \cdot \mathcal{J}_a \\ & + \gamma \cdot (\|\boldsymbol{W}_1\|^2 + \|\boldsymbol{b}_1\|^2 + \|\boldsymbol{W}'_1\|^2 + \|\boldsymbol{b}'_1\|^2 \\ & + \|\boldsymbol{W}_2\|^2 + \|\boldsymbol{b}_2\|^2 + \|\boldsymbol{W}'_2\|^2 + \|\boldsymbol{b}'_2\|^2). \end{aligned} \tag{7}$$

The fourth item is a regularization on model parameters, and α, β, γ are trade-off parameters. α and β control the importance of learning latent factors using autoencoders, and their larger values mean we pay more attention on the optimization of autoencoders. In this framework, we simultaneously learn the latent representations of users and items using the autoencoder, and minimize the deviations by the learnt representations for recommendation. Therefore, we call our framework as *Re*commendation via *D*ual-*A*utoencoders (ReDa for short).

3.2. Model learning

The optimization problem in Eq. (7) is an unconstrained optimization, and does not have closed form solutions. To derive the solutions of all variables, we propose to use gradient descent methods. To simplify the mathematical expressions, we first introduce the following intermediate variables.

$$\begin{aligned} \mathcal{A}_1 &= \boldsymbol{\Xi}_1 \circ (1 - \boldsymbol{\Xi}_1), \\ \mathcal{A}_2 &= \boldsymbol{\Xi}_2 \circ (1 - \boldsymbol{\Xi}_2), \\ \mathcal{B}_1 &= (\boldsymbol{R}_1 - \hat{\boldsymbol{R}}_1) \circ \hat{\boldsymbol{R}}_1 \circ (1 - \hat{\boldsymbol{R}}_1), \\ \mathcal{B}_2 &= (\boldsymbol{R}_2 - \hat{\boldsymbol{R}}_2) \circ \hat{\boldsymbol{R}}_2 \circ (1 - \hat{\boldsymbol{R}}_2), \end{aligned} \tag{8}$$

the partial derivatives of the objective $\mathcal{J}$ in Eq. (7) w.r.t. $\boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{W}'_1, \boldsymbol{b}'_1, \boldsymbol{W}_2, \boldsymbol{b}_2, \boldsymbol{W}'_2$, and $\boldsymbol{b}'_2$ can be computed as follows respectively,

$$\begin{aligned} \frac{\partial \mathcal{J}}{\partial \boldsymbol{W}_1} = {} & -2(\boldsymbol{\Xi}_2(\boldsymbol{I} \circ (\boldsymbol{R} - \boldsymbol{\Xi}_2^\top \boldsymbol{\Xi}_1)) \circ \mathcal{A}_1)\boldsymbol{R}_1^\top \\ & - 2\alpha \boldsymbol{W}_1'^\top (\boldsymbol{I}_1 \circ \mathcal{B}_1) \circ \mathcal{A}_1 \boldsymbol{R}_1^\top + 2\gamma \cdot \boldsymbol{W}_1, \end{aligned} \tag{9}$$

$$\begin{aligned} \frac{\partial \mathcal{J}}{\partial \boldsymbol{b}_1} = {} & -2[\boldsymbol{\Xi}_2(\boldsymbol{I} \circ (\boldsymbol{R} - \boldsymbol{\Xi}_2^\top \boldsymbol{\Xi}_1)) \circ \mathcal{A}_1]^\top \boldsymbol{e}_1 \\ & - 2\alpha [\boldsymbol{W}_1'^\top (\boldsymbol{I}_1 \circ \mathcal{B}_1) \circ \mathcal{A}_1]^\top \boldsymbol{e}_1 + 2\gamma \cdot \boldsymbol{b}_1, \end{aligned} \tag{10}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{W}'_1} = -2\alpha \boldsymbol{I}_1 \circ \mathcal{B}_1 \boldsymbol{\Xi}_1^\top + 2\gamma \cdot \boldsymbol{W}'_1, \tag{11}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{b}'_1} = -2[\alpha \boldsymbol{I}_1 \circ \mathcal{B}_1]^\top \boldsymbol{e}'_1 + 2\gamma \cdot \boldsymbol{b}'_1, \tag{12}$$

$$\begin{aligned} \frac{\partial \mathcal{J}}{\partial \boldsymbol{W}_2} = {} & -2(\boldsymbol{\xi}_1(\boldsymbol{I} \circ (\boldsymbol{R} - \boldsymbol{\Xi}_2^\top \boldsymbol{\Xi}_1))^\top \circ \mathcal{A}_2)\boldsymbol{R}_2^\top \\ & - 2\beta \boldsymbol{W}_2'^\top (\boldsymbol{I}_2 \circ \mathcal{B}_2) \circ \mathcal{A}_2 \boldsymbol{R}_2^\top + 2\gamma \cdot \boldsymbol{W}_2, \end{aligned} \tag{13}$$

$$\begin{aligned} \frac{\partial \mathcal{J}}{\partial \boldsymbol{b}_2} = {} & -2[\boldsymbol{\Xi}_1(\boldsymbol{I} \circ (\boldsymbol{R} - \boldsymbol{\Xi}_2^\top \boldsymbol{\Xi}_1)^\top) \circ \mathcal{A}_2]^\top \boldsymbol{e}_2 \\ & - 2[\beta \boldsymbol{W}_2'^\top (\boldsymbol{I}_2 \circ \mathcal{B}_2) \circ \mathcal{A}_2]^\top \boldsymbol{e}_2 + 2\gamma \cdot \boldsymbol{b}_2, \end{aligned} \tag{14}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{W}'_2} = -2\beta(\boldsymbol{I}_2 \circ \mathcal{B}_2)\boldsymbol{\Xi}_2^\top + 2\gamma \cdot \boldsymbol{W}'_2, \tag{15}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{b}'_2} = -2[\beta \boldsymbol{I}_2 \circ \mathcal{B}_2]^\top \boldsymbol{e}'_2 + 2\gamma \cdot \boldsymbol{b}'_2. \tag{16}$$

Based on the above partial derivatives, we develop an alternatively iterating algorithm to derive the solutions by using the following rules,

$$\begin{aligned} \boldsymbol{W}_1 &\leftarrow \boldsymbol{W}_1 - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{W}_1}, \quad \boldsymbol{b}_1 \leftarrow \boldsymbol{b}_1 - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{b}_1}, \\ \boldsymbol{W}'_1 &\leftarrow \boldsymbol{W}'_1 - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{W}'_1}, \quad \boldsymbol{b}'_1 \leftarrow \boldsymbol{b}'_1 - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{b}'_1}, \\ \boldsymbol{W}_2 &\leftarrow \boldsymbol{W}_2 - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{W}_2}, \quad \boldsymbol{b}_2 \leftarrow \boldsymbol{b}_2 - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{b}_2}, \\ \boldsymbol{W}'_2 &\leftarrow \boldsymbol{W}'_2 - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{W}'_2}, \quad \boldsymbol{b}'_2 \leftarrow \boldsymbol{b}'_2 - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{b}'_2}, \end{aligned} \tag{17}$$

Algorithm 1 Recommendation via Dual-Autoencoders (ReDa)

Input: Given the rating or check-in matrix $\boldsymbol{R} \in \mathbb{R}^{m\times n}$, trade-off parameters α, β, γ, the number of latent factors, k.

Output: The prediction matrix $\hat{\boldsymbol{R}} = \boldsymbol{\xi}_2^\top \boldsymbol{\xi}_1$ for recommendation.

1. Initialize $\boldsymbol{W}_1$, $\boldsymbol{b}_1$, $\boldsymbol{W}'_1$, $\boldsymbol{b}'_1$ and $\boldsymbol{W}_2$, $\boldsymbol{b}_2$, $\boldsymbol{W}'_2$, $\boldsymbol{b}'_2$ by Stacked Autoencoders performed on $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$, respectively;
2. Compute the partial derivatives of all variables according to Eqs. (9)–(16);
3. Iteratively update the variables using Eq. (17);
4. Continue Step 2 and Step 3 until the algorithm converges;
5. Perform recommendation by the output $\hat{\boldsymbol{R}}$.

where η is the step size, which determines the speed of convergence. The pseudo codes of the proposed algorithm are summarized in Algorithm 1. Note that the proposed optimization problem is not convex, and thus there is no guarantee of obtaining an optimal global solution. To achieve a better local optimal solution of the proposed gradient descent approach, we first run Stacked Autoencoders (SAE) on $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ respectively, and then use their outputs to initialize the corresponding encoding and decoding weights. Also, it is worth mentioning that the rating matrix is normalized as $\boldsymbol{R} = \frac{\boldsymbol{R}}{r_{\max}}$, where $r_{\max}$ is the maximal rating score in $\boldsymbol{R}$, since the input value range of autoencoder should be [0, 1]. After Algorithm 1 is finished, $\hat{\boldsymbol{R}}$ is recovered to the original rating value range, i.e., $\hat{\boldsymbol{R}} = \hat{\boldsymbol{R}} \times r_{\max}$ for computing the evaluation measures Mean Absolute Error (MAE) and Root Mean Square Error (RMSE).

4. Experiments

In this section, we conduct experiments on four real-world data sets to systemically evaluate the effectiveness of our proposed framework for recommendation.

4.1. Data sets

Four data sets are used in the experiments for performance comparison among all the methods. Two of them are the famous

Table 2
Statistics of four data sets.

Dataset	No. of users	No. of items	No. of ratings	Rating density	Avg. ratings per user	Avg. ratings per item
MovieLens-100K	943	1682	100 000	6.3%	106.04	59.45
MovieLens-1M	6040	3706	1 000 209	4.47%	165.59	269.89
Douban Book	3399	2394	334 788	4.11%	98.50	139.84
Douban Movie	7292	6060	1 027 070	2.32%	140.85	169.48

recommendation data sets MovieLens 100K[1] and MovieLens 1M,[2] the former one contains 943 users and 1682 movies with 100 000 ratings, while the latter one contains 6040 users and 3706 movies with 1 000 209 ratings. The other two data sets are crawled from Douban, a well known social media network in China.[3] The Douban Book data set contains 3399 users and 2394 books with 334 788 ratings. The Douban Movie data set contains 7292 users and 6060 movies with 1 027 070 ratings. Ratings in these data sets scale from 1 to 5 stars. The detailed characteristics of these four data sets are summarized in Table 2. We can find that MovieLens has the most dense rating relation, while Douban Movie is the most sparse one.

4.2. Baselines and implementation details

Baseline methods Unlike most matrix factorization based recommendation methods, this work focuses on proposing a new latent factor model based on an autoencoder. Therefore, we first compare ReDa with several basic baselines, and then various kinds of state-of-the-art matrix factorization based methods, including

- UserMean: each rating is predicted by the average of the target user's available ratings on every item;
- ItemMean: each rating is predicted by the average of the target item's available ratings by all users;
- NMF (Lee & Seung, 2001): the basic Non-negative Matrix Factorization for recommendation;
- PMF (Mnih & Salakhutdinov, 2007): Probabilistic Matrix Factorization for recommendation;
- BPMF (Salakhutdinov & Mnih, 2008): Bayesian Probabilistic Matrix Factorization (BPMF) for recommendation;
- SVD++ (Koren, 2008)[4]: Merging the latent factor model and neighborhood model for recommendation;
- PRA (Liang & Baldwin, 2015): Probabilistic Rating Auto-encoder, which uses autoencoder to generate latent user feature profiles.

Implementation details The baselines NMF, PMF and BPMF are implemented by the Toolkit PREA.[5] After some preliminary test, the trade-off parameters α, β and γ are set as 0.5, 0.5 and 1 respectively. The number of latent factors k is set as 50 for all baselines.

Evaluation metrics We use Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) to evaluate the prediction performance of all algorithms.

$$\text{MAE} = \frac{\sum_{(u,i)\in \boldsymbol{R}} |\boldsymbol{R}_{u,i} - \hat{\boldsymbol{R}}_{u,i}|}{|\boldsymbol{R}|}, \tag{18}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{(u,i)\in \boldsymbol{R}} (\boldsymbol{R}_{u,i} - \hat{\boldsymbol{R}}_{u,i})^2}{|\boldsymbol{R}|}}, \tag{19}$$

where $\boldsymbol{R}$ denotes the whole rating matrix, $\boldsymbol{R}_{u,i}$ denotes the rating user u gives to item i, and $\hat{\boldsymbol{R}}_{u,i}$ denotes the rating user u gives to item i as prediction. Smaller values of MAE and RMSE mean better performance.

4.3. Experimental results

For each data set, 60%, 70%, 80%, 90% data are randomly sampled for training respectively, and the rest are used for test. We conduct five independent trials for each sampling setting, and their average values and standard deviation are both reported. All the results of four data sets on MAE and RMSE are listed in Tables 3 and 4. From these results, we have the following insightful observations,

- Generally, the performance of all algorithms becomes better when the sampling ratio of training data increases. The matrix factorization based methods PMF, BPMF and SVD++ are significantly affected by the sampling ratios of training data.
- ReDa is significantly better than the basic algorithms UserMean and ItemMean, which shows the effectiveness of the proposed model.
- Overall, ReDa outperforms all the matrix factorization based recommendation models NMF, PMF, BPMF and SVD++ (except in two cases, SVD++ is better than ReDa), which indicates that ReDa can find better latent factors using autoencoder technique. Only when the sampling ratios for training data are higher than 70%, NMF and PMF can perform better than UserMean and ItemMean. While the performance of ReDa is better than all the baselines even when sampling ratio for training data is only 60%.
- ReDa and PRA are both better than other approaches except SVD++, which shows the effectiveness of applying autoencoder technique. ReDa also outperforms PRA, which may indicate the superiority of simultaneously learning the users' and items' latent factors and minimizing reconstruction errors over training data. SVD++ is better than PRA, which may due to the consideration of neighborhood model.
- From the overview of these results, all the approaches can achieve better performance when the ratings become denser. It also seems that ReDa achieves larger improvement compared with NMF when the rating density becomes larger. On the most parse data set Douban Movie, SVD++ is better than ReDa, which may indicate that ReDa is more suitable for more dense data.

4.4. Parameter sensitivity

In this section, we investigate the influence of the parameters α, β and γ in the objective Eq. (7) on the Movielens data set with sampling 80% data for training. ReDa can perform stable when k is not too small or too large, e.g., $k \in [30, 100]$, thus we simply set $k = 50$ for all algorithms. In this experiment, when tuning one parameter, the values of the other two are fixed. Specifically, all three parameters α, β and γ are sampled from $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10\}$, and we report all the results in Fig. 2. From Fig. 2, we set $\alpha = 0.5$, $\beta = 0.5$ and $\gamma = 1$ as the default values for all four data sets.

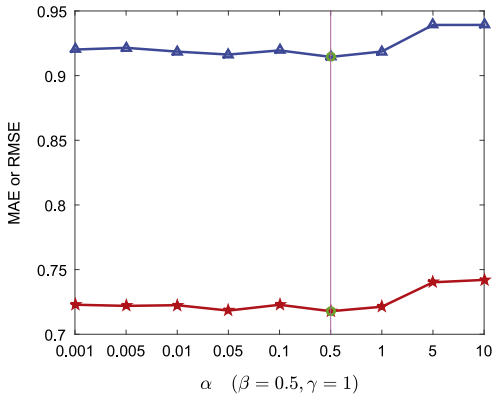
[1] http://files.grouplens.org/datasets/movielens/ml-100k.zip.
[2] http://files.grouplens.org/datasets/movielens/ml-1m.zip.
[3] http://www.douban.com.
[4] The code of SVD++ is downloaded from http://www.librec.net/download.html.
[5] Personalized Recommendation Algorithms Toolkit (PREA) http://prea.gatech.edu/index.html.

Table 3
The performance comparison among UserMean, ItemMean, NMF, PMF, BPMF, SVD++, PRA and ReDa on MovieLens-100K and MovieLens-1M.

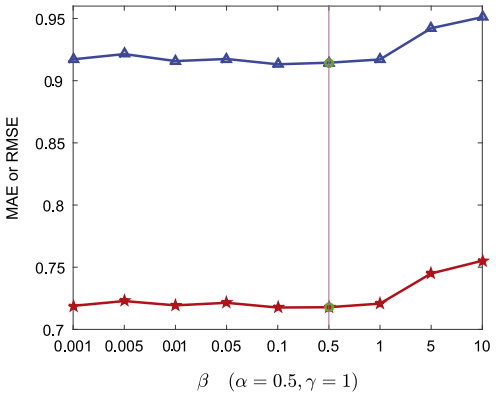
Data set	Setting	Metric	UserMean	ItemMean	NMF	PMF	BPMF	SVD++	PRA	ReDa
MovieLens-100K	60%	MAE	0.8384 ±0.0017	0.8209 ±0.0016	0.7655 ±0.0014	0.8395 ±0.0822	0.9650 ±0.0157	0.7522 0.0006	0.7656 ±0.0031	**0.7332** ±0.0047
		RMSE	1.0465 ±0.0021	1.0293 ±0.0008	0.9702 ±0.0023	1.0253 ±0.0780	1.2319 ±0.0220	0.9652 ±0.0001	0.9753 ±0.0039	**0.9329** ±0.0053
	70%	MAE	0.8342 ±0.0031	0.8162 ±0.0034	0.7599 ±0.0039	0.7919 ±0.0407	0.9040 ±0.0048	0.7400 ±0.0005	0.7632 ±0.0027	**0.7248** ±0.0067
		RMSE	1.0401 ±0.0028	1.0236 ±0.0042	0.9620 ±0.0043	0.9787 ±0.0403	1.1532 ±0.0068	0.9502 ±0.0006	0.9710 ±0.0058	**0.9231** ±0.0081
	80%	MAE	0.8344 ±0.0043	0.8163 ±0.0041	0.7582 ±0.0054	0.7823 ±0.0228	0.8810 ±0.0094	0.7260 ±0.0005	0.7594 ±0.0041	**0.7203** ±0.0043
		RMSE	1.0408 ±0.0046	1.0243 ±0.0050	0.9615 ±0.0075	0.9701 ±0.0266	1.1274 ±0.0118	0.9318 ±0.0009	0.9657 ±0.0042	**0.9190** ±0.0056
	90%	MAE	0.8370 ±0.0083	0.8179 ±0.0093	0.7659 ±0.0077	0.7882 ±0.0280	0.8626 ±0.0129	0.7222 ±0.0021	0.7595 ±0.0040	**0.7153** ±0.0094
		RMSE	1.0425 ±0.0081	1.0232 ±0.0122	0.9665 ±0.0093	0.9750 ±0.0304	1.1032 ±0.0162	0.9240 ±0.0005	0.9649 ±0.0052	**0.9114** ±0.0093
MovieLens-1M	60%	MAE	0.7822 ±0.0033	0.7827 ±0.0008	0.7372 ±0.0001	0.7055 ±0.0035	0.6962 ±0.0001	**0.6782** ±0.0001	0.7108 ±0.0010	0.6789 ±0.0040
		RMSE	0.9791 ±0.0044	0.9803 ±0.0013	0.9438 ±0.0001	0.8998 ±0.0038	0.8946 ±0.0003	**0.8656** ±0.0003	0.9002 ±0.0010	0.8659 ±0.0042
	70%	MAE	0.7820 ±0.0031	0.7828 ±0.0009	0.7282 ±0.0001	0.6971 ±0.0020	0.6882 ±0.0001	0.6736 ±0.0003	0.7122 ±0.0005	**0.6731** ±0.0029
		RMSE	0.9795 ±0.0043	0.9805 ±0.0016	0.9314 ±0.0002	0.8891 ±0.0028	0.8832 ±0.0001	0.8586 ±0.0003	0.9026 ±0.0005	**0.8573** ±0.0033
	80%	MAE	0.7809 ±0.0021	0.7825 ±0.0018	0.7214 ±0.0008	0.6900 ±0.0019	0.6802 ±0.0001	0.6680 ±0.0005	0.7142 ±0.0001	**0.6646** ±0.0029
		RMSE	0.9777 ±0.0028	0.9799 ±0.0014	0.9214 ±0.0013	0.8805 ±0.0027	0.8738 ±0.0001	0.8508 ±0.0004	0.9053 ±0.0002	**0.8485** ±0.0024
	90%	MAE	0.7825 ±0.0021	0.7810 ±0.0022	0.7180 ±0.0015	0.6849 ±0.0031	0.6758 ±0.0009	0.6656 ±0.0008	0.7155 ±0.0001	**0.6647** ±0.0051
		RMSE	0.9794 ±0.0026	0.9782 ±0.0025	0.9164 ±0.0012	0.8748 ±0.0061	0.8688 ±0.0003	0.8478 ±0.0019	0.9071 ±0.0001	**0.8474** ±0.0046

Table 4
The performance comparison among UserMean, ItemMean, NMF, PMF, BPMF, SVD++, PRA and ReDa on Douban Book and Douban Movie.

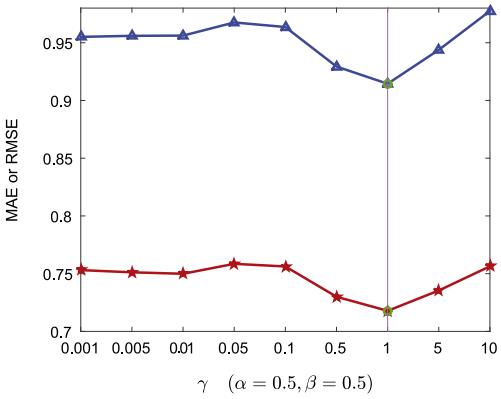
Data set	Setting	Metric	UserMean	ItemMean	NMF	PMF	BPMF	SVD++	PRA	ReDa
Douban Book	60%	MAE	0.6182 ±0.0015	0.5999 ±0.0004	0.5713 ±0.0014	0.5938 ±0.0143	0.6609 ±0.0014	0.5590 ±0.0005	0.5580 ±0.0011	**0.5548** ±0.0008
		RMSE	0.7726 ±0.0021	0.7410 ±0.0010	0.7171 ±0.0020	0.7651 ±0.0338	0.8326 ±0.0021	0.7082 ±0.0001	0.7011 ±0.0012	**0.6979** ±0.0014
	70%	MAE	0.6169 ±0.0012	0.6001 ±0.0006	0.5701 ±0.0008	0.5968 ±0.0173	0.6401 ±0.0010	0.5522 ±0.0009	0.5557 ±0.0011	**0.5500** ±0.0021
		RMSE	0.7717 ±0.0012	0.7408 ±0.0008	0.7159 ±0.0010	0.7712 ±0.0367	0.8073 ±0.0017	0.6982 ±0.0009	0.6983 ±0.0016	**0.6906** ±0.0018
	80%	MAE	0.6158 ±0.0010	0.5988 ±0.0012	0.5686 ±0.0017	0.5860 ±0.0096	0.6433 ±0.0008	0.5468 ±0.0004	0.5553 ±0.0018	**0.5485** ±0.0046
		RMSE	0.7704 ±0.0015	0.7395 ±0.0018	0.7142 ±0.0021	0.7489 ±0.0227	0.8121 ±0.0017	0.6916 ±0.0003	0.6974 ±0.0029	**0.6888** ±0.0039
	90%	MAE	0.6167 ±0.0027	0.5992 ±0.0018	0.5680 ±0.0025	0.5842 ±0.0033	0.6287 ±0.0013	0.5430 ±0.0010	0.5550 ±0.0037	**0.5415** ±0.0043
		RMSE	0.7721 ±0.0029	0.7402 ±0.0024	0.7145 ±0.0029	0.7431 ±0.0059	0.7941 ±0.0013	0.6868 ±0.0006	0.6967 ±0.0039	**0.6813** ±0.0031
Douban Movie	60%	MAE	0.6809 ±0.0008	0.6116 ±0.0010	0.5739 ±0.0006	0.6428 ±0.0497	0.6396 ±0.0066	0.5588 ±0.0004	0.5677 ±0.0004	**0.5537** ±0.0048
		RMSE	0.8521 ±0.0012	0.7635 ±0.0010	0.7261 ±0.0008	0.8015 ±0.0620	0.8157 ±0.0092	0.7082 ±0.0001	0.7189 ±0.0007	**0.7018** ±0.0049
	70%	MAE	0.6799 ±0.0007	0.6109 ±0.0007	0.5730 ±0.0008	0.6365 ±0.0314	0.6274 ±0.0052	0.5548 ±0.0001	0.5672 ±0.0004	**0.5513** ±0.0031
		RMSE	0.8511 ±0.0007	0.7628 ±0.0008	0.7249 ±0.0011	0.7938 ±0.0394	0.8003 ±0.0074	0.7012 ±0.0001	0.7181 ±0.0005	**0.6984** ±0.0026
	80%	MAE	0.6802 ±0.0008	0.6110 ±0.0005	0.5721 ±0.0012	0.6414 ±0.0481	0.6181 ±0.0020	**0.5486** ±0.0002	0.5662 ±0.0007	0.5513 ±0.0029
		RMSE	0.8510 ±0.0012	0.7627 ±0.0006	0.7239 ±0.0011	0.7996 ±0.0596	0.7886 ±0.0027	**0.6940** ±0.0005	0.7164 ±0.0009	0.6981 ±0.0030
	90%	MAE	0.6798 ±0.0021	0.6095 ±0.0016	0.5705 ±0.0013	0.6423 ±0.0503	0.6064 ±0.0044	**0.5452** ±0.0000	0.5654 ±0.0013	0.5545 ±0.0045
		RMSE	0.8506 ±0.0025	0.7605 ±0.0018	0.7220 ±0.0017	0.8008 ±0.0633	0.7734 ±0.0056	**0.6925** ±0.0002	0.7155 ±0.0022	0.7011 ±0.0043



(a) The parameter influence of α.



(b) The parameter influence of β.



(c) The parameter influence of γ.

Fig. 2. The study of parameter influence on ReDa.

5. Related work

Recent years have witnessed a boom of research work in recommendation systems. A number of techniques are employed for recommendations and many sources of data are fused to improve recommendation performances. Traditional recommender systems normally only utilize user–item rating feedback information for recommendation. Collaborative filtering is one of the most popular techniques, whose basic idea is to find similar objects for recommendation through interactive records. Recently, matrix factorization has shown its effectiveness and efficiency in recommender systems, which factorizes user–item rating matrix into two low rank user-specific and item-specific matrices, and then utilizes the factorized matrices to make further predictions (Srebro & Jaakkola, 2003). Assuming that the rating matrix is low-rank within certain neighborhoods of the metric space, Lee, Bengio, Kim, Lebanon, and Singer (2014) combined a recent approach for local low-rank approximation based on the Frobenius norm with a general empirical risk minimization for ranking losses. Actually, this method is an ensemble of basic matrix factorization, and of course we can use it to further improve our model. In this work, we aim to propose a new latent factor framework based on deep learning.

With the prevalence of social media, more and more information is fused for better recommendation. Many researchers study social recommender systems which utilize social relations among users. Ma et al. (2008) fused user–item matrix with users' social trust network by sharing a common latent low-dimensional user feature matrix. Furthermore, authors in Ma, King, and Lyu (2009) coined the social trust ensemble to represent the formulation of the social trust restrictions. Some researchers have begun to use friend relation among users. For example, Yang, Steck, and Liu (2012) inferred category-specific social trust circles from available rating data combined with friend relations. With the surge of heterogeneous information network, some researchers have noticed the importance of heterogeneous information for recommendation. Yu et al. (2014) proposed an implicit feedback recommendation model with systematically extracted latent features from heterogeneous network. More recently, Shi et al. (2015) utilized heterogeneous network to integrate all kinds of information, and adopt a meta path based similarity measure for semantic recommendation. Most of these methods integrate additional information under the matrix factorization framework, in which the latent factors of users and items are usually obtained by directly factorizing the rating matrix. However, in ReDa, we try to propose that the latent factors of users and items can be derived by the autoencoders from the rating matrix.

Recently, deep learning has shown its great success in image and text processing. Some researchers have noticed the potential of deep learning for recommendation. In order to alleviate the cold start problem, Oord et al. (2013) extracted latent factors of music from audio signals with deep convolutional neural network, instead of traditional bag-of-words representation. To address the sparsity problem, Wang et al. (2014) employed the collaborative topic regression model to utilize item content information. In the model, they designed a collaborative deep learning method to couple a Bayesian formulation of the stacked denoising autoencoder and probabilistic matrix factorization. In essence, these methods utilize deep learning to extract latent features from additional information, e.g., music audio and text

content. Unlike these methods, the proposed ReDa employs the deep learning directly on the rating information. The most related work is Liang and Baldwin (2015), in which they proposed a probabilistic rating autoencoder to generate latent user feature profiles, then the neighborhood based collaborative filtering approach is used to make predictions. However, they did not consider the reconstruction error over latent factors, which may further improve the performance.

6. Conclusion and future work

To make full use of the user–item rating information and learn better latent representations, different from previous matrix factorization methods we aim to propose a new representation learning model based on autoencoders for recommendation in this paper. In our proposed framework, we simultaneously learn the latent factors of users and items, and minimize the derivations of training data using the learnt latent factors. Experiments on four data sets validate the superiority of our proposed framework.

Though our proposed recommendation framework ReDa can achieve promising performance, for each iteration in Algorithm 1, the time complexity is $O(kmn)$, where k is the number of latent factors, m is the number of users and n is the number of items. So we will focus on developing an efficient algorithm to derive the solutions in the future. Moreover, it is easy to integrate external information by introducing some regularization items for our model, which will also be our future work.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Nos. 61473273, 91546122, 61573335, 61602438), Guangdong provincial science and technology plan projects (No. 2015 B010109005), the Youth Innovation Promotion Association CAS 2017146 and 2015 Microsoft Research Asia Collaborative Research Program.

References

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, *177*, 734–749.

Alex, Graves, Marcus, Liwicki, Santiago, Fernández, Roman, Bertolami, Horst, Bunke, & J̈'urgen, Schmidhuber (2009). A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(5), 855–868.

Bengio, Yoshua (2009). Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, *2*(1), 1–127.

Koren, Yehuda (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 426–434). ACM.

Koren, Y., & Bell, R. (2011). *Advances in collaborative filtering*, *Recommender systems handbook* (pp. 145–186). Springer, (Chapter) Number.

Koren, Yehuda, Bell, Robert, & Volinsky, Chris (2009). Matrix factorization techniques for recommender systems. *Computer*, *8*, 30–37.

Lee, J., Bengio, S., Kim, S., Lebanon, G., & Singer, Y. (2014). Local collaborative ranking for recommendation. In *Proceedings of the 23rd international world wide web conference (WWW)*.

Lee, Daniel D., & Seung, H. Sebastian (2001). Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems* (pp. 556–562).

Lian, Defu, Zhao, Cong, Xie, Xing, Sun, Guangzhong, Chen, Enhong, & Rui, Yong (2014). Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation. In *KDD*, (pp. 831–840).

Liang, Huizhi, & Baldwin, Timothy (2015). A probabilistic rating auto-encoder for personalized recommender systems. In *Proceedings of the 24th ACM international on conference on information and knowledge management* (pp. 1863–1866). ACM.

Liu, Bin, Fu, Yanjie, Yao, Zijun, & Xiong, Hui (2013). Learning geographical preferences for point-of-interest recommendation. In *KDD* (pp. 1043–1051).

Ma, Hao, King, Irwin, & Lyu, Michael R. (2009). Learning to recommend with social trust ensemble. In *Proceedings of the 32nd international ACM SIGIR conference on research and development in information retrieval* (pp. 203–210). ACM.

Ma, Hao, Yang, Haixuan, Lyu, Michael R., & King, Irwin (2008). Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on information and knowledge management*, (pp. 931–940).

Ma, Hao, Zhou, Tom Chao, Lyu, Michael R., & King, Irwin (2011). Improving recommender systems by incorporating social contextual information. *ACM Transactions on Information Systems (TOIS)*, *29*(2), 9.

Mnih, Andriy, & Salakhutdinov, Ruslan (2007). Probabilistic matrix factorization. In *Advances in neural information processing systems* (pp. 1257–1264).

Oord, Aaron Van den, Dieleman, Sander, & Schrauwen, Benjamin (2013). Deep content-based music recommendation. In *Advances in Neural Information Processing Systems* (pp. 2643–2651).

Salakhutdinov, Ruslan, & Mnih, Andriy (2008). Bayesian probabilistic matrix factorization using Markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, (pp. 880–887).

Salakhutdinov, Ruslan, Mnih, Andriy, & Hinton, Geoffrey (2007). Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning* (pp. 791–798). ACM.

Shi, Chuan, Zhang, Zhiqiang, Luo, Ping, Yu, Philip S., Yue, Yading, & Wu, Bin (2015). Semantic path based personalized recommendation on weighted heterogeneous information networks. In *The 24th ACM international conference on information and knowledge management (CIKM)*.

Socher, R., Huang, E. H., Pennington, J., Ng, A. Y., & Manning, C. D. (2011). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*.

Srebro, Nathan, & Jaakkola, Tommi et al. (2003). Weighted low-rank approximations. In *International conference on machine learning*, Vol. 3, (pp. 720–727).

Wang, Hao, Wang, Naiyan, & Yeung, Dit-Yan (2014). Collaborative deep learning for recommender systems. arXiv preprint arXiv:1409.2944.

Yang, Xiwang, Steck, Harald, & Liu, Yong (2012). Circle-based recommendation in online social networks. In *Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1267–1275). ACM.

Yu, Xiao, Ren, Xiang, Sun, Yizhou, Gu, Quanquan, Sturt, Bradley, Khandelwal, Urvashi, et al. (2014). Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM international conference on web search and data mining* (pp. 283–292). ACM.