A Recommendation Model Based on Deep Neural Network

Libo Zhang, Tiejian Luo*, Fei Zhang and Yanjun Wu

Abstract—In recent years, recommendation systems have been widely used in various commercial platforms to provide recommendations for users. Collaborative filtering algorithms are one of the main algorithms used in recommendation systems. Such algorithms are simple and efficient; however, the sparsity of the data and the scalability of the method limit the performance of these algorithms, and it is difficult to further improve the quality of the recommendation results. Therefore, a model combining a collaborative filtering recommendation algorithm with deep learning technology is proposed, therein consisting of two parts. First, the model uses a feature representation method based on a quadric polynomial regression model, which obtains the latent features more accurately by improving upon the traditional matrix factorization algorithm. Then, these latent features are regarded as the input data of the deep neural network model, which is the second part of the proposed model and is used to predict the rating scores. Finally, by comparing with other recommendation algorithms on three public datasets, it is verified that the recommendation performance can be effectively improved by our model.

Index Terms—Recommendation system, Collaborative Filtering, Quadric Polynomial Regression, Deep Neural Network

I. Introduction

WITH the development of artificial intelligence technology, increasingly more intelligent products are being applied in daily life and provide convenience for people in various aspects. The intelligent recommendation function of personalized recommendation systems can effectively provide users with valuable information from massive Internet data; thus, it is widely used in many network platforms such as movie, music and shopping platforms.

The recommendation algorithm is the most important part of a recommendation system and directly determines the quality of the recommendation results and the performance of the system. The commonly used algorithms can be divided into two main categories: content-based[1] methods and collaborative filtering[2], [3], [4] methods. Content-based methods construct portraits of users and items through the analysis of extra information, such as document content, user profiles and the attributes of items, to make recommendations. In most cases, the information that is used to construct the portraits is difficult to obtain or even fake; therefore, its performance and application range suffer from significant limitations. Collaborative filtering algorithms are the most widely used algorithms in recommendation systems; they are different from content-based methods in that they do not require information about users or items, and they make accurate recommendations based only on interaction information between users and items such as clicks, browsing and rating. Although this method is simple and effective, with the rapid development of the Internet, the high sparsity of the data limits the performance of the algorithm; therefore, researchers have begun to look for other methods of improving the recommendation performance.

In recent years, Deep Neural Networks (DNNs) have achieved great success in various fields, such as computer vision[5], speech recognition[6] and natural language processing[7]; however, there are few studies on recommendation systems with these technologies. Some researchers have recently proposed recommendation models based on deep learning, but most of these models use additional features, such as text content and audio information, to enhance their performances. Considering that the abovementioned information may be difficult to obtain for most recommendation systems, in this paper, we propose a recommendation model based on DNNs that does not need any extra information aside from the interaction between users and items. The main framework of our model is shown in Figure 1. First, we use the user-item rating matrix to obtain the features of the users and items, which we will discuss in Section 3. Then, we regard these features as the input of the neural network. In the output layer, we will obtain some probability values that represent the probabilities of the scores that the user might give. Finally, the score with the highest probability will be used as the prediction result. By comparing with some commonly used and state-of-the-art algorithms on three public datasets, it is proved that the proposed model can effectively improve the recommendation accuracy.

The remainder of this paper is organized as follows: In Section 2, we introduce the CF methods and some recommendation algorithms based on DNNs. We will give a detailed description of our model in Section 3. Section 4 contains some experimental evaluations and discussion. We provide a brief conclusion in Section 5.

II. Related Work

Breese et al.[8] divide the CF algorithm into two classes: memory-based methods and model-based methods. The

Libo Zhang and Yanjun Wu are with Institute of Software, Chinese Academy of Sciences, Beijing 100190, China. Tiejian Luo and Fei Zhang are with the University of Chinese Academy of Science, Beijing 101408, China.

Tiejian Luo is the corresponding author.

Manuscript received XXXX; revised XXXX.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2789866, IEEE Access

memory-based CF uses the similarities between users[9] or items[10] to make recommendations. This method is widely used because it is effective and easy to implement, but with the increase in the scale of the recommendation system, the calculation of the similarity becomes increasingly more difficult; in addition, high data sparsity also limits the performance of this method.

To solve the above-mentioned problems, many model-based recommendation algorithms have been proposed such as latent semantic models[11], Bayesian models[12], regression-based models[13], clustering models[14], and matrix factorization models[15]. Among the various CF technologies, matrix factorization is the most popular method. This method maps both users and items to vectors with the same dimension, which represents the latent features of the users or items. The representative works of this method include Nonparametric Probabilistic Principal Component Analysis (NPCA)[16], Singular Value Decomposition (SVD)[17], and Probabilistic Matrix Factorization (PMF)[18]. However, the latent features learned by matrix factorization methods are often not sufficiently effective, especially when the rating matrix is very sparse.

On the other hand, deep learning techniques have recently achieved great success in the computer vision and natural language processing fields. Such techniques show great potential in learning feature representations; therefore, researchers have begun to apply deep learning methods to the field of recommendations. Salakhutdinov et al.[19] use a restricted Boltzmann machine instead of the traditional matrix factorization to perform the CF, and Georgiev et al.[20] expanded the work by incorporating the correlation between users and between items. There are also other studies that have proposed methods based on deep learning, but they mainly focus on music recommendation, such as [21] and [22]. These studies use traditional convolutional neural networks and deep trust networks, respectively, to learn the music content features. In addition to music recommendation, Wang et al.[23] propose a hierarchical Bayesian model that uses a deep learning model to obtain content features and a traditional CF model to address the rating information. As we can see, these methods based on deep learning techniques more or less make recommendations by learning the content features of items such as text content and the spectrum of music. These methods are not applicable when we are unable to obtain the contents of items. Therefore, He et al.[24] propose a new recommendation framework based on deep learning. In their method, users and items are represented via one-hot encoding of their ID; obviously, this method only uses the ID information during the training phase of the model, which makes a large amount

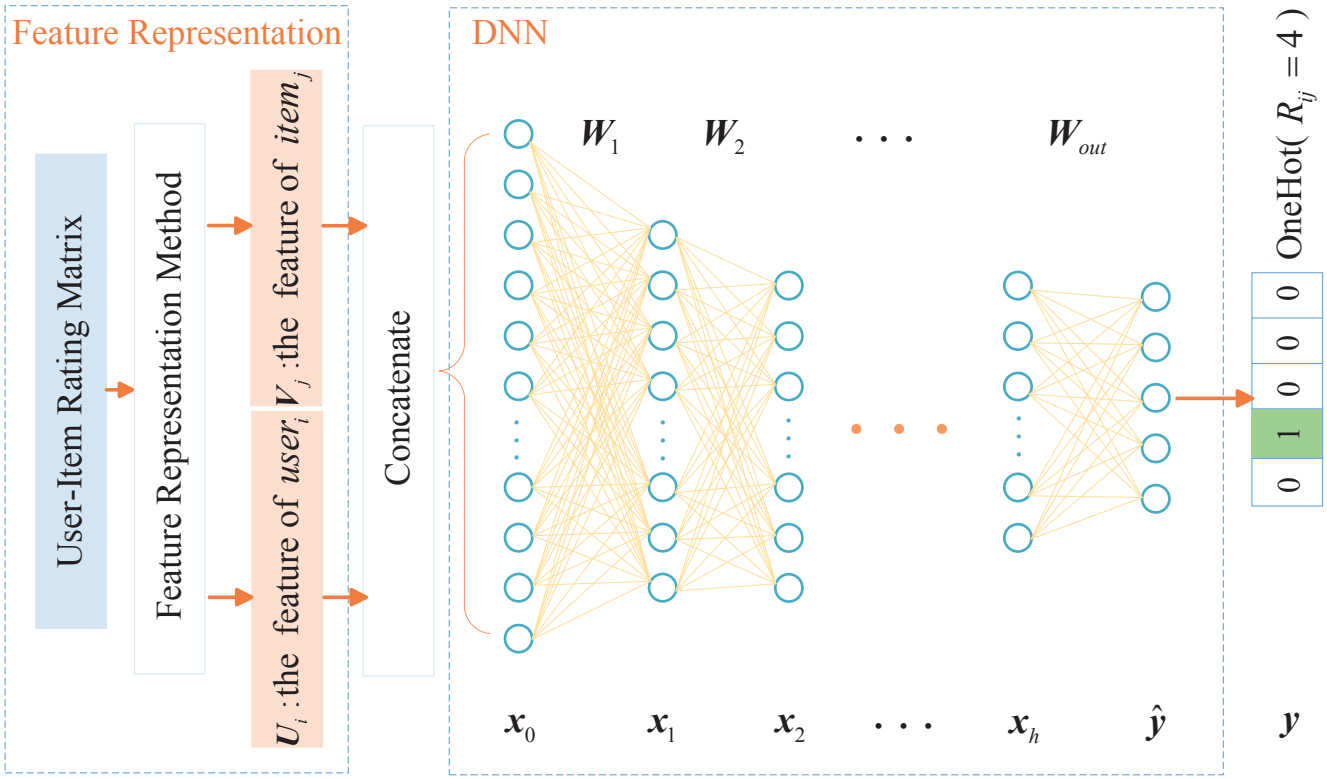

Fig. 1. The framework of the proposed model includes two steps: first, the features of users and items are obtained from the rating matrix; then, the two features are concatenated together as the input data for the DNN model. After training the DNN, the probability distribution of the rating score will be obtained from the output layer.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2789866, IEEE Access

of prior information unable to be used. Therefore, the effectiveness of feature learning is difficult to guarantee.

III. Our Algorithm

A. Feature Representation Model

As seen in Figure 1, in our method, we need to obtain the features of users and items according to the rating matrix. In this section, we will discuss various feature representation methods.

Let the user-item rating matrix of n users to m items be $\boldsymbol{R} \in \mathbb{R}^{n\times m}$. The entry R_{ij} represents the i-th user's rating score of the j-th item; if the rating record does not exist, then $R_{ij} = 0$. Moreover, we let the users' latent feature matrix be $\boldsymbol{U} \in \mathbb{R}^{n\times a}$, the vector $\boldsymbol{U}_i$ of the i-th row represents the features of the i-th user, and a represents the dimension of the features. Likewise, the latent features of the items are represented by the matrix $\boldsymbol{V} \in \mathbb{R}^{m\times b}$.

1) Ratings as Features: Ratings as Features (RaF) is a feature representation method whose main idea is that a user's rating data are regarded as the feature of the user directly. Specifically, $\boldsymbol{U} = \boldsymbol{R}$, and likewise, the features of items are $\boldsymbol{V} = \boldsymbol{R}^{\mathrm{T}}$.

Although the RaF method is simple and effective, it also has some shortcomings. The rating matrix is highly sparse; thus, there are many missing data. If one does not address the missing data carefully, the resulting model may not be sufficiently accurate, and finding reasonable methods to address such data may require much additional work.

2) ID as Features: The ID of a user or item is unique; therefore, it can be considered as our desired feature. However, the ID is a categorical variable that cannot be compared or summed or subject to other mathematical operations; therefore, He et al[24] propose a method called Neural Collaborative Filtering (NCF). The framework of this method is shown in Figure 2.

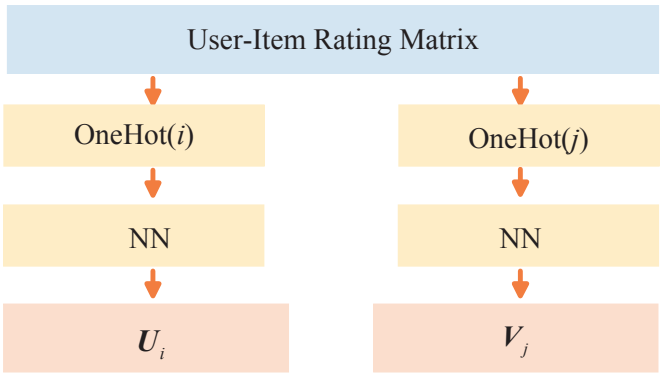

Fig. 2. The feature representation method of the NCF model. First, use the One-Hot encoding method to encode the ID of the users and items; then, use two additional neural network models to train them to obtain the features.

According to figure 2, we have

$$\boldsymbol{U}_i = \mathrm{NN}(\mathrm{OneHot}(i)), \tag{1}$$

$$\boldsymbol{V}_j = \mathrm{NN}(\mathrm{OneHot}(j)), \tag{2}$$

where $\mathrm{OneHot}(i)$ indicates using the One-Hot method to encode the ID i, which will generate a zero vector with a specified dimension, and the i-th position of the vector will be set to 1. $\mathrm{NN}(\boldsymbol{x})$ denotes the output of the neural network when the input is $\boldsymbol{x}$.

3) Singular Value Decomposition: The matrix factorization technique achieves a strong performance with decreasing dimensions and feature representations; therefore, it is a feasible choice when needing to learn features from a matrix.

The SVD algorithm is one of the most famous matrix factorization algorithms; this algorithm decomposes the matrix into three matrices, whereby we can obtain the features of users and items. The expression of SVD is as follows:

$$\boldsymbol{R} = \boldsymbol{U} \cdot \boldsymbol{S} \cdot \boldsymbol{V}^{\mathrm{T}}, \tag{3}$$

where the diagonal matrix $\boldsymbol{S} \in \mathbb{R}^{n\times m}$ and the main diagonal element consists of the eigenvalues of the matrix $\boldsymbol{R}$.

This method also needs to preprocess a large number of missing values; on the other hand, the performance is poor when addressing large-scale data due to its high complexity when solving for eigenvalues.

4) Probabilistic Matrix Factorization: The PMF algorithm is proposed by Mnih et al[18]. They use the product of the user's latent features and the item's features to fit the corresponding rating score. When the least square error is used as the loss function, the expression is as follows:

$$L = \frac{1}{2}||\boldsymbol{R} - \boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}||_F^2 + \frac{\lambda_1}{2}||\boldsymbol{U}||_F^2 + \frac{\lambda_2}{2}||\boldsymbol{V}||_F^2 \tag{4}$$

where the parameters λ_1 and λ_2 represent the L2 regular parameters. Because of the existence of missing values, which should not be considered when calculating losses, the modified loss function is as follows:

$$\begin{aligned} L =& \frac{1}{2}\sum_{i,j}\delta_{ij}||R_{ij} - \boldsymbol{U}_i{\boldsymbol{V}_j}^T||_F^2 + \\ & \frac{\lambda_1}{2}||\boldsymbol{U}||_F^2 + \frac{\lambda_2}{2}||\boldsymbol{V}||_F^2 \end{aligned} \tag{5}$$

where $\delta_{ij} = 0$ when $R_{ij} = 0$; otherwise, $\delta_{ij} = 1$.

It can be observed that the PMF algorithm can effectively avoid the problem of missing values; however, this method also has other limitations. The basic assumption of this algorithm is that the features of both users and items are completely independent, while in the field of recommendation, there may be some correlation between different features.

5) Quadric Polynomial Regression: Considering the above-mentioned disadvantages of the RaF, NCF, SVD and PMF methods, this paper proposes a new feature representation method based on Quadric Polynomial Regression (QPR), which not only avoids the issue whereby the preprocessing of missing values may lead to inaccurate results in feature learning but also can consider the correlations between features.

In the traditional quadric polynomial regression model, for the feature vector $\boldsymbol{x}$, the corresponding supervised value y is fitted by the following expression:

$$\hat{y} = z + \sum_{i=1}^{l} w_i x_i + \sum_{i=1}^{l-1} \sum_{j=i+1}^{l} W_{ij} x_i x_j, \tag{6}$$

where the parameter l represents the dimension of the vector $\boldsymbol{x}$, z represents the coefficient of the constant term, $\boldsymbol{w}$ represents the first-order coefficients, and $\boldsymbol{W} \in \mathbb{R}^{l \times l}$ represents the second-order coefficients. Therefore, in the proposed method, there are $\boldsymbol{x} = (\boldsymbol{U}_u, \boldsymbol{V}_v)$ and $y = R_{uv}$. Then, we obtain

$$\begin{aligned} \hat{R}_{uv} =& z + \sum_{i=1}^{a} w_i U_{ui} + \sum_{j=1}^{b} w_{j+a} V_{vj} + \\ & \sum_{i=1}^{a-1} \sum_{j=i+1}^{a} W_{ij} U_{ui} U_{uj} + \\ & \sum_{i=1}^{b-1} \sum_{j=i+1}^{b} W_{i+a,j+a} V_{vi} V_{vj} + \\ & \sum_{i=1}^{a} \sum_{j=1}^{b} W_{i,j+a} U_{ui} V_{vj}, \end{aligned} \tag{7}$$

If we set

$$p_u = \sum_{i=1}^{a} w_i U_{ui} + \sum_{i=1}^{a-1} \sum_{j=i+1}^{a} W_{ij} U_{ui} U_{uj}, \tag{8}$$

$$q_v = \sum_{j=1}^{b} w_{j+a} V_{vj} + \sum_{i=1}^{b-1} \sum_{j=i+1}^{b} W_{i+a,j+a} V_{vi} V_{vj}, \tag{9}$$

then it can be observed that p_u is only concerned with the user u, and q_v is only related to the item v; therefore, we obtain the final model:

$$\hat{R}_{uv} = z + p_u + q_v + \sum_{i=1}^{a} \sum_{j=1}^{b} W_{ij} U_{ui} V_{vj}, \tag{10}$$

Notice that we redefine $\boldsymbol{W} \in \mathbb{R}^{a \times b}$ here.

As with the PMF algorithm, the training loss of QPR is evaluated by the least square method. Thus, we have

$$L = \frac{1}{2} \sum_{u,v} \delta_{uv} (R_{uv} - \hat{R}_{uv})^2 \tag{11}$$

By optimizing Eq. (11) to minimize L, we can obtain the two matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ that we need.

B. Deep Neural Network Model

In this subsection, we will introduce the deep neural network model in figure 1, which takes the latent features of users and items as the inputs and uses the forward propagation algorithm to predict the rating scores.

According to our model, in the input layer, the input vector $\boldsymbol{x}_0$ is concatenated by the latent features of users and items; therefore, for any record R_{ij}, we have

$$\boldsymbol{x}_0 = concatenate(\boldsymbol{U}_i, \boldsymbol{V}_j), \tag{12}$$

where the function $concatenate()$ is used to concatenate two vectors. When $\boldsymbol{x}_0$ passes through the first hidden layer, the output of the first hidden layer is obtained by the following equation:

$$\boldsymbol{x}_1 = activation(\boldsymbol{W}_1 \boldsymbol{x}_0 + \boldsymbol{b}_1), \tag{13}$$

where $\boldsymbol{W}_1$ is the weight matrix between the input layer and the first hidden layer, $\boldsymbol{b}_1$ is the bias vector, the $activation()$ indicates the activation function, which is designed to make the neural network model nonlinear and multilayer neural networks become meaningful. In the DNN model, the activation functions that we use include the sigmoid, tanh, and ReLU functions. In this paper, we choose ReLU as the activation function for our model because it is more effective[25] and easier to optimize[26].

By Eq. (13) and the discussion above, we can obtain the output at the l-th hidden layer:

$$\boldsymbol{x}_l = \mathrm{ReLU}(\boldsymbol{W}_l \boldsymbol{x}_{l-1} + \boldsymbol{b}_l). \tag{14}$$

In the output layer, our training goal is to predict the user's rating score R_{ij}. We use the One-Hot encoding method again to obtain the supervised value $\boldsymbol{y} = \mathrm{OneHot}(R_{ij})$; therefore, we need to transform the output result by the softmax method to obtain the prediction value of the corresponding position of $\boldsymbol{y}$, that is,

$$\hat{\boldsymbol{y}} = \mathrm{softmax}(\boldsymbol{W}_{out} \boldsymbol{x}_h + \boldsymbol{b}_{out}), \tag{15}$$

where h represents the number of hidden layers, $\boldsymbol{x}_h$ is the output of the last hidden layer, and $\boldsymbol{W}_{out}$ and $\boldsymbol{b}_{out}$ represent the weight and bias of the output layer, respectively. Finally, we use a cross entropy method to evaluate the difference between the prediction result $\hat{\boldsymbol{y}}$ and the supervised value $\boldsymbol{y}$:

$$\varepsilon = -\sum_{i=1}^{d} (y_i ln(\hat{y}_i) + (1 - y_i) ln(1 - \hat{y}_i)), \tag{16}$$

where d represents the dimension of the vector $\boldsymbol{y}$, which is equivalent to the number of neurons in the output layer. Finally, our model predicts the i-th user's rating score on the j-th item by the following equation:

$$\hat{R}_{ij} = \arg\max_k (\hat{y}_k). \tag{17}$$

C. Training Model

In this subsection, we shall describe the training details of our model. First, we need to use the QPR model to obtain the features. We use the gradient descent method to solve Eq. (11), and the learning rate is set to η. Thus, the rules for updating each parameter are as follows:

$$z = z - \eta \sum_{u,v} \Delta_{uv}, \tag{18}$$

$$p_u = p_u - \eta \sum_{v} \Delta_{uv}, \tag{19}$$

$$q_v = q_v - \eta \sum_{u} \Delta_{uv}, \tag{20}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2789866, IEEE Access

$$W_{ij} = W_{ij} - \eta \sum_{u,v} \Delta_{uv} U_{ui} V_{vj}, \tag{21}$$

$$U_{ui} = U_{ui} - \eta \sum_{v} (\Delta_{uv} \sum_{j=1}^{b} W_{ij} V_{vj}), \tag{22}$$

$$V_{vj} = V_{vj} - \eta \sum_{u} (\Delta_{uv} \sum_{i=1}^{a} W_{ij} U_{ui}), \tag{23}$$

where

$$\Delta_{uv} = \delta_{uv}(\hat{R}_{uv} - R_{uv}). \tag{24}$$

The algorithm for training the feature representation model is shown in Algorithm 1.

Algorithm 1 Algorithm for Feature Representation

Require:
User-item rating matrix $\boldsymbol{R}$;
Parameters n, m, a, b, η;
Ensure:
Matrix $\boldsymbol{U}$, $\boldsymbol{V}$;

```
Randomly initialize z, p, q, U, V, W;
repeat
    Calculate Δ according to Eq.(24);
    Update z according to Eq.(18);
    for u = 1 to n do
        Update p_u according to Eq.(19);
        Update U_u according to Eq.(22);
    end for
    for v = 1 to m do
        Update q_v according to Eq.(20);
        Update V_v according to Eq.(23);
    end for
    Update W according to Eq.(21);
until convergence
```

Next, we will discuss the training methods of the deep neural network model. Through the analysis in Section 3.2, we can find that the training of the deep neural network is mainly based on the learning of the weight matrix $\boldsymbol{W}$ and the bias vector $\boldsymbol{b}$. We define $W_{l,ij}$ to represent the connection weight between the i-th neuron in the l-th layer and the j-th neuron in the $(l-1)$-th layer. In particular, the 0-th layer represents the input layer. $b_{l,i}$ represents the bias on i-th neuron in the l-th layer.

We use the gradient descent method to solve $\boldsymbol{W}$ according to Eq.(16); then, we have

$$W_{l,ij} = W_{l,ij} - \eta \frac{\partial \varepsilon}{\partial W_{l,ij}}, \tag{25}$$

Fig. 3 shows that $W_{l,ij}$ can only affect the loss ε through neuron i; therefore, we use $net_{l,i}$ to represent the weighted input of the neuron i in the l-th layer:

$$net_{l,i} = b_{l,i} + \sum_{j} W_{l,ij} x_{l-1,j}, \tag{26}$$

Then, we have

$$\frac{\partial \varepsilon}{\partial W_{l,ij}} = \frac{\partial \varepsilon}{\partial net_{l,i}} \frac{\partial net_{l,i}}{\partial W_{l,ij}} = \frac{\partial \varepsilon}{\partial net_{l,i}} x_{l-1,j}, \tag{27}$$

Next, we will discuss the value of $\frac{\partial \varepsilon}{\partial net_{l,i}}$ in two cases.

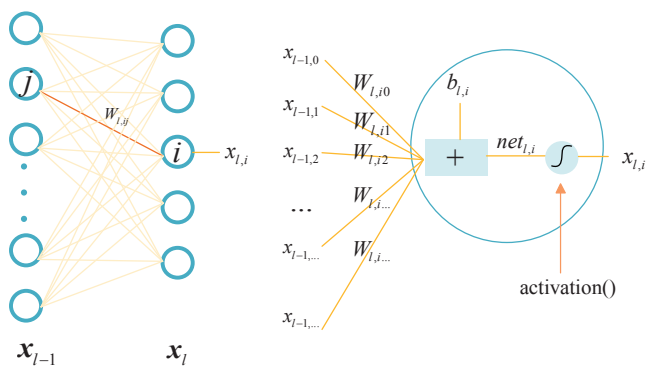

Fig. 3. The influence of $W_{l,ij}$ on the loss ε and the internal structure of the neuron i.

When layer l is the output layer,

$$\frac{\partial \varepsilon}{\partial net_{l,i}} = \frac{\partial \varepsilon}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial net_{l,i}}. \tag{28}$$

According to Eq. (16),

$$\frac{\partial \varepsilon}{\partial \hat{y}_i} = \frac{\hat{y}_i - y_i}{\hat{y}_i(1-\hat{y}_i)}, \tag{29}$$

and according to Eq. (15),

$$\frac{\partial \hat{y}_i}{\partial net_{l,i}} = \frac{\partial \text{softmax}(net_{l,i})}{\partial net_{l,i}} = \hat{y}_i(1-\hat{y}_i), \tag{30}$$

Taking Eq. (29) and Eq. (30) into Eq. (28), we obtain

$$\frac{\partial \varepsilon}{\partial net_{l,i}} = \hat{y}_i - y_i. \tag{31}$$

By the same analysis, when layer l is a hidden layer,

$$\frac{\partial \varepsilon}{\partial net_{l,i}} = \sum_{k} \frac{\partial \varepsilon}{\partial net_{l+1,k}} W_{l+1,ki}. \tag{32}$$

Thus, according to Eq. (27), Eq. (31) and Eq. (32), we obtain the update rules for $\boldsymbol{W}$ of each layer.

We analyze the update rules for $\boldsymbol{b}$ in the same way, and from Eq. (26) we have

$$\frac{\partial net_{l,i}}{\partial b_{l,i}} = 1. \tag{33}$$

Therefore,

$$\frac{\partial \varepsilon}{\partial b_{l,i}} = \frac{\partial \varepsilon}{\partial net_{l,i}} \frac{\partial net_{l,i}}{\partial b_{l,i}} = \frac{\partial \varepsilon}{\partial net_{l,i}}. \tag{34}$$

Based on the above discussion, we obtain the following rules for updating the parameters in the DNN model:

$$b_{l,i} = b_{l,i} - \eta \frac{\partial \varepsilon}{\partial net_{l,i}}, \tag{35}$$

$$W_{l,ij} = W_{l,ij} - \eta \frac{\partial \varepsilon}{\partial net_{l,i}} x_{l-1,j}, \tag{36}$$

where $\frac{\partial \varepsilon}{\partial net_{l,i}}$ is determined by Eq.(31) and Eq.(32).

D. Making Recommendations

When the training of the proposed model is completed, we can use it to predict a user's rating score on the items that have not been rated by the user. When making recommendations for a specific user, we can recommend the items with the highest predicted score for the user.

IV. Experimental Evaluation

A. Data Description

In our experiment, we use three real datasets to test the performance of our model: MovieLens-100K, MovieLens-1M, and Epinions.

The MovieLens-100K dataset contains nearly 100,000 rating records of 943 users on 1,682 items; the dataset comes from the MovieLens website, and all the rating scores are positive and not greater than 5.

The MovieLens-1M dataset also comes from the MovieLens website, but it contains 1,000,209 rating records from 6,040 users for 3,952 movies. In addition, it was released later than the previous database, and each user has rated at least 20 movies.

The Epinions dataset is from the Epinions website. Before our experiments, the users who have rated fewer than 10 items are removed from the dataset, and the items that have been rated fewer than 10 times are also removed. Ultimately, 354,857 rating records of 15,687 users on 11,657 items remain.

The statistics of the three datasets are shown in Table 1.

TABLE I
The Statistics of the Three Datasets

	MovieLens-100K	MovieLens-1M	Epinions
# of users	943	6,040	15,687
# of items	1,682	3,952	11,657
# of ratings	100,000	1,000,209	354,857
# of ratings per user	106.4	165.6	22.62
# of ratings per item	59.45	253.09	30.44
Rating Sparsity	93.70%	95.81%	99.81%

B. Experimental Environment

1) Hardware: Two pieces of GPU were used in this experiment, they are all NVIDIA TITAN Xp, and the CPU is Intel Core i7-7500U.

2) Software: The operating system used in this experiment is Ubuntu 16.04, and we use Python language to achieve our program, the specific version is 3.5. Since the deep learning method is used in this paper, we adopt a more popular framework to implement the deep learning module in this pape - Tensorflow 1.2.0.

C. Evaluation Measure

We adopt the Mean Absolute Error (MAE) method and the Root Mean Squared Error (RMSE) method, which are widely used in many fields, including recommendation systems, to evaluate the performance of our model. The expressions are as follows:

$$MAE = \frac{1}{N}\sum_{i,j}|R_{ij} - \hat{R}_{ij}|,$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i,j}\left(R_{ij} - \hat{R}_{ij}\right)^2},$$

where N is the number of testing data samples and $\hat{R}_{ij}$ represents the prediction score according to Eq. (17). Obviously, the lower values of MAE and $RMSE$ indicate better performance of the model.

D. Compared Methods

In this paper, our experiment is divided into two parts. The first part compares the performance of different feature representation methods introduced in Section 3.1, with the goal of proving the effectiveness of the proposed feature representation method. In the second part of the experiment, we verify the accuracy of the proposed model by comparing with other recommendation algorithms. We chose the following famous and state-of-the-art recommendation algorithms to compare with our method:

- Item-based (IB)[10]: The basic idea of the Item-based method is to calculate the similarities between the items; then, the item that is similar to the items that are preferred by the active user is recommended.
- SVD [17]: This algorithm is based on the Singular Value Decomposition method, where the rating matrix is decomposed into three matrices. These matrices will be used to make the prediction.
- PMF [18]: A widely used matrix factorization model. In our experiments, we set the regularization parameters with the grid {0.01, 0.1, 0.2, 0.5, 1, 2, 5}.
- MCoC [27]: This method first clusters the users and items into subgroups, and then, it uses the basis CF method to make recommendations within each subgroup.
- DsRec [14]: This is a hybrid model that combines the matrix factorization model and the basis clustering model to improve the prediction accuracy.
- PMMMF [28]: This method uses Proximal Support Vector Machine (PSVM) to improve upon the traditional MMMF method.
- Hern[29]: This method decomposes the rating matrix into two non-negative matrices using the matrix factorization method based on a Bayesian probability model.
- SCC [30]: This method is also a recommendation algorithm based on clustering; the difference is that it only clusters the items using a self-constructing clustering method.
- TyCo [31]: The main idea of this method is from cognitive psychology for finding "neighbors" of users based on user typicality degrees in user groups.

In addition to feature reduction and feature extraction, the SVD and PMF algorithms are also commonly used

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2789866, IEEE Access

recommendation algorithms; therefore, in the second part of the experiment, we use those methods again.

E. Setting Parameters

Before the experiments, we need to set the parameters for our model. In the feature representation model, the dimensions of the user features and item features are related to a specific dataset. For the MovieLens-100K dataset, we set $a = 16$ and $b = 18$; for the MovieLens-1M dataset, we set $a = 20$ and $b = 20$; and for the Epinions dataset, we set $a = 24$ and $b = 22$. For all the datasets, we set the learning rate $\eta = 0.01$.

In the deep neural network model, the number of hidden layers is set to 2. In the input layer, the number of neurons is the sum of the parameters a and b, and the number of neurons in the first hidden layer is 27; there are 12 neurons in the second hidden layer and 5 in the output layer. The learning rate η for updating weights and biases is set to 0.001.

F. Experimental Results

In this paper, we have carried out the experiments about 20 times on all three datasets, and then removed one of the best and one of the worst result, taking the average value of the rest as the experimental result. In the first part of the experiment, we tested the effectiveness of the proposed feature representation model. First, we randomly selected 20% of the data of the dataset as the test set and the remaining 80% as the training set. We only used the MAE measure method in this experiment, and the experimental results are shown in Figure 4.

By observing the experimental results of Fig. 4, we can draw the following conclusions:

1) The experimental results of the RaF method on the three datasets are the worst. The reason for this result may be that the input data required by the neural network model need to be continuous and comparable; however, according to our assumptions, the missing value in the rating matrix will be replaced by 0. Applying mathematical operations to these values is meaningless during the training phase; meanwhile, those missing values will be considered as minimum values (0 less than 1), which means that the user does not like the item, which is clearly inaccurate.

2) The NCF method performs well on the MovieLens-100k dataset, but it performs poorly on the other two datasets. By analyzing the statistics of the three datasets in Table 1, we believe that the reason for this phenomenon is that the MovieLens-100k dataset is small. It is reasonable to use the One-Hot encoding technique; however, on larger datasets, the encoded vectors are very sparse, which makes it difficult for the neural network to learn effective features. Therefore, the performance will be worse.

3) The performance of the SVD algorithm on the three datasets is not ideal . The reason for this result may be that the method needs to preprocess the missing values, and the general approach to do this is to replace them with averages or modes, which makes the decomposed features not reliable enough.

4) Both the PMF method and the QPR model in this paper achieve better performances because the two methods do not need to focus on the impact of missing values. However, the experimental results show that the QPR model achieves better performance than the PMF method, which means that the features of the users and items are not completely independent; there may be some correlation between them. The QPR model proposed in this paper can better address these relationships by adding quadratic terms; therefore, it can obtain the best performance.

In the first part of the experiment, we verified the effectiveness of the proposed feature representation method. Next, we will verify that the combination of the feature representation model and the neural network model can improve the prediction accuracy. In this experiment, we use both the MAE and $RMSE$ metrics to evaluate the results. The experimental results are shown in Table 2.

By observing the experimental results in Table 2, we draw the following conclusions:

1) Our method achieves better performance under both the MAE and $RMSE$ metrics on all three datasets, which means that our model achieves higher prediction accuracy and proves that the combination of the QPR model and DNN model is effective.

2) It can be observed from the experimental results that the $RMSE$ value is larger than the MAE value under the same conditions, which means that the $RMSE$ metric is a better indicator of the performance of the algorithm. A greater penalty is applied to an element with a larger prediction error; therefore, to some extent, it reflects the prediction stability of the algorithm. The experimental results show that the proposed model achieves better performance than the other algorithms with respect to the $RMSE$ metric method, which means that the proposed model has high prediction stability.

3) The prediction performance of our model is very different on the three datasets. Given the statistical information in Table 1, we believe that this difference is caused by the number of ratings per user. The greater the number of user ratings, the more accurate are the features that can be learned by the model; therefore, the model achieves a higher prediction accuracy on that dataset.

G. Impact of Parameters

In this subsection, we will discuss the impact of the parameters on the performance of our model. We mainly discuss the influence of the feature dimensions a and b and the number of hidden layers h. In the following experiments, we used the MAE metric method.

To verify the effect of the feature dimensions on the experimental results, we selected various values of a and b in the experiment, and the results are shown in Figure 5. In Figure 5(a), we fixed $b = 20$; then, by adjusting the value of a to observe its effect on the performance, we

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2789866, IEEE Access

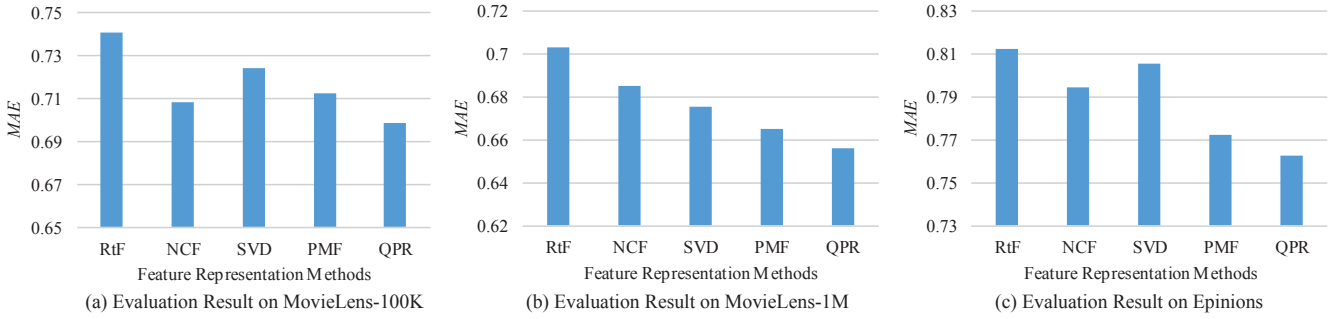

Fig. 4. The experimental results on the three datasets with different feature representation methods.

TABLE II
Evaluation Results with Different Recommendation Algorithms on the Three Datasets

		IB	SVD	PMF	MCoC	DsRec	PMMMF	Hern	SCC	TyCo	Our
MovieLens-100K	MAE	0.7324±0.0016	0.7392±0.0018	0.7421±0.0026	0.7215±0.0022	0.7118±0.0017	0.7226±0.0020	0.7061±0.0029	0.7334±0.0031	0.7313±0.0022	0.6962±0.0023
	RMSE	1.2311±0.0232	1.1032±0.0303	1.0683±0.0284	1.0943±0.0316	0.9923±0.0277	1.0291±0.0310	1.1022±0.0309	1.0036±0.0289	1.0316±0.0265	0.9874±0.0291
MovieLens-1M	MAE	0.7004±0.0014	0.6977±0.0020	0.6901±0.0013	0.6834±0.0011	0.6790±0.0013	0.6767±0.0015	0.6671±0.0015	0.6922±0.0023	0.6646±0.0017	0.6586±0.0012
	RMSE	0.9642±0.0133	0.9533±0.0203	0.9512±0.0185	0.9676±0.0152	0.9424±0.0176	0.9515±0.0184	0.9473±0.0142	0.9441±0.0200	0.9522±0.0154	0.9357±0.0151
Epinions	MAE	0.8421±0.0036	0.8577±0.0033	0.8544±0.0021	0.8482±0.0026	0.8127±0.0024	0.8233±0.0026	0.8081±0.0030	0.8342±0.0027	0.8136±0.0027	0.7747±0.0025
	RMSE	1.4322±0.0322	1.3467±0.0364	1.3022±0.0310	1.2785±0.0375	1.2534±0.0355	1.2573±0.0338	1.2467±0.0346	1.2485±0.0382	1.2502±0.0337	1.2405±0.0344

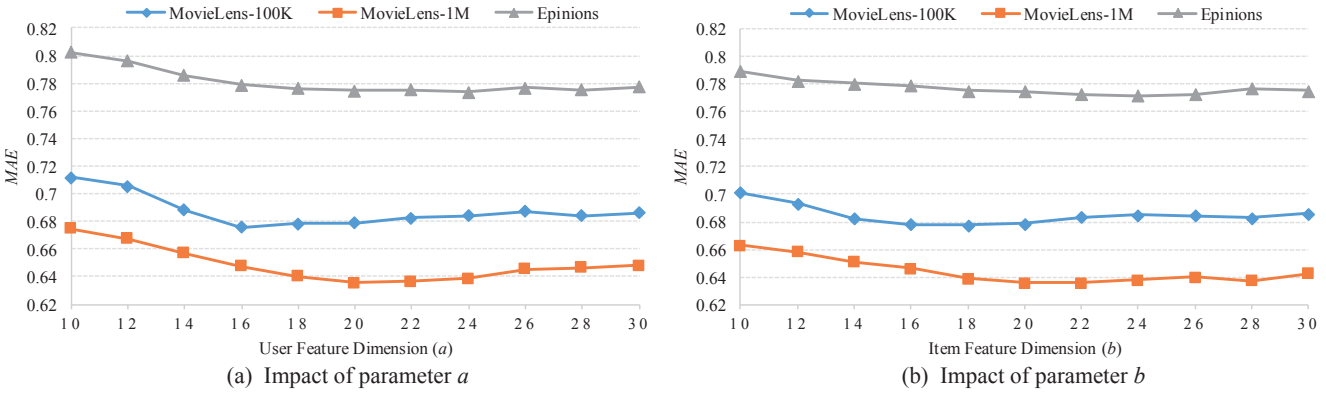

Fig. 5. The impact of the parameters a and b.

fixed $a = 20$ in Figure 5(b). The experimental results show that with increasing feature dimension, the performance of the model gradually improves and ultimately becomes stable. When the best performance is achieved, the values of a and b on the three datasets are consistent with the settings in Section 4.4.

Through further analysis of the experimental results, we observe that when the feature dimension is low, the features learned by our model are not sufficiently accurate, and there remains significant room for improvement. However, when the dimension reaches the best value but continues to increase, overfitting may occur. When the model achieves the best performance, larger datasets result in higher feature dimensions.

Next, we will discuss the influence of the number of hidden layers on the performance. The experimental results are shown in Figure 6, which shows that appropriately increasing the number of hidden layers can significantly improve the performance of the model; however, if more than 2 hidden layers are used, the performance of the model is almost no longer improved. Through analysis, we think that too many hidden layers may cause overfitting due to the low feature dimension of the input which comes from the feature representation model, therefore, we propose to set the number of hidden layers in the DNN model to 2.

V. Conclusion

In this paper, we discussed the effectiveness and implementation details of applying the DNN model to non-content-based recommendation systems. We first introduce a method of using a QPR model to obtain the latent features of users and items, then, we combine them with the DNN model. The experimental results show that the proposed model achieves good prediction performance, which proved that the application of deep learning model in recommender system is a successful attempt. Our framework is simple and generic, therefore, it is not limited to the method presented in this paper. One can regard the framework as a guideline for developing deep learning

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2789866, IEEE Access

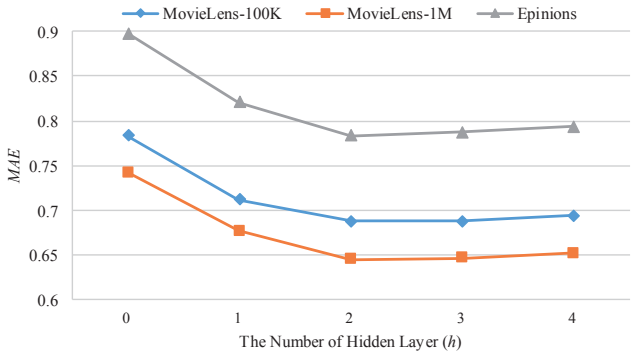

Fig. 6. The impact of the parameter h.

methods for recommendation systems. This paper is a preliminary attempt to apply deep learning methods to recommendation systems, so there are many possibilities for improvement, such as building more complex models, or using other deep learning methods.

In future work, we will study the application of other deep learning techniques, such as the convolutional neural network method in recommendation systems and attempt to further improve their performance. Specifically, we try to construct some user images by using the user's rating information, each element of the image corresponds to a certain feature of the user, and then use the convolutional neural network to mine the local features of the user, so as to cluster and recommend.

References

[1] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in The adaptive web. Springer, 2007, pp. 325–341.
[2] G. Adomavicius and A. Tuzhilin, "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, vol. 17, no. 6, pp. 734–749, 2005.
[3] Z. Huang, D. Zeng, and H. Chen, "A comparison of collaborative-filtering recommendation algorithms for e-commerce," IEEE Intelligent Systems, vol. 22, no. 5, 2007.
[4] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," Advances in artificial intelligence, vol. 2009, p. 4, 2009.
[5] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, 2012, pp. 3642–3649.
[6] F. Richardson, D. Reynolds, and N. Dehak, "Deep neural network approaches to speaker and language recognition," IEEE Signal Processing Letters, vol. 22, no. 10, pp. 1671–1675, 2015.
[7] E. Arisoy, T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Deep neural network language models," in Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT. Association for Computational Linguistics, 2012, pp. 20–28.
[8] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1998, pp. 43–52.
[9] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 1999, pp. 230–237.
[10] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in Proceedings of the 10th international conference on World Wide Web. ACM, 2001, pp. 285–295.
[11] T. Hofmann and J. Puzicha, "Latent class models for collaborative filtering," in IJCAI, vol. 99, no. 1999, 1999.
[12] K. Miyahara and M. Pazzani, "Collaborative filtering with the simple bayesian classifier," PRICAI 2000 Topics in Artificial Intelligence, pp. 679–689, 2000.
[13] S. Vucetic and Z. Obradovic, "Collaborative filtering using a regression-based approach," Knowledge and Information Systems, vol. 7, no. 1, pp. 1–22, 2005.
[14] J. Liu, Y. Jiang, Z. Li, X. Zhang, and H. Lu, "Domain-sensitive recommendation with user-item subgroup analysis," IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 4, pp. 939–950, 2016.
[15] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer, vol. 42, no. 8, 2009.
[16] 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. ACM, 2009, pp. 211–218.
[17] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender system-a case study," DTIC Document, Tech. Rep., 2000.
[18] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in Advances in neural information processing systems, 2008, pp. 1257–1264.
[19] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in Proceedings of the 24th international conference on Machine learning. ACM, 2007, pp. 791–798.
[20] K. Georgiev and P. Nakov, "A non-iid framework for collaborative filtering with restricted boltzmann machines," in International Conference on Machine Learning, 2013, pp. 1148–1156.
[21] A. Van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in Advances in neural information processing systems, 2013, pp. 2643–2651.
[22] X. Wang and Y. Wang, "Improving content-based and hybrid music recommendation using deep learning," in Proceedings of the 22nd ACM international conference on Multimedia. ACM, 2014, pp. 627–636.
[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. ACM, 2015, pp. 1235–1244.
[24] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in Proceedings of the 26th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.
[25] K. Jarrett, K. Kavukcuoglu, Y. LeCun et al., "What is the best multi-stage architecture for object recognition?" in Computer Vision, 2009 IEEE 12th International Conference on. IEEE, 2009, pp. 2146–2153.
[26] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323.
[27] B. Xu, J. Bu, C. Chen, and D. Cai, "An exploration of improving collaborative recommender systems via user-item subgroups," in Proceedings of the 21st international conference on World Wide Web. ACM, 2012, pp. 21–30.
[28] V. Kumar, A. K. Pujari, S. K. Sahu, V. R. Kagita, and V. Padmanabhan, "Proximal maximum margin matrix factorization for collaborative filtering," Pattern Recognition Letters, vol. 86, pp. 62–67, 2017.
[29] A. Hernando, J. Bobadilla, and F. Ortega, "A non negative matrix factorization for collaborative filtering recommender systems based on a bayesian probabilistic model," Knowledge-Based Systems, vol. 97, pp. 188–202, 2016.
[30] C.-L. Liao and S.-J. Lee, "A clustering based approach to improving the efficiency of collaborative filtering recommendation," Electronic Commerce Research and Applications, vol. 18, pp. 1–9, 2016.

[31] Y. Cai, H.-f. Leung, Q. Li, H. Min, J. Tang, and J. Li, "Typicality-based collaborative filtering recommendation," IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 3, pp. 766–779, 2014.