A Novel Deep Learning-Based Collaborative Filtering Model for Recommendation System

Mingsheng Fu, Hong Qu, *Member, IEEE*, Zhang Yi, *Fellow, IEEE*, Li Lu, *Member, IEEE*, and Yongsheng Liu

***Abstract*—The collaborative filtering (CF) based models are capable of grasping the interaction or correlation of users and items under consideration. However, existing CF-based methods can only grasp single type of relation, such as restricted Boltzmann machine which distinctly seize the correlation of user–user or item–item relation. On the other hand, matrix factorization explicitly captures the interaction between them. To overcome these setbacks in CF-based methods, we propose a novel deep learning method which imitates an effective intelligent recommendation by understanding the users and items beforehand. In the initial stage, corresponding low-dimensional vectors of users and items are learned separately, which embeds the semantic information reflecting the user–user and item–item correlation. During the prediction stage, a feed-forward neural networks is employed to simulate the interaction between user and item, where the corresponding pretrained representational vectors are taken as inputs of the neural networks. Several experiments based on two benchmark datasets (MovieLens 1M and MovieLens 10M) are carried out to verify the effectiveness of the proposed method, and the result shows that our model outperforms previous methods that used feed-forward neural networks by a significant margin and performs very comparably with state-of-the-art methods on both datasets.**

***Index Terms*—Collaborative filtering (CF), deep learning, feed-forward neural networks, recommender system.**

I. Introduction

Recommendation systems (RS) play an increasingly important role in modern online services. Existing methods for *RS* can be generally categorized into two classes: 1) content-based model and 2) collaborative filtering (CF) based model. *Content-based* methods [1], [2] use features that are extracted from the profile of the users or description of items to make new recommendations. CF-based methods [3]–[5] adopt users or items previous history of interactions, such as the rating given to the item by the user, which have attracted more attention due to their better prediction performance than the *content-based* method.

Manuscript received September 8, 2017; revised January 3, 2018; accepted January 11, 2018. This work was supported in part by the National Science Foundation of China under Grant 61573081 and Grant 61432012, and in part by the Foundation for Youth Science and Technology Innovation Research Team of Sichuan Province under Grant 2016TD0018. This paper was recommended by Associate Editor J. Liu. *(Corresponding author: Hong Qu.)*

M. Fu, H. Qu, and Y. Liu are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China, and also with the Institute for the Science and Application of Intelligent Learning, University of Electronic Science and Technology of China, Chengdu 610054, China (e-mail: uestc_fumingsheng@126.com; hongqu@uestc.edu.cn; rosstntt@gmail.com).

Z. Yi is with the College of Computer Science, Sichuan University, Chengdu 610065, China (e-mail: zhangyi@scu.edu.cn).

L. Lu is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China (e-mail: luli2009@uestc.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2018.2795041

There are two effective CF-based methods with the impressive performance: 1) matrix factorization (MF) [6]–[8] and 2) restricted Boltzmann machine (RBM) like methods [9]–[11]. MF directly learns the latent vectors of the users and the items from the user-item ratings matrix and captures the interaction between the user and the item. However, complex interactions cannot be captured by MF since its estimated rating is produced by the simple inner product between corresponding latent vectors of user and item. The RBM-like methods explicitly make recommendation from either user or item side via constructing independent models for users or items, respectively. However, in this model the correlation can only be considered from a single side, that is item–item or user–user, thus ignoring the other completely. Besides, the RBM-like method cannot retain the item-user interaction and they are not deep enough to capture complex features.

The most powerful approach to capture complex relations is to use deep learning techniques, which have made tremendous successes on many other pattern recognition tasks, such as computer vision [12]–[14], natural language processing [15], and speech recognition [16], [17]. Recently, deep neural networks for RS are also beginning to gain enormous attention [18]–[20]. Specially, for *content-based* model, numerous methods applying deep learning to recommendation had been proposed [21]–[24].

However, there are very limited researches on employing deep neural networks to CF-based model. Dziugaite and Roy [25] proposed a neural networks MF (NNMF) model to estimate rating by replacing the inner product in traditional MF to a feed-forward neural networks. Nonetheless, its performance is not competitive with the state-of-the-art methods. Wang *et al.* [26] introduced CDL, which constructed the latent vector of item from item's content text by *autoencoder* and then the ratings are estimated via MF using this latent vector. Deng *et al.* [27] proposed a deep learning-based MF, which employed deep *autoencoder* to generate initial vectors of users and items and adopted MF with the pretrained vectors to prediction for recommendation in social rating networks. However, it also requires additional information of user relationship and interest to produce estimation. Clearly, these methods still are the extensions of the traditional MF.

To overcome the setbacks of the CF-based methods above and develop a deep neural networks-based model beyond the simple extensions of MF, there are two crucial issues that are needed to be addressed: 1) the representations of users and items and 2) the architecture of the prediction neural networks. For the first issue, an effective features representation can significantly improve the performance on the relative tasks, such as the low-dimensional dense embedding for word and sentences [28], [29], and the image representation extracted from fixed pretraining neural networks [30]–[32]. However, unlike the form of representation of content which has been intensively exploited, generating the features of the user and the item from the ratings matrix (the usual data sources of CF-based method) for deep neural networks is still an open problem. For the second issue, the neural networks with special architecture adapted to a certain task have better performance than the normal neural networks, such as convolutional neural network (CNN) for computer vision [33], [34] and the recurrent neural network (RNN) for NLP [35], [36]. Hence, a well-designed architecture of the neural networks for CF-based RS is a pressing problem.

To effectively address the issues above, we propose to build prior-knowledge on users and items, then make prediction leveraging these obtained knowledge. Intuitively, the prior-knowledge can facilitate and benefit the prediction of user behavior [37]. This prior knowledge may originate from the past experience of the user [38]. To build the prior-knowledge of users from their past experience, inspired by the word embedding in NLP [28], [39], which can encode syntactic and semantic information of words into low-dimensional vector based on the context. We believe the semantic information of user can also be captured by learning the corresponding embedding from the "context" of the user, where the user–user co-occurrence in the user's past experience can be considered as the context of the user. Likewise, the knowledge of items can also be learned via item–item co-occurrence. Afterward, we propose using neural network to generate prediction from the prelearned embeddings of user and item. Consequently, This framework can be generally divided into two major stages: 1) understanding and 2) prediction. In the first stage, the embeddings of user and item can capture the user–user and item–item co-occurrence, respectively. In the second stage, the interaction between item and user can be simulated by the predictive neural network.

To construct the corresponding embeddings from the ratings matrix, we propose two different but complemental representational learning models. These include the constraint model (CM) and the rating independent model (RIM). To effectively extract the high-level features from the pretrained representations and estimate the rating accurately, we introduce several neural network methods that capture the interactions between users and items from two different directions: 1) current user and item and 2) the historical records of the items and users. For each direction, we implement two types of feed-forward neural networks that address different types of representations.

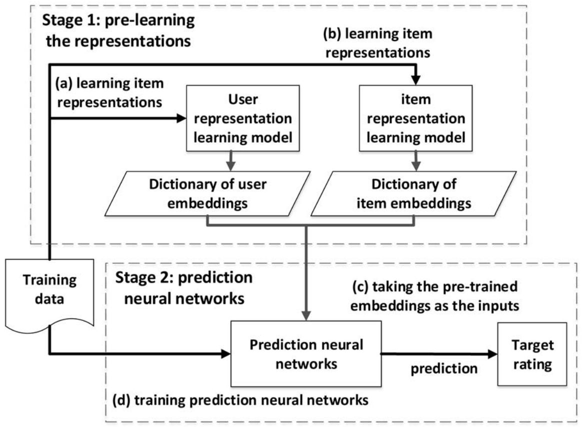

Stage 1: pre-learning the representations
(b) learning item representations
(a) learning item representations
User representation learning model
item representation learning model
Dictionary of user embeddings
Dictionary of item embeddings
Training data
Stage 2: prediction neural networks
(c) taking the pre-trained embeddings as the inputs
Prediction neural networks
prediction
Target rating
(d) training prediction neural networks

Fig. 1. Framework of our proposed model.

Inspired by the multiview neural networks for *content-base model* [23], [40], [41], we developed a multiview feed-forward neural networks that will take the information of both given user and item with their corresponding historical information into consideration. Several experiments based on two benchmark datasets (*MovieLens 1M* and *MovieLens 10M*) show results close to the state of the art feed-forward neural network and shear outperforming results when compared to previous CF-based methods.

The main contributions of this paper are as follows.

1) We introduce an effective pipeline to address implementation of deep feed-forward neural networks to CF, which first learns the embeddings of users and items, and then builds neural networks on them.
2) Two different but complemental representational learning models are proposed to generate the low-dimensional local and global representations for both users and items. These representations grasp the co-occurrences of users and items.
3) A multiview feed-forward neural networks are proposed to take all factors into consideration to capture the interaction between user and item, which includes the view of current user and item as well as their historical background.

II. Overview of Our Model

Generally, in our overall model, the target rating is estimated by a neural network which takes corresponding low-dimensional dense embeddings with respect to the given user and item, where the user and item embeddings are pretrained via special representation learning models before training the neural network. As can be seen in Fig. 1, the operations in our model can be subdivided into two major stages.

1) *Learning the Representations:* The representations learning model generates the corresponding low-dimensional dense embeddings of users and items from the ratings matrix according to co-occurrence of user–user and item–item relation as shown in steps (a) and (b), respectively, where the processes of learning the representations of users and items are independent of each other. In this paper, there are two models CM and RIM

which learn both user and item representations. These models are presented in Section III.

2) *Training the Prediction Neural Networks:* The target rating are estimated form a multiview neural network proposed in Section IV, whose inputs are the corresponding prelearned representations of the user and item these are CM and RIM learned representation as shown in step (c). The neural networks are trained from the rating-matrix according to the difference between the estimated rating and the real one as shown step (d). Note that the same rating-matrix are used to train the embeddings and neural network. However, the perspectives to the rating-matrix are different, the embeddings are learned from the perspective of user–user and item–item co-occurrence, but the network is learned from the perspective of user-item interaction.

Our overall model has the following advantages.

1) The pretrained representations contain the co-occurrence information with respect to the user and item. In addition, the pretrained representations can be flexible and thus freely be used in the multiviews neural networks since they are weakly coupled.
2) Training the embedding and neural networks separately is more effective and it yields a better performance in our overall model than jointly training them. In addition, the separated learning processes can capture both user-user, item–item co-occurrence, and the user-item interaction.

III. Representation Learning Models

Representational learning model aims to retain the semantic information by using an embedding of item and user denotation. In this section, we propose an effective method of learning the corresponding dense vector denotation of users and item, respectively, that captures the inter-relational information. First the co-occurrence of users and items of our model is introduced in Section III-A. Then, the model is extended to take account the rating information in Section III-B, in which we introduce two different yet complemental strategies. Finally, the approaches used for the model training are portrayed in Section III-C.

A. Learning Representations via Co-Occurrence

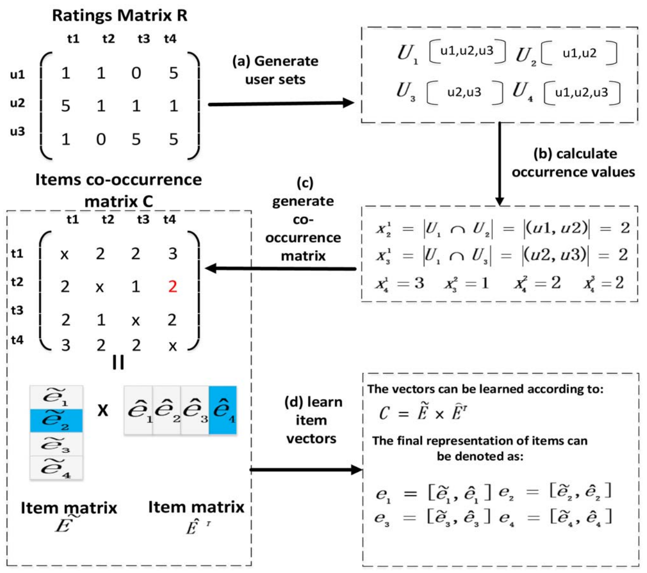
Inspired by the idea of Pennington's method [39] which can learn valid word vectors with semantic information by using the word-word co-occurrence in contexts, we believe that the similarity between items or users in RS can also be indicated by their co-occurrence.

Suppose that there are M items, N users for a given RS S and R_{N*M} is the ratings matrix in S, we describe notations separately in Table I.

Especially, $x_i^j = |U_i \cap U_j|$ and $\hat{x}_i^j = |T_i \cap T_j|$. e_i and r_i should be learned in our method.

To learn item embeddings from C, similar to MF, we represent C as

$$C = \tilde{E}\hat{E}^T = \left[\tilde{e}_1, \tilde{e}_2 \cdots \tilde{e}_M\right] * \left[\hat{e}_1, \hat{e}_2 \cdots \hat{e}_M\right]^T$$



Ratings Matrix R
t1 t2 t3 t4
u1 1 1 0 5
u2 5 1 1 1
u3 1 0 5 5
(a) Generate user sets
U_1 [u1,u2,u3] U_2 [u1,u2]
U_3 [u2,u3] U_4 [u1,u2,u3]
(b) calculate occurrence values
$x_2^1 = |U_1 \cap U_2| = |(u1, u2)| = 2$
$x_3^1 = |U_1 \cap U_3| = |(u2, u3)| = 2$
$x_4^1 = 3$ $x_3^2 = 1$ $x_4^2 = 2$ $x_4^3 = 2$
(c) generate co-occurrence matrix
Items co-occurrence matrix C
t1 t2 t3 t4
t1 x 2 2 3
t2 2 x 1 2
t3 2 1 x 2
t4 3 2 2 x
$\tilde{e}_1$ $\tilde{e}_2$ $\tilde{e}_3$ $\tilde{e}_4$
x
$\hat{e}_1$ $\hat{e}_2$ $\hat{e}_3$ $\hat{e}_4$
Item matrix $\tilde{E}$
Item matrix $\hat{E}^T$
(d) learn item vectors
The vectors can be learned according to:
$C = \tilde{E} \times \hat{E}^T$
The final representation of items can be denoted as:
$e_1 = [\tilde{e}_1, \hat{e}_1]$ $e_2 = [\tilde{e}_2, \hat{e}_2]$
$e_3 = [\tilde{e}_3, \hat{e}_3]$ $e_4 = [\tilde{e}_4, \hat{e}_4]$

Fig. 2. General flow of learning the embeddings of items includes four steps. In the first step (a), a set of users who marked a given item are extracted from the rating matrix in which each row and column represent a user and an item, respectively. Here, the value of an element is mapped in accordance with related rating [0 (no rating), 1 (negative rating), and 5 (positive rating)]. Afterward, the co-occurrence values of items are calculated according to the corresponding user sets in step (b), then in the following step (c) the item–item co-occurrence matrix is constructed from the original rating matrix and the precomputed values of co-occurrence. Finally, in step (d), the item vectors can be learned from the item–item co-occurrence matrix via minimizing (4).

TABLE I
Notation Descriptions

Notation	Description
t_i	the i-th item in S, where $i = 1, 2, \ldots .M$
u_i	the i-th user in S, where $i = 1, 2, \ldots .N$
U_i	set of users who had marked to a given item t_i
T_i	set of items which had been given rating by user u_i
x_i^j	the count of co-occurrence of items t_i and t_j
$\hat{x}_i^j$	the counts of co-occurrence users u_i and u_j
C	the co-occurrence matrix of items, and $C_{ij} = x_i^j$
$\hat{C}$	the co-occurrence matrix of users, and $\hat{C}_{ij} = \hat{x}_i^j$
e_i	embedding of the item t_i
r_i	embedding of the user u_i

where $\tilde{E}$ and $\hat{E}$ are two different matrices of embeddings of items which contain different semantic information of items, $\tilde{e}_i$ and $\hat{e}_i$ are the column vectors denoted item i in $\tilde{E}$ and $\hat{E}$, respectively.

Since $x_i^j = \tilde{e}_i\hat{e}_j^T$, and the real value of x_i^j can be obtained by statistics on R. Thus, $\tilde{e}_i$ and $\hat{e}_j$ can be learned by minimizing the difference between x_i^j and $\tilde{e}_i\hat{e}_j^T$. To smooth the co-occurrence value, $\log x_i^j$ is employed to replace the original x_i^j in our method. Consequently, the expression to capture the co-occurrence between t_i and t_j by the corresponding latent vectors can be denoted as

$$\log x_i^j = \tilde{e}_i^T \hat{e}_j \tag{1}$$

where $\tilde{e}_i$ and $\hat{e}_i$ are two different vectors which should be learned by x_i^j, and each of them contains different semantic information, respectively.

Then, the overall semantic information of item t_i can be denoted by

$$e_i = \left[\tilde{e}_i, \hat{e}_i\right] \tag{2}$$

where e_i is the concatenation of the vectors $\tilde{e}_i$ and $\hat{e}_i$ includes the semantic information of both $\tilde{e}_i$ and $\hat{e}_i$. The concatenated vector can also reduce over-fitting and noise [39].

In addition, involving additional biases b_i and b_j with respect to items t_i and t_j into (1) can further enhance the effectiveness of embeddings, and the final expression becomes

$$\log x_i^j = \tilde{e}_i{}^T \hat{e}_j + b_i + b_j. \tag{3}$$

Therefore, items' corresponding embeddings can be learned via minimizing the squared error between the predicting co-occurrence count of items with the real one

$$\min_{\tilde{e}_*, \hat{e}_*, b_*} \sum_{t_i, t_j \in T, i \neq j, x_i^j > 0}^{T} (\tilde{e}_i^T \hat{e}_j + b_i + b_j - \log x_i^j)^2 \tag{4}$$

where T is the set of items. The main flow of learning embeddings of items is illustrated in Fig. 2.

Similarly, for the user

$$\hat{C} = \tilde{U}\hat{U}^T = \left[\tilde{r}_1, \tilde{r}_2 \cdots \tilde{r}_N\right] * \left[\hat{r}_1, \hat{r}_2 \cdots \hat{r}_N\right]^T.$$

Then, the cost function to learning $\tilde{r}$ and $\hat{r}$ is denoted to

$$\min_{\tilde{r}_*, \hat{r}_*, \hat{b}_*} \sum_{u_i, u_j \in U, i \neq j, \hat{x}_i^j > 0}^{U} \left(\tilde{r}_i^T \hat{r}_j + \hat{b}_i + \hat{b}_j - \log \hat{x}_i^j\right)^2 \tag{5}$$

where $\hat{b}_i$ is the bias of user i.

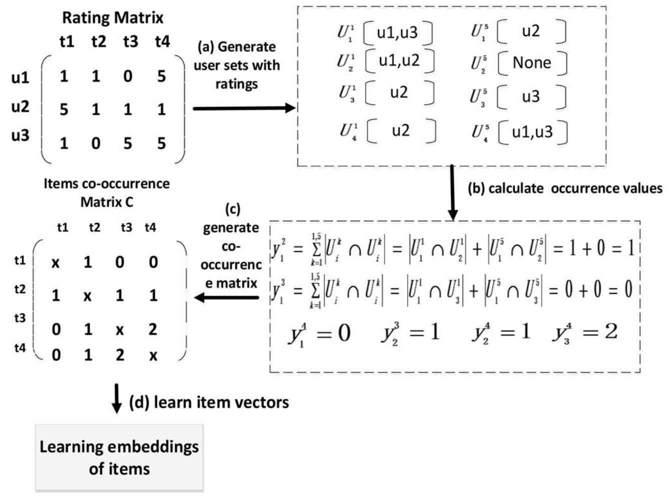
Similar to (2), the embedding r_i of u_i can be denoted to $r_i = [\tilde{r}_i, \hat{r}_i]$.

B. Learning Representations Using Ratings Information

So far our representation model without explicit consideration of rating can lead to the imprecise rating prediction in the later neural network. To address the rating information effectually, we propose two different yet complemental methods derived from the method in Section III-A: 1) CM and 2) RIM, which capture the overall co-occurrence and the local co-occurrence given by certain rating, respectively.

1) Constraint Model: Intuitively, the items are more similar if they are acquired more same rating from users. Otherwise, the items are irrelevant whose rating pairs are different usually. However, the basic model mentioned before cannot distinguish the pair of ratings. To address this problem of the basic model, a effective way is to count only the items giving the same ratings and drop the ones giving different ratings, this is named as CM.

Suppose that there are K kinds of ratings in S, let U_i^k be the set of users who give the item t_i to k rating and $z_{i,k}^{j,l} = |U_i^k \cap U_j^l|$ be the count of co-occurrence to the item t_i with k rating and the item t_j with l rating. Then, $y_i^j = \sum_{k=1}^{K} z_{i,k}^{j,k}$ is the total count of co-occurrence between t_i and t_j with the constraint, which takes account the one with same rating but drops the ones in different.



Rating Matrix
$$\begin{array}{c|cccc} & t1 & t2 & t3 & t4 \\ \hline u1 & 1 & 1 & 0 & 5 \\ u2 & 5 & 1 & 1 & 1 \\ u3 & 1 & 0 & 5 & 5 \end{array}$$
(a) Generate user sets with ratings
U_1^1 [u1,u3] U_1^5 [u2]
U_2^1 [u1,u2] U_2^5 [None]
U_3^1 [u2] U_3^5 [u3]
U_4^1 [u2] U_4^5 [u1,u3]
(b) calculate occurrence values
$y_1^2 = \sum_{k=1}^{1,5} |U_i^k \cap U_i^k| = |U_1^1 \cap U_2^1| + |U_1^5 \cap U_2^5| = 1 + 0 = 1$
$y_1^3 = \sum_{k=1}^{1,5} |U_i^k \cap U_i^k| = |U_1^1 \cap U_3^1| + |U_1^5 \cap U_3^5| = 0 + 0 = 0$
$y_1^4 = 0$ $y_2^3 = 1$ $y_2^4 = 1$ $y_3^4 = 2$
(c) generate co-occurrence matrix
Items co-occurrence Matrix C
$$\begin{array}{c|cccc} & t1 & t2 & t3 & t4 \\ \hline t1 & x & 1 & 0 & 0 \\ t2 & 1 & x & 1 & 1 \\ t3 & 0 & 1 & x & 2 \\ t4 & 0 & 1 & 2 & x \end{array}$$
(d) learn item vectors
Learning embeddings of items

Fig. 3. Main flow of learning CM item embeddings.

Consequently, the cost function of CM can be denoted to

$$\min_{\tilde{e}_*, \hat{e}_*, b_*} J_t = \sum_{t_i, t_j \in T, i \neq j, y_i^j > 0}^{T} \left(\tilde{e}_i^T \hat{e}_j + b_i + b_j - \log y_i^j\right)^2. \tag{6}$$

Similarly, the user representations can be learned by the cost function as follow:

$$\min_{\tilde{r}_*, \hat{r}_*, \hat{b}_*} J_u = \sum_{u_i, u_j \in U, i \neq j, \hat{y}_i^j > 0}^{U} \left(\tilde{r}_i^T \hat{r}_j + \hat{b}_i + \hat{b}_j - \log \hat{y}_i^j\right)^2 \tag{7}$$

where $\hat{y}_i^j = \sum_{k=1}^{K} \hat{z}_{i,k}^{j,k}$ and $\hat{z}_{i,k}^{j,k} = |\, T_i^k \cap T_j^l \,|$ and T_i^k is the set of items given rating k by u_i.

The main flow of CM (illustrated in Fig. 3) is similar to the one in Fig. 2, but the way of counting the co-occurrence are replaced to the restrictive way in step (b). Thus, the generated co-occurrence matrix in Fig. 3 is different from the one in Fig. 2, which conduces to produces more effective representation.

2) Rating Independent Model: CM forces on capturing the global relationship between items. However, the item with different ratings also implies different local information of item, and which cannot be revealed by CM. To address this problem, RIM is proposed, in which the item giving different ratings are denoted by different and independent embeddings.

Given different ratings, suppose that each item has K kinds of embeddings to represent an item, and e_i^k is the embedding of item t_i with the kth rating. The value of co-occurrence between e_i^k and e_j^l is equal to $z_{i,k}^{j,l}$. Consequently, the cost function of this model is

$$\min_{\tilde{e}_*, \hat{e}_*, b_*} \hat{J}_t = \sum_{t_i^k, t_j^l \in T, k \neq l, z_{ti,k}^{tj,l} > 0}^{T} \left(\left(\tilde{e}_i^k\right)^T \hat{e}_j^l + b_i + b_j - \log z_{ti,k}^{tj,l}\right)^2. \tag{8}$$

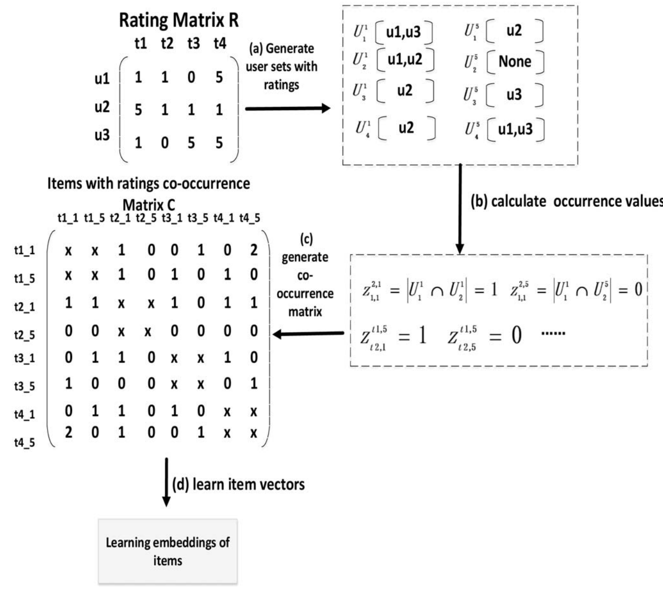

Rating Matrix R
t1 t2 t3 t4
u1 1 1 0 5
u2 5 1 1 1
u3 1 0 5 5
(a) Generate user sets with ratings
U_1^1 [u1,u3] U_1^5 [u2]
U_2^1 [u1,u2] U_2^5 [None]
U_3^1 [u2] U_3^5 [u3]
U_4^1 [u2] U_4^5 [u1,u3]
(b) calculate occurrence values
$z_{1,1}^{2,1} = |U_1^1 \cap U_2^1| = 1$ $z_{1,1}^{2,5} = |U_1^1 \cap U_2^5| = 0$
$z_{t2,1}^{t1,5} = 1$ $z_{t2,5}^{t1,5} = 0$ ......
(c) generate co-occurrence matrix
Items with ratings co-occurrence Matrix C
t1_1 t1_5 t2_1 t2_5 t3_1 t3_5 t4_1 t4_5
t1_1 x x 1 0 0 1 0 2
t1_5 x x 1 0 1 0 1 0
t2_1 1 1 x x 1 0 1 1
t2_5 0 0 x x 0 0 0 0
t3_1 0 1 1 0 x x 1 0
t3_5 1 0 0 0 x x 0 1
t4_1 0 1 1 0 1 0 x x
t4_5 2 0 1 0 0 1 x x
(d) learn item vectors
Learning embeddings of items

Fig. 4. Main flow of learning RIM item embeddings.

Similarly, RIM also can be adopted to learn the representations of users as follow:

$$\min_{\tilde{r}_*,\hat{r}_*,\hat{b}_*} \hat{J}_u = \sum_{u_i^k,u_j^l\in U,k\neq l,\hat{z}_{i,k}^{j,l}>0}^{U} \left(\left(\tilde{r}_i^k\right)^T \hat{r}_j^l + \hat{b}_i + \hat{b}_j - \log \hat{z}_{ti,k}^{tj,l}\right)^2 \tag{9}$$

where $\tilde{r}_i^k$ and $\hat{r}_i^k$ are the user vectors for u_i with rating k.

The main flow of RIM to learning representations is illustrated in Fig. 4. The converted co-occurrence matrix is different from the ones in Figs. 2 and 3, where each row and each column denote an item giving a certain rating. Thus, the calculation of a particular co-occurrence value related to certain row and column does not take the full range of ratings into account. Besides, the distinction between an item given different ratings can be exhibited by RIM.

C. Optimization

To learning the item and user embeddings of CM, AdaGrad [42] is employed to minimize the cost functions (6)–(9). The derivations of the cost functions considering all parameters can be calculated as

$$\begin{aligned}\frac{\partial J_t}{\partial \tilde{e}_i} &= 2\psi\tilde{e}_i,\ \frac{\partial J_t}{\partial \hat{e}_j} = 2\psi\hat{e}_j,\ \frac{\partial J_t}{\partial b_i} = 2\psi,\ \frac{\partial J_t}{\partial b_j} = 2\psi\\ \frac{\partial J_u}{\partial \tilde{r}_i} &= 2\varphi\tilde{r}_i,\ \frac{\partial J_u}{\partial \hat{r}_j} = 2\varphi\hat{r}_j,\ \frac{\partial J_u}{\partial \hat{b}_i} = 2\varphi,\ \frac{\partial J_u}{\partial \hat{b}_j} = 2\varphi\end{aligned} \tag{10}$$

where

$$\begin{aligned}\psi &= \tilde{e}_i^T\hat{e}_j + b_i + b_j - \log y_i^j\\ \varphi &= \tilde{r}_i^T\hat{r}_j + \hat{b}_i + \hat{b}_j - \log \hat{y}_i^j.\end{aligned}$$

Similarly, to learning the representations of RIM, the derivations of (8) and (9) can be stated as

$$\begin{aligned}\frac{\partial \hat{J}_t}{\partial \tilde{e}_i^k} &= 2\hat{\psi}\tilde{e}_i^k,\ \frac{\partial \hat{J}_t}{\partial \hat{e}_j^l} = 2\hat{\psi}\hat{e}_j^l,\ \frac{\partial \hat{J}_t}{\partial b_i} = 2\hat{\psi},\ \frac{\partial \hat{J}_t}{\partial b_j} = 2\hat{\psi}\\ \frac{\partial \hat{J}_u}{\partial \tilde{r}_i^k} &= 2\hat{\varphi}\tilde{r}_i^k,\ \frac{\partial \hat{J}_u}{\partial \hat{r}_j^l} = 2\hat{\varphi}\hat{r}_j^l,\ \frac{\partial \hat{J}_u}{\partial \hat{b}_i} = 2\hat{\varphi},\ \frac{\partial \hat{J}_u}{\partial \hat{b}_j} = 2\hat{\varphi}\end{aligned} \tag{11}$$

where

$$\begin{aligned}\hat{\psi} &= (\tilde{e}_i^k)^T\hat{e}_j^l + b_i + b_j - \log z_{i,k}^{j,l}\\ \hat{\varphi} &= (\tilde{r}_i^k)^T\hat{r}_j^l + \hat{b}_i + \hat{b}_j - \log \hat{z}_{i,k}^{j,l}.\end{aligned}$$

Then, the parameters can be updated by

$$\begin{aligned}&\tilde{e}_i \leftarrow \tilde{e}_i - \eta\frac{\partial J_t}{\partial \tilde{e}_i},\ \hat{e}_j \leftarrow \hat{e}_j - \eta\frac{\partial J_t}{\partial \hat{e}_j},\ \tilde{e}_i^k \leftarrow \tilde{e}_i^k - \eta\frac{\partial \hat{J}_t}{\partial \tilde{e}_i^k}\\ &\hat{e}_j^l \leftarrow \hat{e}_j^l - \eta\frac{\partial \hat{J}_t}{\partial \hat{e}_j^l},\ \tilde{r}_i \leftarrow \tilde{r}_i - \eta\frac{\partial J_u}{\partial \tilde{r}_i},\ \hat{r}_j \leftarrow \hat{r}_j - \eta\frac{\partial J_u}{\partial \hat{r}_j}\\ &\tilde{r}_i^k \leftarrow \tilde{r}_i^k - \eta\frac{\partial \hat{J}_u}{\partial \tilde{r}_i^k},\ \hat{r}_j^l \leftarrow \hat{r}_j^l - \eta\frac{\partial \hat{J}_u}{\partial \hat{r}_j^l}\end{aligned} \tag{12}$$

where η is the learning rate.

D. Comparison

In contrast with RIM, CM captures the global co-occurrence, while RIM captures the local co-occurrence in certain ratings. We use a case to show the difference clear. Given items A and B, there are three kinds of different ratings combinations: 1) *A@4-B@5*; 2) *A@4-B@4*; and 3) *A@5-B@5*, then *A@4-B@5* indicates items A is given 4-star rating, and items B is given 5-star.

In this case, There are a total of five RIM embedding. Two local RIM embeddings of item A $\tilde{e}_A^4$ and $\tilde{e}_A^5$ are employed to capture the semantic information of item A from the combinations. Similarly, there are two RIM embeddings $\hat{e}_B^4$ and $\hat{e}_B^5$ for item B. In here, *A@4-B@5* can be captured by $\tilde{e}_A^4$ and $\hat{e}_B^5$. Likewise, *A@4-B@4* can be captured by $\tilde{e}_A^4$ and $\hat{e}_B^4$, as well $\tilde{e}_A^5$ and $\hat{e}_B^5$ for *A@5-B@5*.

On the other hand, there are only two global CM embeddings $\tilde{e}_A$ and $\hat{e}_B$ for items A and B to capture the overall co-occurrence of A and B. Note that *A@4-B@4* and *A@5-B@5* are considered in learning $\tilde{e}_A$ and $\hat{e}_B$, but *A@4-B@5* is ignored, since we want capture rating similarity in CM rather than the differences.

As we can see from above example, the local co-occurrences with respect to different ratings can be explicitly revealed by RIM embeddings. For example, $\tilde{e}_A^4$ indicates special information of item A on 4-star rating. But, the co-occurrence captured by CM is an overall representation. Generally, RIM can retain more general information, but it is slightly sensitive. On the other hand, CM is more robust, but it can only capture constrained information. In this, RIM and CM are of different perspective yet complemental. In addition, experiment results show the performance of using both of CM and RIM in the prediction neural networks is superior to the ones using only CM or RIM.

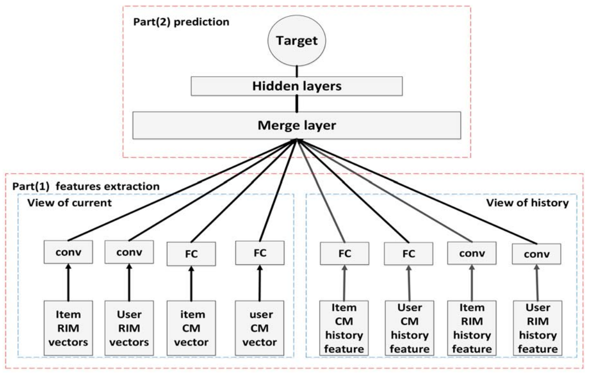


Fig. 5. Architecture of the multiviews neural networks includes two parts: multiview features extraction and prediction integration. In the multiview features extraction, the inputs are pretrained user and item vectors which are learned via CM and RIM methods and generated historical features. "FC" and "CONV" denoting fully connected transformation layer and convolutional transformation layer, respectively; FC and CONV are exhibited in Sections IV-B1 and IV-B2.

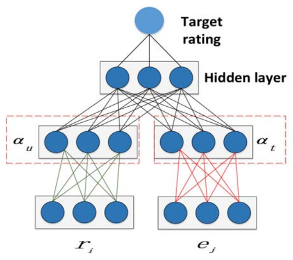


Fig. 6. Architecture of the neural networks using only CM embeddings on the view of current user and item. The inputs are CM embeddings r_i and e_j with respect to user u_i and item t_j, respectively. α_u and α_t are the output of transformation layers for user and item, respectively, (denoted by green and red connection lines, respectively).

CM and RIM are different from the traditional MF method. In MF, the learned user and item representations are directly related to rating prediction. On the other hand, our learned representations are not directly correlated to rating prediction, they focus on embedding the semantic information of user and item in to low-dimensional vectors. To generate the prediction, additional neural networks are employed, and the representations of CM or RIM are fed as inputs of the neural networks (the detail of prediction networks are exhibited in Section IV). Thus, the co-occurrence relationships of user–user and item–item can be grasped by our methods and the user-item interaction can be captured by the subsequent neural networks to predict the target rating. Capturing both relationships of co-occurrence and interaction is the advantage of our overall method with the traditional MF. Moreover, experiment results show it can outperform MF in terms of root mean square error (RMSE) and recall.

IV. Prediction Neural Networks

A. *Overall Architecture of Multiviews Neural Networks*

The learned CM and RIM embeddings include the semantic information with respect to the user and item. However, the embeddings cannot directly generate target rating. Thus, the additional component is required to make prediction from these embeddings. In this paper, neural networks are addressed to be the predictive component since neural networks can effectively extract features from prelearned embeddings and make accurate prediction. In addition, it can effectively merge the information from different sides.

The most direct path to follow in generating target rating of given user and item via neural network is to feed the corresponding pretrained CM or RIM embeddings of user and item into a feed-forward neural network, and the network will output the target rating. This way the prediction can be seen from the perspective of the user and item as it only depends on the two embedding vectors. However, its performance is not good enough. To improve the performance, we take additional historical-information into account, inspired by the SVD++ which considers additional item history information to make prediction. Thus, we propose a multiviews neural networks relying on both current and historical information. Moreover, both CM or RIM are fed into the multiviews neural networks to further enhance the performance since CM or RIM capture different kinds of information.

The overall architecture of the multiview neural networks is illustrated in Fig. 5. The networks take the pretrained CM and RIM embeddings with respect to user and item as input, and it outputs the target rating, and the overall network can be separated into two major parts: 1) multiview features extracting and 2) integrated prediction.

To predict the rating that a user u_i giving to t_j, in the part of extracting feature, there are two kinds of views to be taken account: 1) current user and item: u_i and t_j themselves and 2) historical records: history items of u_i and set of users who have rated t_j.

During extracting features, each kind of representations is taken into account independently. The fully connected transformation layer and the convolutional transformation layer are fed to CM and RIM, respectively, to extract effective features from the original representations.

During integration of predictions, a merged layer is adapted to concatenate the features produced by the previous layers. After merging these predictions, a traditional feed-forward multilayers neural networks are built on the concatenated features to produce the final rating estimation.

Suppose that for the user u_i and an item t_j, r_i is u_i's vector of CM and r_j^k is u_i' vector of RIM with rating k, where $k = 1, 2, \ldots .K$. e_j is t_j's vector of CM and e_j^k is a vector with rating k, where $k = 1, 2, \ldots .K$. Let $\tilde{e}_i$ and $\tilde{r}_j$ be the historical features of u_i and t_j, respectively. Then, let α_u and α_t be the CM extracted features from the perspective of current user and item, and β_u and β_t be the RIM extracted features on the perspective of current user and item. Similarly, γ_u and γ_t be the CM extracted features from the perspective of historical records, and δ_i^j be the RIM extracted features from the perspective of historical records.

Thus, the concatenating features in the merged layer can be denoted as

$$\mu_u^t = \big[\alpha_u(ui), \alpha_t(tj), \beta_u(ui), \beta_t(tj), \gamma_u(ui)$$
$$\gamma_t(tj), \delta_u(ui), \delta_t(tj)\big]$$

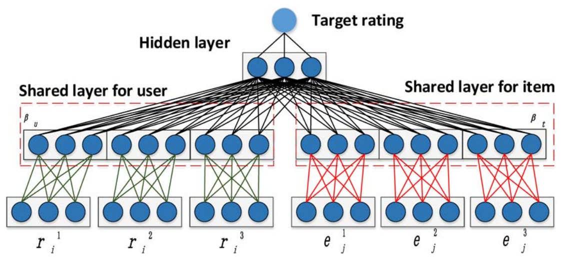


Fig. 7. Architecture of the neural networks using only the RIM embeddings on the view of current user and item. Given ratings 1, 2, and 3, there are three RIM embeddings r_i^1, r_i^2 and r_i^3 for user u_i. Likewise, there are three RIM embeddings $e_j^1 e_j^2$ and e_j^3 for item t_j. All of them are fed into the neural networks. β_u and β_t are the outputs of the convolutional transformation layers for user and item, respectively.

and the rest of the model which is the prediction neural networks is, $p_{i,j} = f_\Theta(\mu_u^t)$, where f_Θ is a feed-forward neural networks and $p_{i,j}$ is the predicted rating with respect to ui given to tj, and the activation function g in f_Θ is rectified linear unit which can be defined as $g(x) = \max(x, 0)$, however, the activation function does not apply to the last layer because $p_{i,j}$ is the real value.

The details of extracting features on the views of current and historical records are introduced in Sections IV-B and IV-C, respectively.

B. Neural Networks on the Current User and Item View

Essentially, the transformation layer converts the representations in the original space into a new space, where the transformed features are more suitable for the final task. To address this transformation task for vectors of CM and RIM, we propose two kinds of neural networks architectures as follow.

1) Fully Connected Transformation Layer for CM Embeddings: For the *CM* embeddings of users and items, there exist two different original embedding spaces, the user space and the item space. To transform the original CM embeddings r_i and e_j, a traditional fully connected layers are used since the embeddings is overall and identical to the item or user. Let W^e and W^r be the fully connected weight matrices for the item and the user, respectively, then the fully connected layers can be denoted as follows:

$$\alpha_t(tj) = g(W^e e_j), \quad \alpha_u(ui) = g(W^r r_i). \tag{13}$$

Note that by using only CM embeddings, we can construct a single view prediction neural network in the perspective of current user and item as the architecture illustrated in Fig. 6.

2) Convolutional Transformation Layer for RIM Embeddings: Given K kinds of rating scores, in RIM, user u_i and item t_j can be denoted as

$$\nu_i(ui) = \left[r_i^1, r_i^2, \ldots, r_i^k\right], \quad \mu_j(tj) = \left[e_j^1, e_j^2, \ldots, e_j^k\right] \tag{14}$$

where $\nu_i(ui)$ and $\mu_j(tj)$ are the unique overall embeddings of u_i and t_j, respectively. Note that zeros vectors are used to denote the r_i^k and e_j^k which do not appear in the training set.

In order to extract effective features from ν_i and μ_j, the most direct way is to subject each local user and item embeddings r_i^k or e_j^k to fully connected transformation matrices W_r^k and W_k^e, respectively, which are similar to the way single user and item in CM embeddings are processed. In addition, the multiple fully connected layers with respect to K rating scores can also be seen as two special locally connected layers handling ν_i and μ_j. Let β_u and β_t be the outputs of the locally connected layers handling ν_i and μ_j, respectively, the local connection layers can be denoted as follows:

$$\beta_u(ui) = \left[\nu_i^1, \nu_i^2, \ldots \nu_i^k\right], \quad \beta_t(tj) = \left[\mu_i^1, \mu_i^2, \ldots \mu_i^k\right] \tag{15}$$

where ν_i^k and μ_i^k can be yielded by

$$\begin{aligned} \nu_i^k &= g\left(W_k^r \nu_i(ui)[(k-1) \times l_r : k \times l_r]\right) \\ \mu_j^k &= g\left(W_k^e \mu_j(tj)[(k-1) \times l_e : k \times l_e]\right) \end{aligned} \tag{16}$$

where l_u and l_e are the dimensions of r_i^k and e_j^k, respectively, $\nu_i(ui)[(k-1) \times l_e : k \times l_e]$ is the subvector of $\nu_i(ui)$ from the $(k-1) \times l$th element to the $k \times l$th element.

Because the user RIM embeddings regarded as different rating scores that actually exist on the same user space, various W_r^k in (16) can be alternated by a shared transformation matrixes $\hat{W}^r$ between different ratings. $\hat{W}^r$ can be viewed as the weight to transform the original user space to the new features space. Similarly, different W_e^k in (16) can be alternated by a shared $\hat{W}^e$. Thus, (16) can be modified to

$$\begin{aligned} \nu_i^k(ui) &= g\left(\hat{W}^r \nu_i(ui)[(k-1) \times l_r : k \times l_r]\right) \\ \mu_j^k(tj) &= g\left(\hat{W}^e \mu_j(tj)[(k-1) \times l_e : k \times l_e]\right). \end{aligned} \tag{17}$$

Note that these shared transformation layers can be seen as special convolutional layers.

The architecture of the neural network using only RIM embeddings on the view of current user and item is illustrated in Fig. 7.

C. Neural Networks on the Historical View

To effectively take the historical records into account, we propose simple but effective methods to generate features of historical records from the original CM and RIM embeddings of users and items, and then fit them into corresponding subneural networks on the two different views.

1) Fully Connected Transformation Layer for CM Historical Features: Suppose that T_i is the set of historical items rated by user u_i, and U_j is the set of users marking item t_j. The most direct way to represent u_i's historical records $h(ui)$ is the averaged vectors of items in T_i. t_j's historical records $\hat{h}(tj)$ also can be obtained by the similar way. Formally, $h(ui)$ and $\hat{h}(tj)$ can be denoted as

$$h(ui) = \frac{\sum_{t_q \in T_i} e_q}{|T_i|}, \quad \hat{h}(tj) = \frac{\sum_{u_q \in U_j} r_q}{|U_j|}. \tag{18}$$

Similar to the neural network with the current user and item's view, addressed in CM embedding, fully connected transformation layers can be employed to handle the historical features which can be denoted as

$$\gamma_u(ui) = g\left(\tilde{W}^{hu} h(ui)\right), \quad \gamma_t(tj) = g\left(\tilde{W}^{ht} \hat{h}(tj)\right) \tag{19}$$

where $\tilde{W}^{hu}$ and $\tilde{W}^{ht}$ are the fully connected transformation weight matrices to h and $\hat{h}$, respectively.

The architecture of the neural network using CM historical features is similar to the one shown in Fig. 6.

2) Convolutional Transformation Layer for RIM Historical Features: Similar to CM, the historical representations $h^k(ui)$ and $\hat{h}^k(tj)$ with respect to u_i and t_j in the kth rating, respectively, can also be directly represented by the averaged embeddings. Let T_i^k be the set of items giving k rating by u_i, and U_j^k be the set of users marking t_j to k rating. Thus, $h^k(ui)$ and $\hat{h}^k(th)$ can be denoted by

$$h^k(ui) = \frac{\sum_{t_q \in T_i^k} e_q^k}{\left|T_i^k\right|}, \quad \hat{h}^k(tj) = \frac{\sum_{u_q \in U_j^k} r_q^k}{\left|U_j^k\right|}. \tag{20}$$

Then, the representations of user and item regarding K kinds of ratings $p(ui)$ and $\hat{p}(tj)$ can be denoted as

$$\begin{aligned} p(ui) &= \left[h^1(ui), h^2(ui), \dots h^K(ui)\right] \\ \hat{p}(tj) &= \left[\hat{h}^1(tj), \hat{h}^2(tj), \dots \hat{h}^K(tj)\right]. \end{aligned} \tag{21}$$

Similar to (15), δ_u and δ_t can be obtained by

$$\begin{aligned} \delta_u(ui) &= \left[p(ui)^1, p(ui)^2, \dots p(ui)^K\right] \\ \delta_t(tj) &= \left[\hat{p}(tj)^1, \hat{p}(tj)^2, \dots \hat{p}(tj)^K\right] \end{aligned} \tag{22}$$

where $p(ui)^k$ and $\hat{p}(tj)^k$ can be yielded by

$$\begin{aligned} p(ui)^k &= g\left(\hat{W}^r \hat{hu}_i[(k-1) \times l_r : k \times l_r]\right) \\ \hat{p}(tj)^k &= g\left(\hat{W}^e \hat{ht}_j[(k-1) \times l_e : k \times l_e]\right) \end{aligned} \tag{23}$$

where $\hat{W}^r$ and $\hat{W}^e$ are shared weight matrices with respect to all kinds of ratings for the user and the item, respectively.

Thus, this architecture is similar to the counterpart in RIM on the view of current as illustrated in Fig. 7.

D. Optimization

The cost function of the neural networks is *Mean Square Error*: $\min (1/|D|) \sum_{u_i, t_j \in D} (f_\vartheta(u_i, t_j) - r_i^j)^2$, where f_ϑ denotes the multiview neural networks and D is the training set and r_i^j is the actual rating that u_i gave to t_j.

The prediction neural networks are trained via stochastic gradient descent (SGD), the detailed processes of training is sketched in Fig. 8, where the early stop condition is the improvement of loss in the validation set. Especially, the learning rate η is adjusted to the half of current η while the improvement is lower than the tolerance value ε, which can further reduce the loss in practice.

Note that we do not update any user and item vectors in term of CM and RIM in training prediction neural networks because of additional learning dose damage to the original semantic meaning of the representations. In addition, a certain user or item representation could occur in different branches of the neural networks which results in extremely sophisticated representations for training.

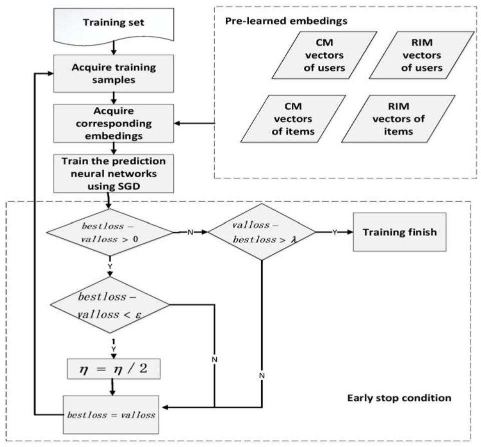


Fig. 8. Framework of training the prediction neural networks takes prelearned and pregenerated vectors as inputs. "Valloss" is the MSE loss of validation set V in current epoch, which can be obtained by $(1/|V|) \sum_{u_i, t_j} (f_\vartheta(u_i, t_j) - r_i^j)^2$. "bestloss" is the minimums *Valloss* obtained beforehand. η is the training rate and it will be adjusted in training. ε is the tolerance value determining the improvement in the learning rate and λ is the tolerance value determining the continuity of the training.

V. Experiments

A. Evaluation Datasets

We evaluated the performance of our proposed method on two real-world datasets MovieLens 1M and MovieLens 10M. The details of the datasets are shown as follow.

1) *MovieLens 1M* contains around 1 million ratings of approximately 3900 movies by 6040 users in which there are five grade ratings and each user rated at least 20 items.
2) *MovieLens 10M* contains about 10 million ratings of 10 681 movies by 71 567 users in which there are ten grade ratings and each user also rated at least 20 items.

Following LLORMA [43], 10% of the ratings in both datasets are randomly selected as the test set, leaving the remaining 90% of the samples as the training set. Among the samples in the training set, 1% are randomly chosen as the validation set. We use the averaged rating in training set to assign the items which do not contain training set.

B. Analysis of Learned Representations

1) Visualization of Learned Representations: To exhibit the learned representations clearly, we visualize the items' RIM vectors in two dimensions by t-SNE [44], where the vectors are learned from *MovieLen 1M*. There are two visualized results exhibited in Fig. 9(a) and (b).

In Fig. 9 (a), the items are classified into three classes including negative (rating 1 and 2), neutral (rating 3), and positive (rating 4 and 5). As can be seen in Fig. 9(a), items are clustered very well and the positions of clusters have a sentiment decreasing relations, where the clusters of negative, neutral, and positive lie on the bottom, center, and top, respectively.

In Fig. 9(b), the items are divided into five classes (the types of ratings). As can be seen in Fig. 9(b), the adjacent clusters

TABLE II
SIMILAR MOVIES ACQUIRED BY THE *RIM* EMBEDDING

cases	Star Trek: First Contact (5)	Toy Story (5)	Star Wars: Episode IV - A New Hope (5)
top 5 most similar movies	Star Trek: Generations (5)	Toy Story 2 (5)	Star Wars: Episode V - The Empire Strikes Back (5)
	Star Trek: Insurrection (5)	Tarzan (5)	Star Wars: Episode VI - Return of the Jedi (5)
	Star Trek IV: The Voyage Home (5)	A Bug's Life (5)	Raiders of the Lost Ark (5)
	Star Trek VI: The Undiscovered Country (5)	Aladdin (5)	The Matrix (5)
	Star Trek III: The Search for Spock (5)	The Lion King (5)	Indiana Jones and the Last Crusade (5)

TABLE III
SIMILAR MOVIES ACQUIRED BY THE *CM* EMBEDDING

cases	Star Trek: First Contact	Toy Story	Star Wars: Episode IV - A New Hope
top 5 most similar movie	Star Trek VI: The Undiscovered Country	Toy Story 2	Star Wars: Episode V - The Empire Strikes Back
	Star Trek IV: The Voyage Home	Groundhog Day	Raiders of the Lost Ark
	Star Trek: Generations	Raiders of the Lost Ark	The Matrix
	Star Trek: Insurrection	Aladdin	The Godfather
	Star Trek: The Wrath of Khan	The Princess Bride	Star Wars: Episode VI - Return of the Jedi

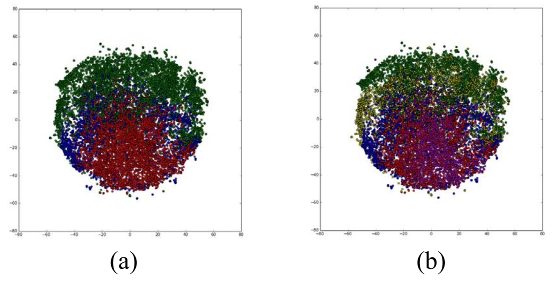


Fig. 9. t-SNE visualization of the RIM embedding of items. (a) Three classes, negative (red), neutral (blue), and positive (green). (b) Four classes, 1 rating (magenta), 2 rating (red), 3 rating (blue), 4 rating (yellow), and 5 rating (green).

are partially overlapped, such as 5 to 4 and 1 to 2 because closer ratings are too similar.

2) Analysis of Similarity: In CM and RIM, closer vectors are relative to meaning similarity. To show this property, we employ three cases and their top five most similar movies retrieved by the cosine similarity between vectors, and the results are showed in Tables II and III for RIM and CM, respectively.

As can be seen in Table II, retrieved movies are similar to their corresponding cases. For example, in the case of *Star Trek: First Contact*, all of movies belong to the series of *Star Trek*. In the case of *Toy story*, all of them are cartoon.

As can be seen in Table III, the results are different with the ones in Table III, but they are also reasonable. In the case of *Star Trek: First Contact*, the retrieved movies also belong to the series of *Star Trek*. In the top five most similar movies of *Toy story*, although only *Toy story 2* and *Aladdin* belong to the animated film, all of them have imaginative fantasy plot.

Consequently, both kinds of embeddings exhibit the characteristics on closer vectors with meaning similarity. In addition, for a item, different kinds of representations have varying perspective which conduces to their complementary property.

C. Rating Prediction Accuracy

1) Evaluation Metric: To measure the performance of prediction, we adopt the most popular metrics RMSE, $\text{RMSE} = \sqrt{(\sum_{i=1}^{S}(r_i - p_i)^2/|S|)}$, where S denotes the test set and $|S|$ denotes the number of ratings in S. Following LLORMA [43], we report the average RMSE on test set over five different splits and compare our methods with some strong baseline methods.

2) Experiment Settings: In both datasets, the dimensions of the vectors of users and items in term of CM and RIM are equal to 400. To effectively train the multiviews prediction neural network using the pretrained vectors, we adopt SGD as the training method and L2-regularization are employed to prevent over-fitting. The number of layers of neural networks is equal to 5 (one input layer, one transformation layer, one merge layer, one hidden layer, and one output layer). The learning rates for both datasets are 0.0005. The bath-size for both datasets are 64. In both datasets, the numbers of cells in the FC layers in both views of current and history are both equal to 100. The numbers of cells in the CONV layers in the views of current and history are 200 and 150, respectively. The number of cells in the final hidden layer is 2000. The ratios of L2 in both datasets are 0.0015 and 0.0005, respectively.

3) Baseline Methods: We compare our model with the following strong baseline algorithms.

1) *Bias Matrix Factorization (BMF) [7]:* It involves the biases of user and item to further improves the performance of traditional MF
2) *LLORMA [43]:* Multiple low-rank submatrices are employed to represent the original rating matrix.
3) *RBM-CF [9]:* RBM makes recommendation via reconstructing the missing value from known values.
4) *Autorec [10]:* Similar to RBM-CF, but the RBM is replaced to *auto-encoder*.
5) *CF-NADE [11]:* CF-NADE predict the rating by NADE which is also an alternative to RBM and it achieves state-of-the-art performance.
6) *NNMF [25]:* A feed-forward neural networks estimate rating from the user and item, which is most similar model with our methods among the baseline methods.
7) *Zanotti's Methods [45]:* It estimates the rating according to the similarity of the neighbors of the features, where the CBOW and Skip-gram models are adopted to training the features of users and items.

TABLE IV
PREDICTION PERFORMANCE COMPARISON ON TWO DATASETS

Type	Method	RMSE on MovieLens 1M	RMSE on MovieLens 10M
Matrix Factorization	**BiasMF** [10]	0.845	0.803
	LLORMA-GLOBAL [43]	0.865	0.822
	LLORMA-LOCAL [43]	0.833	0.782
Feed-forward NN	**NNMF** [25]	0.843	–
latent representations	**Zanotti's IBCB** [45]	–	0.905
	Zanotti's hybrid method [45]	–	0.858
RBM-like	**U-RBM** [10]	0.881	0.823
	I-RBM [10]	0.854	0.825
	U-ATUOREC [10]	0.874	0.867
	I-ATUOREC [10]	0.831	0.782
	U-CF-NADE-S (2 **layers)** [11]	0.845	**0.771**
	I-CF-NADE-S (2 **layers)** [11]	**0.829**	–
Our methods	**multi-views NN**	**0.833**	**0.774**
	multi-views NN+ BiasMF's vectors	**0.830**	**0.776**

8) *Multiview Neural Networks:* It is our basic method, which includes the view of current and history.
9) *Multiview Neural Networks+BiasMF:* It is a special multiview neural networks. But it is has an additional branch on the view of current user and item, which employs the users and items vectors obtained from the pretraining *BiasMF* model.

4) Prediction Performance Comparison: Table IV illustrates experimental results measured by RMSE on two datasets. we show results of our methods, *multiview neural networks* and *multiview neural networks+BiasMF*, and compare them with some strong baseline methods.

In contrast with popular MF methods, our methods outperform or equal to *LLORMA-LOCAL* (state-of-the-art of MF method) in relatively small scale dataset MovieLens 1M. But *multiview neural networks* exceeds *LLORMA-LOCAL* by a relatively big margin (0.008) in the large scale MovieLens 10M. These results show that our methods have impressive performance on different scale datasets because our deep neural networks can capture more complex interaction than the simple vector inner product in MF.

Zanotti combined traditional MF with pretrained user and item vectors, which were trained via *word2vec* (a popular method to generate the word vectors in NLP). However, its best performance only has 0.905 in MovieLens 10M. This result shows that combination of our pretrained embeddings with the neural network is much more effective than Zanotti's model. NNMF replaces the inner product in MF to a feed-forward neural networks, our *multiview neural networks* significantly outperforms it by a large margin, which shows that using pretrained representations with the architecture of multiview can further improve the performance of the method using feed-forward neural networks.

As can be seen in Table IV, RBM-like methods exhibit impressive performances. However, the RBM-like method cannot consistently achieve these results in a certain side. In the methods of RBM-CF, the best performances in MovieLens 1M and MovieLens 10M are acquired from the item side (I-RBM) and user side (U-RBM), respectively. Similarly, the best performances of *CF-NADE* in MovieLens 1M and MovieLens 10M are also obtained from different sides. In contrast with these models, our methods make recommendation from both the user side and the item side, and attain very comparable performance with the best model.

D. Performance on Top-N Task

1) Metric and Experiment Settings: The testing methodology adopted in this section is similar to [46], and the details of experiment are shown as follows. We measure the Top-N performance of our model on Movielens 1M similar to the experiments in Section V-C. Ninety percentage of ratings are randomly selected as the training set, but the test set contains only 5-star ratings of the remaining 10% samples.

We adopt the metric of recall to justify the performance of top-N task. In order to measure recall, we first train the model over the ratings in training set, where the training detail is the same as the one in Section V-C. Afterward, for each item i rated 5-stars by user u in test-set, We randomly select 1000 additional items unrated by user u. Further, we use the trained model first to obtain the estimated ratings of additional 1000 items as well as item i. Then, we form a ranked list l by ordering all the 1001 items according to their predicted ratings from large to small. Finally, We form a top-N recommendation list $l-N$ by picking the N top ranked items from the list l. If item i is in list $l-N$, we have a hit to item i, otherwise we have a miss. Let $\#hits$ be the count of hitting items in test-set, and $|T|$ be the number of ratings in test-set, the overall recall in test-set can be denoted as $recall(N) = (\#hits/|T|)$.

2) Baseline Methods: We compare our method with the baseline methods as follow.
1) *Movie Average [46]:* Recommends top-N items with the highest average rating.
2) *Top Popular [46]:* Recommends top-N items with the largest number of ratings.
3) *BMF:* As shown in Section V-C.
4) *SVD++ [47]:* A hybrid CF method associated with bias MF with item-oriented neighborhood-based CF.

3) Performance Comparision: Fig. 10 reports the performance of different methods on the Movielens 1M. It is clear that our method can significantly outperform the baseline methods in terms of recall, where it can obtain 0.2816 in terms of recall at $N = 10$. The result shows that our method not only has impressive rating prediction accuracy but

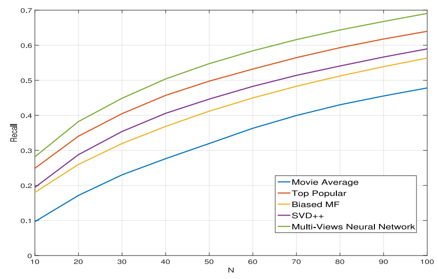


Fig. 10. Movielens 1M : recall at N

TABLE V
MovieLens 1M: RMSE on Different Views

View Type	View	Embedding Model	RMSE
single-view	current user and item	CM	0.846
	current user and item	RIM	0.845
	historical records	CM	0.864
	historical records	RIM	0.835
Multi-views	current user and item	CM and RIM	0.843
	historical records	CM and RIM	0.835
	both views	CM and RIM	**0.833**

also has a good Top-N recommendation accuracy. However, the methods with relatively higher RMSE performance cannot guarantee good performance in terms of top-N accuracy. For example, the biased MF can obtain a good performance 0.845 in terms of RMSE, while it can only achieve 0.1802 in terms of recall at $N = 10$ which is lower than 0.2489 achieved by the simple nonpersonalized method *Top Popular* at $N = 10$.

E. Impact of Different Views

In Table V, we report the rating prediction accuracy of the neural network utilizing various views including single-view and multiviews. In the view of current user and item, we construct two kinds of model which use the CM embedding or the RIM embedding. Likewise, there are also two kinds of embedding model used in the view of history. Note that the models using both CM and RIM embeddings are classified as multiview model. Among these single-view models, the one using RIM embedding on the historical record view is the best among them which can achieve RMSE of 0.835. This performance is close to the result 0.833 which is acquired by the final multiviews method. This suggests that embedding model of RIM on historical records has the greatest impact on the prediction of multiviews neural networks. While joint implementation of RIM and CM embeddings are synthetically considered, their performance (0.843) exceeds the corresponding single-view counterparts which are 0.846 and 0.845, respectively. Although the performance acquired by the method using both RIM and CM embeddings in the view of historical records does not outperform the single-view counterpart with RIM representations, their results are comparably very close. Therefore, the performances of multiviews models are generally better or equal to the counterparts using single view.

TABLE VI
MovieLens 1M: RMSE on Different Transformation Layers

Embedding Type	transformation layer	RMSE
CM	without transformation layer	0.852
	fully connection transformation layer	**0.846**
RIM	without transformation layer	0.853
	local connection transformation layer	0.849
	convolutional transformation layer	**0.845**

TABLE VII
MovieLens 1M: RMSE on Different Embeddings

Embedding Type	random	skip-gram	basic	CM	RIM
RMSE	0.889	0.901	0.862	**0.846**	**0.845**

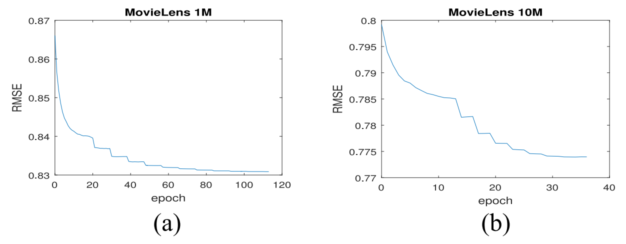


Fig. 11. Convergence curves of our overall model on two datasets.

F. Impact of Transformation Layers

In Table VI, we also report the RMSE performances of the models using different transformation layers. To compare conveniently, the neural networks are only considered in the view of current user and item. In the models using CM embedding, the one without fully connected layer can only achieve 0.852 RMSE value, while the other one with fully connected layer achieves a reduced value of RMSE 0.846. Similarly, among these methods using RIM embedding, the ones using different transformation layers also exceed the one without transformation layer. These results indicate that the method using transformation layer is better than the one without the transformation layer. In addition, the results of using local connection transformation layer and convolutional transformation layer (0.849 and 0.845) suggest that the performance can be improved by replacing the distinct connection layers to the shared connection layers for different ratings.

G. Impact of Types of Embeddings

To investigate the effects on the overall performance from the embeddings standpoint, we present the results of prediction of the neural networks including view of current user and item with different kinds of embeddings in Table VII. First, the result of using random embeddings of user and item is only equal to 0.9672 and it is much lower than the results of our methods, which suggests that the embeddings learned by our models can capture the effective relationship with respect to the users and the items. Skip-gram is a popular model for learning word vectors which are directly addressed of the learning of item and user embeddings [45]. However, its performance is not completive with our methods because it does not take account of the rating information. Similarly, the performance of our basic model without the consideration

TABLE VIII
SCALABILITY OF OUR OVERALL MODEL

Dataset	Params (million)	Train Time (per epoch)	Test Time (all test-set)
ML 1M	7.762	8 minutes	1 minutes
ML 10M	14.7620	2 hours	8 minutes

of rating is also lower than CM and RIM, which explicitly considers the ratings.

H. Complexity and Convergence Analysis

The convergence curves of *multiviews neural networks* on both datasets are illustrated in Fig. 11(a) and (b), respectively, which show that RMSE of *multiviews neural networks* becomes stable after about 80 and 30 iterations on MovieLens 1M and MovieLens 10M, respectively. Thus, *multiviews neural networks* have a satisfying convergence rate and can be trained efficiently.

We trained the *multiviews neural networks* on a single GPU (Titan GTX). Table VIII shows the running time of one epoch and the number of parameters of *multiviews neural networks*. Although training time in our model is relatively large. However, during prediction, the running time is negligible, where it only takes 1 min on 100K samples for 1M dataset and 8 min on 1M for 10M dataset.

VI. Conclusion

This paper presented a novel CF framework based on deep learning. The framework contains two stages: 1) learning low-dimensional embeddings for both users and items and 2) generating predicted ratings by using a multiview feed-forward neural networks. In the future, we intend to improve our method form three aspects: 1) construct a end-to-end neural networks on history view. For example: a) RNN [35] to consider the temporal relations between the history items of user and b) CNN [34] to address overall historical data of users and items; 2) consider the content information, such as text, images, and video via appending additional view of content into our multiview neural networks; and 3) improving the training time of our model.

References

[1] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76–80, Jan./Feb. 2003.

[2] J. Liu, P. Dolan, and E. R. Pedersen, "Personalized news recommendation based on click behavior," in *Proc. ACM 15th Int. Conf. Intell. User Interfaces*, Hong Kong, 2010, pp. 31–40.

[3] R. M. Bell and Y. Koren, "Improved neighborhood-based collaborative filtering," in *Proc. KDD Cup Workshop 13th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2007, pp. 7–14.

[4] Y. Ren, G. Li, J. Zhang, and W. Zhou, "Lazy collaborative filtering for data sets with missing values," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1822–1834, Dec. 2013.

[5] B. Li, X. Zhu, R. Li, and C. Zhang, "Rating knowledge sharing in cross-domain collaborative filtering," *IEEE Trans. Cybern.*, vol. 45, no. 5, pp. 1068–1082, May 2015.

[6] J. D. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *Proc. 22nd Int. Conf. Mach. Learn.*, Bonn, Germany, 2005, pp. 713–719.

[7] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[8] X. Luo *et al.*, "A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 3, pp. 579–592, Mar. 2016.

[9] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering," in *Proc. ACM 24th Int. Conf. Mach. Learn.*, 2007, pp. 791–798.

[10] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proc. ACM 24th Int. Conf. World Wide Web*, Corvallis, OR, USA, 2015, pp. 111–112.

[11] Y. Zheng, B. Tang, W. Ding, and H. Zhou, "A neural autoregressive approach to collaborative filtering," in *Proc. 33rd Int. Conf. Mach. Learn.*, New York, NY, USA, 2016, pp. 764–773.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Columbus, OH, USA, 2014, pp. 580–587.

[14] P. Rodriguez *et al.*, "Deep pain: Exploiting long short-term memory networks for facial expression classification," *IEEE Trans. Cybern.*, to be published.

[15] Y. Wu *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[16] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[17] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Trans. Audio, Speech, Language Process.*, vol. 20, no. 1, pp. 30–42, Jan. 2012.

[18] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–10.

[19] S. Li, J. Kawale, and Y. Fu, "Deep collaborative filtering via marginalized denoising auto-encoder," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manag.*, Melbourne, VIC, Australia, 2015, pp. 811–820.

[20] H. Zhang, T. W. S. Chow, and Q. M. J. Wu, "Organizing books and authors by multilayer SOM," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 12, pp. 2537–2550, Dec. 2015.

[21] A. Van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2643–2651.

[22] H.-T. Cheng *et al.*, "Wide & deep learning for recommender systems," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, Boston, MA, USA, 2016, pp. 7–10.

[23] A. M. Elkahky, Y. Song, and X. He, "A multi-view deep learning approach for cross domain user modeling in recommendation systems," in *Proc. 24th ACM Int. Conf. World Wide Web*, Florence, Italy, 2015, pp. 278–288.

[24] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for YouTube recommendations," in *Proc. 10th ACM Conf. Recommender Syst.*, Boston, MA, USA, 2016, pp. 191–198.

[25] G. K. Dziugaite and D. M. Roy, "Neural network matrix factorization," *arXiv preprint arXiv:1511.06443*, 2015.

[26] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, Sydney, NSW, Australia, 2015, pp. 1235–1244.

[27] S. Deng, L. Huang, G. Xu, X. Wu, and Z. Wu, "On deep learning for trust-aware recommendations in social networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 5, pp. 1164–1177, May 2017.

[28] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[29] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. 31st Int. Conf. Mach. Learn.*, Beijing, China, 2014, pp. 1188–1196.

[30] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Columbus, OH, USA, 2014, pp. 1717–1724.

[31] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, “CNN features off-the-shelf: An astounding baseline for recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Columbus, OH, USA, 2014, pp. 806–813.

[32] R. Ren, T. Hung, and K. C. Tan, “A generic deep-learning-based approach for automated surface inspection,” *IEEE Trans. Cybern.*, to be published.

[33] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[34] S. Ji, W. Xu, M. Yang, and K. Yu, “3D convolutional neural networks for human action recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2013.

[35] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, “Recurrent neural network based language model,” in *Proc. Conf. Int. Speech Commun. Assoc. (INTERSPEECH)*, Chiba, Japan, Sep. 2010, pp. 1045–1048.

[36] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–15.

[37] W. Hu, T. Tan, L. Wang, and S. Maybank, “A survey on visual surveillance of object motion and behaviors,” *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 34, no. 3, pp. 334–352, Aug. 2004.

[38] V. Chambon *et al.*, “What are they up to? The role of sensory evidence and prior knowledge in action understanding,” *PLoS ONE*, vol. 6, no. 2, 2011, Art. no. e17133.

[39] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proc. Conf. Empir. Methods Nat. Lang. Process.*, vol. 14, 2014, pp. 1532–1543.

[40] J. Ngiam *et al.*, “Multimodal deep learning,” in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 689–696.

[41] Y. Zhang, M. J. Er, R. Zhao, and M. Pratama, “Multiview convolutional neural networks for multidocument extractive summarization,” *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3230–3242, Oct. 2017.

[42] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Jul. 2011.

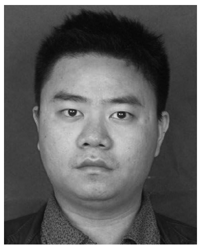
[43] J. Lee, S. Kim, G. Lebanon, and Y. Singer, “Local low-rank matrix approximation,” *J. Mach. Learn. Res.*, vol. 28, no. 2, pp. 82–90, 2013.

[44] L. V. D. Maaten and G. Hinton, “Visualizing data using t-SNE,” *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 2579–2605, 2008.

[45] G. Zanotti, M. Horvath, L. N. Barbosa, V. T. K. G. Immedisetty, and J. Gemmell, “Infusing collaborative recommenders with distributed representations,” in *Proc. 1st ACM Workshop Deep Learn. Recommender Syst.*, 2016, pp. 35–42.

[46] P. Cremonesi, Y. Koren, and R. Turrin, “Performance of recommender algorithms on top-N recommendation tasks,” in *Proc. 4th ACM Conf. Recommender Syst.*, Barcelona, Spain, 2010, pp. 39–46.

[47] Y. Koren, “Factorization meets the neighborhood: A multifaceted collaborative filtering model,” in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, Las Vegas, NV, USA, 2008, pp. 426–434.



Mingsheng Fu is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China.

His current research interests include neural networks, machine learning, and recommendation system.

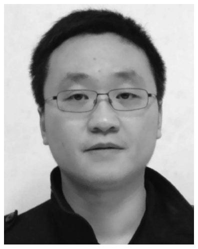
Hong Qu (M’09) received the Ph.D. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2006.

From 2007 to 2008, he was a Post-Doctoral Fellow with the Advanced Robotics and Intelligent Systems Laboratory, School of Engineering, University of Guelph, Guelph, ON, Canada. From 2014 to 2015, he was a Visiting Scholar with the Potsdam Institute for Climate Impact Research, Potsdam, Germany, and with the Humboldt University of Berlin, Berlin, Germany. He is currently a Professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His current research interests include neural networks, machine learning, and big data.

Zhang Yi (F’16) received the Ph.D. degree in mathematics from the Institute of Mathematics, Chinese Academy of Science, Beijing, China, in 1994.

He is currently a Professor with the Machine Intelligence Laboratory, College of Computer Science, Sichuan University, Chengdu, China. He has co-authored three books entitled *Convergence Analysis of Recurrent Neural Networks* (Kluwer Academic Publishers, 2004), *Neural Networks: Computational Models and Applications* (Springer, 2007), and *Subspace Learning of Neural Networks* (CRC Press, 2010). His current research interests include neural networks and big data.

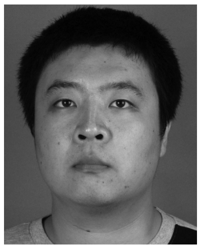
Dr. Zhang was an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, from 2009 to 2012. He is an Associate Editor of the IEEE TRANSACTIONS ON CYBERNETICS in 2014.



Li Lu (M’07) received the Ph.D. degree from the Key Laboratory of Information Security, Chinese Academy of Science, Beijing, China, in 2007.

He is a Professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. His current research interests include RFID technology and system, wireless network, and network security.

Dr. Lu is a member of the IEEE Communication Society and ACM.



Yongsheng Liu is a currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China.

His current research interests include neural networks, machine learning, and recommendation system.