IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 2, 2019, accepted January 30, 2019, date of publication March 13, 2019, date of current version May 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2900698

Deep Probabilistic Matrix Factorization Framework for Online Collaborative Filtering

KANGKANG LI[1], XIUZE ZHOU[2], FAN LIN[1], WENHUA ZENG[1], AND GIL ALTEROVITZ[3]

[1]Software School, Xiamen University, Xiamen 361005, Fujian, China
[2]Department of Automation, Xiamen University, Xiamen 361005, Fujian, China
[3]Harvard Medical School, Computational Health Informatics Program, Boston, MA 02138, USA

Corresponding authors: Fan Lin (iamafan@xmu.edu.cn) and Wenhua Zeng (whzeng@xmu.edu.cn)

ABSTRACT As living data growing and evolving rapidly, traditional machine learning algorithms are hard to update models when dealing with new training data. When new data arrives, traditional collaborative filtering methods have to train their model from scratch. It is expensive for them to retrain a model and update their parameters. Compared with traditional collaborative filtering, the online collaborative filtering is effective to update the models instantly when new data arrives. But the cold start and data sparsity remain major problems for online collaborative filtering. In this paper, we try to utilize the convolutional neural network to extract user/item features from user/item side information to address these problems. First, we proposed a deep bias probabilistic matrix factorization (DBPMF) model by utilizing the convolutional neural network to extract latent user/item features and adding the bias into probabilistic matrix factorization to track user rating behavior and item popularity. Second, we constrain user-specific and item-specific feature vectors to further improve the performance of the DBPMF. Third, we update two models by an online learning algorithm. The extensive experiments for three datasets (MovieLens100K, MovieLens1M, and HetRec2011) show that our methods have a better performance than baseline approaches.

INDEX TERMS Deep learning, deep learning-based recommender systems, online collaborative filtering, probabilistic matrix factorization.

I. INTRODUCTION

Nowadays, it is difficult for people to make reasonable decision in the face of a variety of choices. In order to accurately predict what customers need, effective methods are needed to analyze data. The recommender system is one of solutions to help companies decide what should show to different users to help users quickly find items that match their interests [1]. Many corporations have tried to design different models to apply to recommender system. For example, [2] and [3] use recommender systems to discover new songs for users. Social network use recommender systems to recommend like-minded friends [4].

One class of successful methods applied to recommender system is collaborative filtering (*CF*) [5], [6]. The key idea of *CF* is that if two users have similar preferences in the past, they will also like similar products in the future [7]. *CF* methods do not involve any information of items and users, but instead make recommendations only by using past interactive information, such as users' explicit ratings or implicit comments on items. *CF* methods contain many machine learning approaches. The matrix factorization (*MF*) method is the most popular one applied to recommender systems [8]. Also, probabilistic matrix factorization (*PMF*) is widely applied in collaborative filtering [9].

In recent years, deep learning has achieved some success in recommendation system. For example, Cataldo Musto *et al.* adopt word2vec to make recommendation, which uses textual features extracted from Wikipedia to learn user profiles [10]. Donghyun Kim *et al.* [11] applied convolutional matrix factorization to improve performance of recommendation systems. Wang *et al.* [12] proposed a collaborative deep learning (*CDL*) method, which combines deep learning and *CF* for the ratings. Elkahky *et al.* [13] applied a deep learning method, combining features from all domains produces, in cross domain user modeling. Liu *et al.* [14] proposed a probabilistic model of hybrid deep collaborative filtering for recommender systems.

However, traditional *CF* methods and deep learning train their models in a batch learning algorithm or train entire data

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Afzal.

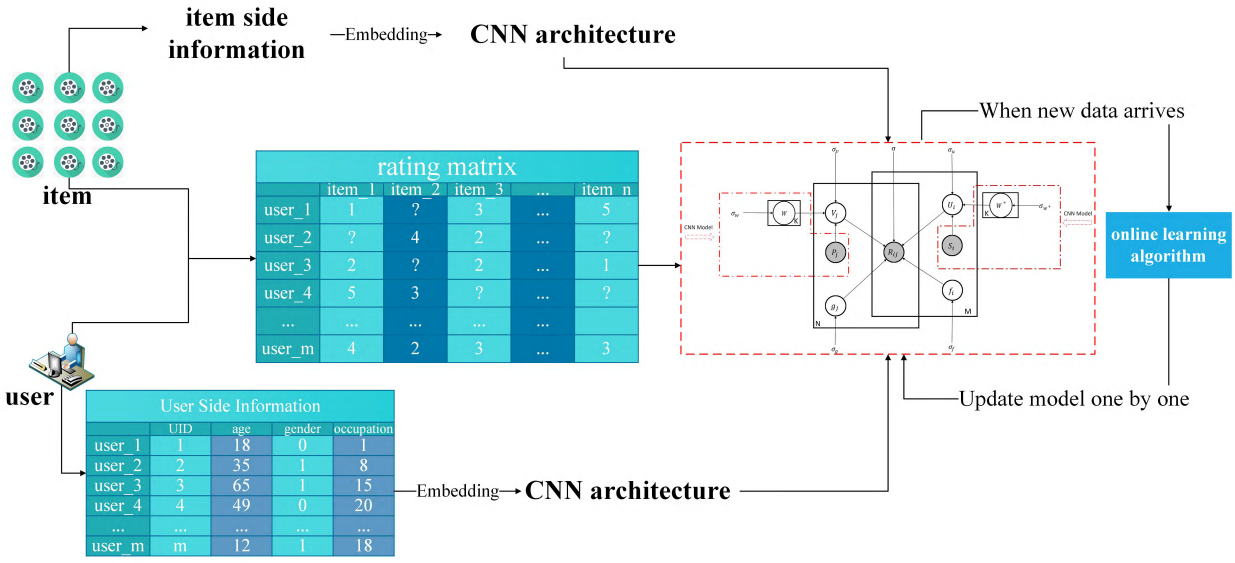


FIGURE 1. The framework of DBPMF.

set at once [15]. In the era of big data, continuous data streams are stored in the system from time to time. User needs and preferences are changing all the time. When new data arrives, traditional *CF* methods and deep learning have to train their model from scratch. It is expensive for them to retrain a model and update their parameters [16].

To alleviate these problems, a series of online learning algorithm have been proposed. Online learning algorithm is a method of machine learning for sequential data [15], [17]. It learns and updates the best predictor for future data at every step. Many online collaborative filtering (*OCF*) [18]–[20] methods are proposed. Such as, Jacob Abernethy *et al.* proposed a method to learn a rank-k matrix factorization for *OCF* tasks [17]. Wang *et al.* [31] exploited a principle similar to that of online multitask learning for *OCF*, which optimized the model by Semi-definite Programming. Zhou *et al.* [20] added confidence-weighted learning method into *OCF* to improve the accuracy. Liu *et al.* [32] proposed the online Bayesian inference algorithm by incorporating content information into *OCF*. The key idea of *OCF* methods is that they update the model on a sequence of rating data when they receive a new instance [15].

In this paper, we proposed a method called Deep Bias PMF (*DBPMF*) for OCF. The framework of *DBPMF* is shown in Figure 1. Firstly, we use convolutional neural network (*CNN*) model to extract user and item latent features. Then we add the bias to *PMF* for tracking user preference and item attributes. Because the bias can help *CF* algorithm to proofread model and to represent features better [20]. Specifically, user-bias can track user's rating behavior and interests; item-bias can track item's population and attributes. Both factors are added into *PMF* to improve the accuracy of *PMF*. Secondly, we proposed an improved approach which is called Deep Constrain Bias PMF (*DCBPMF*) based on *DBPMF*.

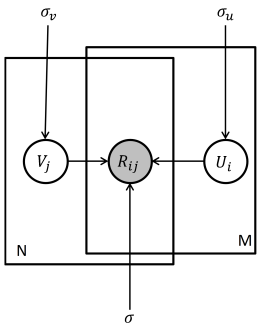


FIGURE 2. The graphical model for PMF.

When ratings are few, the predicted ratings from a fitted *PMF* model will close to the average ratings and user and item feature vectors will close to the prior mean. Our improved method that constrains users and items feature vectors can exert great impact on infrequent users and items.

II. BACKGROUND AND RELATED WORK

A. PROBABILISTIC MATRIX FACTORIZATION AND PROBLEM DESCRIPTION

PMF is one of the most successful techniques among all *CF* methods [14], [21]. It is used to generate potential space to represent users or items. In the potential space, the recommender systems calculate the similarity between users and items, thus obtaining the possible ranking of a series of items to make recommendation for users.

Based on observed rating data, it gets two low-rank matrices, whose product are used to predict the unknown ratings. The graphical model for *PMF* shows in Figure 2.

Given a recommender system with M users and N items, $R \in \mathbb{R}^{M \times N}$ denotes the user-item rating matrix, where R_{ij}

represents the rating of user i on item j. $U \in \mathbb{R}^{K\times M}$ and $V \in \mathbb{R}^{K\times N}$ denote the latent user and item feature matrix. U_i denotes the latent feature vector of user, i; V_j denotes the latent feature vector of item, j.

From the probabilistic point of view, the conditional distribution over observed rating as

$$P(R|U, V, \sigma^2) = \prod_{i=1}^{M}\prod_{j=1}^{N}[N(R_{ij}|U_i^{\mathrm{T}}V_j, \sigma^2)]^{I_{ij}} \tag{1}$$

where $N(x|\mu, \sigma^2)$ is the probability density function of the Gaussian distribution with mean, μ and variance, σ^2, and I_{ij} is the indicator function as

$$I_{ij} = \begin{cases} 1, & \text{if user } i \text{ has rated item } j \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

We assume that $U_i \sim N(0, \sigma_u^2 I)$, $V_j \sim N(0, \sigma_v^2 I)$. Then we get the posterior distribution over user and item feature as

$$P(U, V|R, \sigma^2, \sigma_u^2, \sigma_v^2) \tag{3}$$

Finally, the objective function defined as following:

$$E = \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}(R_{ij} - U_i^{\mathrm{T}}V_j)^2 + \frac{\lambda_u}{2}\sum_{i=1}^{M}\|U_i\|_F^2 + \frac{\lambda_v}{2}\sum_{j=1}^{N}\|V_j\|_F^2 \tag{4}$$

where $\lambda_u = \frac{\sigma^2}{\sigma_u^2}$, $\lambda_v = \frac{\sigma^2}{\sigma_v^2}$, and $\|\cdot\|_F^2$ denotes the Frobenius norm.

This is the formal formulation of *CF* problem. The goal of *CF* task is to minimize the loss function. Then, it predicts the unknown rating based on all of the previous observed ratings of user, u.

We adopt Root Mean Square Error (*RMSE*) as predictive accuracy measures, which is defined as following:

$$RMSE = \sqrt{\frac{1}{|C|}\sum_{(i,j)\in C}(R_{ij} - \hat{R}_{ij})^2} \tag{5}$$

where R_{ij} denotes the observed rating, and $\hat{R}_{ij}$ denotes the predicted rating. C is the set of all observed ratings, and $|C|$ is the number of the set C.

To optimize the *RMSE* metric, the loss of this model is defined as:

$$L_1 = (R_{ij} - U_i^{\mathrm{T}}V_j)^2 \tag{6}$$

B. CONVOLUTIONAL NEURAL NETWORK BASED RECOMMENDER SYSTEM

Recently, convolution neural network (*CNN*) has been applied into recommender system to extract user and item features [11], [14]. For example, Gong and Zhang [22] design an attention *CNN* model to recommender hashtag in microblog. The model use *CNN* as global channel to encode input words and utilize attention layer as local channel to select informative words. Kim *et al.* [11] proposed a *ConvMF* model to combine *CNN* with *PMF*. The *ConvMF* model mainly employs *CNN* to extract latent item features. The objective function of *ConvMF* defined as:

$$E = \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}(R_{ij} - U_i^{\mathrm{T}}V_j)^2 + \frac{\lambda_u}{2}\sum_{i=1}^{M}\|U_i\|_F^2 + \frac{\lambda_v}{2}\sum_{j=1}^{N}\|V_j - cnn(W, P_j)\|_F^2 + \frac{\lambda_W}{2}\sum_{k}^{|W_k|}\|W_k\|_F^2 \tag{7}$$

where $\lambda_u, \lambda_v, \lambda_W$ are hyper-parameters. It adopts coordinate descent to optimize parameters.

The *ConvMF* just uses *CNN* to extract latent item features and ignores latent user features. The model employs batch learning algorithm to update their model, which may lead to expensive re-training cost and poor scalability when new data arrives. So we try to employ *CNN* to extract latent user and item features and to utilize *OCF* methods to improve the efficiency and scalability.

C. OCF METHODS

Traditional *CF* methods have their own drawbacks, such as the cold start problem: there is no way to recommend items that have not yet received user ratings. When the rating data is very sparse, the accuracy of its predictions will decline dramatically [23]. When the number of users and items increase sharply, new challenges will be posed to the performance of these *CF* methods.

The existing *CF* methods have some limitations. First, they adopt batch learning methods, which is expensive for them to retrain their models, for they require all data available before they begin to retrain [15]. Second, they fail to capture the drift of user interest and the changing popularity of the items [24].

To overcome the limitations of tradition batch learning methods, some online learning methods proposed. Recently, online learning is hot in machine learning community. like online multi-task *CF* [18], dual-averaging online *PMF* [8], and second-order *OCF* [19]. Those methods obtained faster convergence speed based on better update strategies. And it makes more sense for large-scale applications when data arrives sequentially [25].

Online learning avoids retraining from scratch, for it can learning incrementally from data streams. It is more efficient to update its model. In online learning, user's tastes and item's popularity are changing. To capture both changes, adding bias into *OCF* is a wise strategy. At each round, the model makes prediction when it receives a new instance.

With the development of online learning, many improved *OCF* methods [26], [27] were proposed. Online low-rank (*OLR*) approximation method is the most simple and well-known methods in *OCF* [15]. It used Stochastic Gradient Descent (*SGD*) to obtain the low-rank matrices. *SGD* is simply and widely used in convex optimization of online learning problems. Wang *et al.* [28] exacted soft confidence-weighted learning for *OCF*. Lin *et al.* [29] proposed a method for sparse

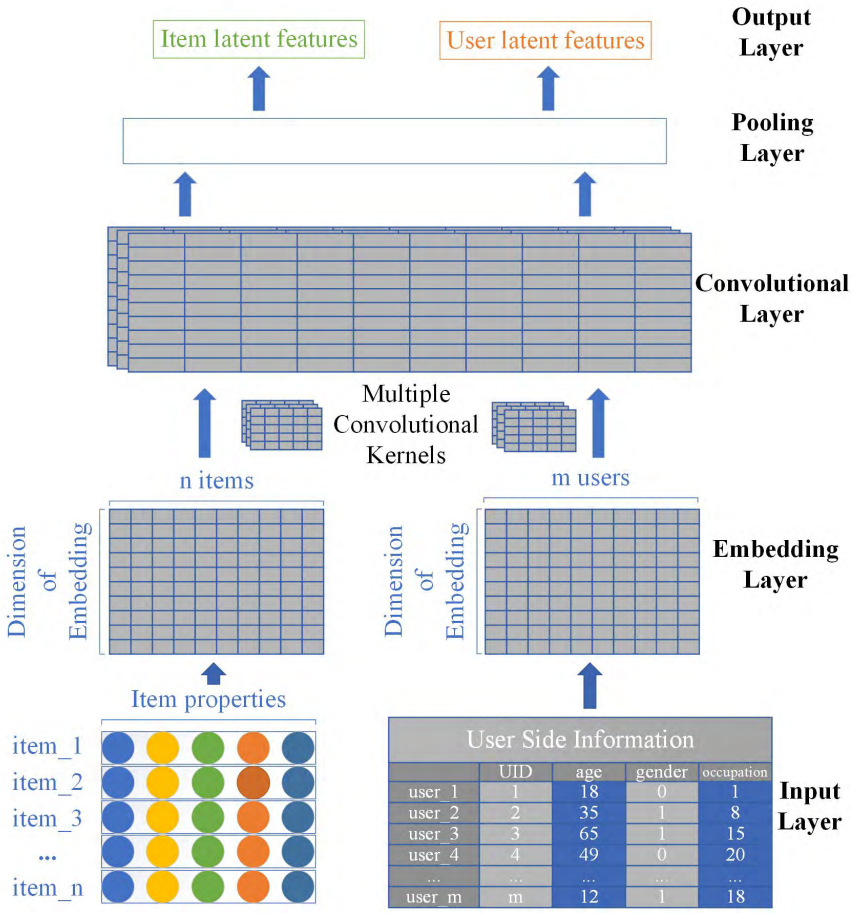


FIGURE 3. The CNN architecture of DBPMF and DCBPMF.

problem which would have seriously affect the performance of algorithms. Many second-order approaches were proposed to speed up the convergence of the optimization by using both the first-order and second-order information [19], [24].

III. MODELS

A. CONVOLUTIONAL NEURAL NETWORK OF DBPMF AND DCBPMF

We utilize *CNN* architecture to extract latent user and item features from user side information and item side information. Figure 3 shows our *CNN* architecture, which consists of four layers: embedding layer, convolution layer, pooling layer and output layer.

1) EMBEDDING LAYER

The main purpose of the embedding layer is to encode user and item side information as two matrices by one-hot. Item side information matrix $D_{item} \in \mathbb{R}^{p\times n}$ and user side information matrix $D_{user} \in \mathbb{R}^{s\times m}$ can be visualized as following:

$$\begin{bmatrix} W_{11} & \cdots & W_{1n} \\ \vdots & \ddots & \vdots \\ W_{p1} & \cdots & W_{pn} \end{bmatrix} \begin{bmatrix} W_{11} & \cdots & W_{1m} \\ \vdots & \ddots & \vdots \\ W_{s1} & \cdots & W_{sm} \end{bmatrix}$$

where n is the number of item, p is the length of item one-hot vector, m is the number of user, s is the length of user one-hot vector.

2) CONVOLUTION LAYER

The convolution layer is used to extract user and item features. User i feature $c^j_{user_i} \in R$ is extracted by jth convolution kernel $W^j_c \in \mathbb{R}^{m\times ws}$, ws is window size of convolution kernel. $c^j_{user_i} \in R$ is formulated as follows:

$$c^j_{user_i} = f(W^j_c * D_{user}(:, i : (i + ws - 1)) + b^j_c) \tag{8}$$

Also, item i feature $c^j_{item_i} \in R$ is extracted by jth shared weight $W^j_c \in \mathbb{R}^{m\times ws}$, ws is window size of convolution kernel. $c^j_{item_i} \in R$ is formulated as follows:

$$c^j_{item_i} = f(W^j_c * D_{item}(:, i : (i + ws - 1)) + b^j_c) \tag{9}$$

where $*$ is a convolution operator, b^j_c is a bias for W^j_c, $f()$ is rectified linear unit($ReLU$), which is a nonlinear activation function and can avoid vanishing gradient. Then, the user feature vector $c^j_{user} \in \mathbb{R}^{m-ws+1}$ and item feature vector $c^j_{item} \in \mathbb{R}^{n-ws+1}$ of side information with convolution kernel

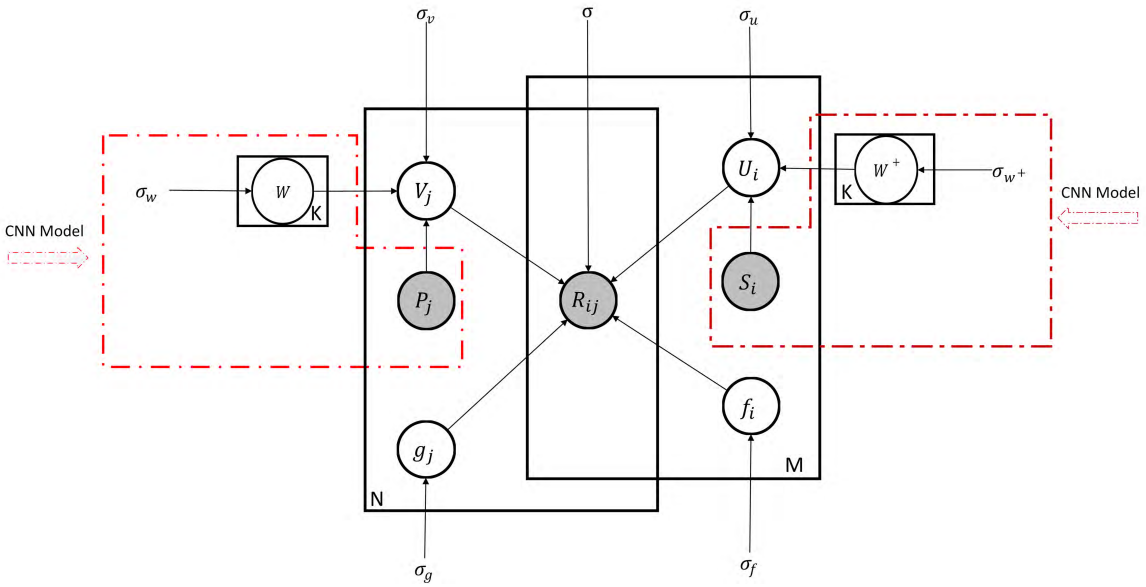


FIGURE 4. The graphical model for DBPMF.

W_c^j are constructed by:

$$c^j_{user} = [c^j_{user_1}, c^j_{user_2}, \cdots, c^j_{user_i}, \cdots, c^j_{user_{m-ws+1}}] \quad (10)$$

$$c^j_{item} = [c^j_{item_1}, c^j_{item_2}, \cdots, c^j_{item_i}, \cdots, c^j_{item_{n-ws+1}}] \quad (11)$$

We use multiple convolution kernels to capture multiple types of latent features. For example, we choose the number n_c of W_c (i.e., W_c^j where $j = 1, 2, \ldots, n_c$), which will generate the number n_c of user and item feature vectors respectively.

3) POOLING LAYER

The main purpose of the pooling layer is to reduce the user and item feature vectors and to fix vectors length by extracting the maximum feature from each feature vector, which is shown as follows:

$$d_{f\_user} = [max(c^1_{user}), max(c^2_{user}), \cdots, max(c^j_{user}), \cdots, max(c^{n_c}_{user})] \quad (12)$$

$$d_{f\_item} = [max(c^1_{item}), max(c^2_{item}), \cdots, max(c^j_{item}), \cdots, max(c^{n_c}_{itemr})] \quad (13)$$

where c^j_{user} and c^j_{item} are feature vectors extracted by *jth* convolution kernel W_c^j respectively.

4) OUTPUT LAYER

In output layer, we adopt conventional nonlinear function to transform d_{f_user} and d_{f_item} to a k dimensional space of users and items' latent factors for our recommendation task. Transformation formula are as follows:

$$S_{user} = tanh(W_{f_2}\{tanh(W_{f_1} d_{f_{user}} + b_{f_1})\} + b_{f_2}) \quad (14)$$

$$S_{item} = tanh(W_{f_4}\{tanh(W_{f_3} d_{f_{item}} + b_{f_3})\} + b_{f_4}) \quad (15)$$

where $W_{f_1} \in \mathbb{R}^{f_{user} \times n_c}$, $W_{f_2} \in \mathbb{R}^{k \times f_{user}}$, $W_{f_3} \in \mathbb{R}^{f_{item} \times n_c}$, $W_{f_4} \in \mathbb{R}^{k \times f_{item}}$ are projection matrices. $b_{f_1}, b_{f_2}, b_{f_3}, b_{f_4}$ are bias vectors. Finally, our *CNN* architecture takes one user or item one-hot vector as input, and returns a latent user or item vector as output:

$$S_{user_i} = cnn(W^+, S_i) \quad (16)$$

$$S_{item_j} = cnn(W, P_j) \quad (17)$$

where W^+ and W denote all the weight and bias variables of *CNN*, S_i represents one-hot vector of user i side information. P_j denotes one-hot vector of item j side information, S_{user_i} and S_{item_j} represent user i and item j latent vector.

B. DBPMF FOR OCF

We have tried to use *CNN* model to extract user and item latent features. Then, we add the bias into *PMF* to track the user's rating behavior and the item's popularity. Bias can help *CF* algorithm to proofread model and represent features better. Specifically, user-bias can track user's rating behavior and interests; item-bias can track item's population and attributes.The graphical model for *DBPMF* shows in Figure 4.

From the probabilistic point of view, the conditional distribution over observed rating as

$$P(R|U, V, f, g, \sigma^2) = \prod_{i=1}^{M}\prod_{j=1}^{N}[N(R_{ij}|U_i^{\mathrm{T}}V_j + f_i + g_j, \sigma^2)]^{I_{ij}} \quad (18)$$

where $N(x|\mu, \sigma^2)$ is the probability density function of the Gaussian distribution with mean, μ and variance, σ^2, and I_{ij} is the indicator function. We assume that $f_i \sim N(x|0, \sigma_f^2 I)$, $g_j \sim N(x|0, \sigma_g^2 I)$, f_i denotes user-bias,σ_f denotes the observed deviation of user i. g_j denotes item-bias,σ_g denotes the observed deviation of item j. Then we

get the posterior distribution over user and item features as $P(U, V, f, g|R, \sigma^2, \sigma_u^2, \sigma_v^2, \sigma_f^2, \sigma_g^2)$.

A user's latent feature consists of three parts: internal weights W^+ in *CNN*, S_i which is one-hot vector of user i side information, and ε_i variable as Gaussian noise, which is applied to optimize the user's latent factor for the rating. Thus, the final user's latent factor can be generated by the following equations.

$$U_i = cnn(W^+, S_i) + \varepsilon_i \tag{19}$$

where $cnn()$ represents the output of *CNN* architecture and $\varepsilon_i \sim N(0, \sigma_u^2 I)$. For each weight w_k^+ in W^+, we set zero-mean spherical Gaussian prior.

$$P(w^+|\sigma_{W^+}^2) = \prod_k N(w_k^+|0, \sigma_{W^+}^2) \tag{20}$$

$$P(U|W^+, S, \sigma_u^2) = \prod_i^M N(u_i|cnn(W^+, S_i), \sigma_u^2 I) \tag{21}$$

Similarly, an item's latent factor consists of three variables: internal weights W in *CNN*, P_j representing one-hot vector of item j side information, and ε_j variable as Gaussian noise, which is used to further optimize an item's latent factor for rating. Hence, the final item's latent factor can be given by the following equations.

$$V_j = cnn(W, P_j) + \varepsilon_j \tag{22}$$

where $cnn()$ represents the output of *CNN* architecture and $\varepsilon_j \sim N(0, \sigma_v^2 I)$. For each weight w_k in W, we set zero-mean spherical Gaussian prior.

$$P(w|\sigma_W^2) = \prod_k N(w_k|0, \sigma_W^2) \tag{23}$$

$$P(V|W, P, \sigma_v^2) = \prod_j^N N(v_j|cnn(W, P_j), \sigma_v^2 I) \tag{24}$$

Finally, the objective function defined as follows:

$$\begin{aligned} E = &\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}(R_{ij} - U_i^{\mathrm{T}}V_j - f_i - g_j)^2 \\ &+\frac{\lambda_u}{2}\sum_{i=1}^{M}\|U_i - cnn(W^+, S_i)\|_F^2 \\ &+\frac{\lambda_v}{2}\sum_{j=1}^{N}\|V_j - cnn(W, P_j)\|_F^2 \\ &+\frac{\lambda_{W^+}}{2}\sum_{k}^{|W_k^+|}\|W_k^+\|_2^2 + \frac{\lambda_W}{2}\sum_{k}^{|W_k|}\|W_k\|_2^2 \\ &+\frac{\lambda_f}{2}\sum_{i=1}^{M} f_i^2 + \frac{\lambda_g}{2}\sum_{i=1}^{N} g_j^2 \end{aligned} \tag{25}$$

in which $\lambda_u = \frac{\sigma^2}{\sigma_u^2}$, $\lambda_v = \frac{\sigma^2}{\sigma_v^2}$, $\lambda_{W^+} = \frac{\sigma^2}{\sigma_{W^+}^2}$, $\lambda_W = \frac{\sigma^2}{\sigma_W^2}$, $\lambda_f = \frac{\sigma^2}{\sigma_f^2}$, $\lambda_g = \frac{\sigma^2}{\sigma_g^2}$, and $\|\cdot\|_F^2$ denotes the Frobenius norm.

Algorithm 1 DBPMF for OCF

Input: a sequence of rating pairs (i, j, R_{ij}), user i side information S_i and item j side information P_j

Output: a predicted rating $\hat{R}_{ij}$;

1: Initialize random matrix for CNN weight W, W^+ and $f \in R^{k\times m}$, $g \in R^{k\times n}$ respectively
2: Encode user side information S and item side information P as one-hot vector. Input S and P into CNN model and output latent user features $U \in R^{k\times m}$ and latent item features $V \in R^{k\times n}$.
3: **for** $t = 1, 2, \ldots, T$ **do**
4: Receive rating prediction request of user i on item j
5: Make prediction $\hat{R}_{ij} = U_i^{\mathrm{T}}V_j + f_i + g_j$
6: The algorithm suffers a loss $l(U_i, V_j, f_i, g_j, R_{ij})$. The true rating R_{ij} is revealed
7: Update U_i, V_j, f_i and g_j according to: (26), (27), (28) and (29) respectively
8: Update W and W^+ according to the back propagation algorithm
9: **end for**

We use *SGD* to optimize Eq. 25:

$$U_i = (1 - \eta\lambda_u)\cdot U_i + \eta e_{ij}\cdot V_j + \eta\lambda_u \cdot cnn(W^+, S_i) \tag{26}$$

$$V_j = (1 - \eta\lambda_v)\cdot V_j + \eta e_{ij}\cdot U_i + \eta\lambda_v \cdot cnn(W, P_j) \tag{27}$$

$$f_i = (1 - \eta\lambda_f) f_i + \eta e_{ij} \tag{28}$$

$$g_j = (1 - \eta\lambda_g) g_j + \eta e_{ij} \tag{29}$$

where $e_{ij} = R_{ij} - \hat{R}_{ij}$.When receiving a new rating, the model will make prediction: $\hat{R}_{ij} = U_i^{\mathrm{T}}V_j + f_i + g_j = (cnn(W^+, S_i) + \varepsilon_i)^{\mathrm{T}}(cnn(W, P_j) + \varepsilon_j) + f_i + g_j$. However, there is no an analytic solution to optimize W^+ and W [11], [14]. But Eq. 25 can be interpreted as a squared error function with L_2 regularized terms as follows when U, V, f, g are temporarily constant.

$$\begin{aligned} \Phi(W^+) = &\frac{\lambda_u}{2}\sum_{i=1}^{M}\|U_i - cnn(W^+, S_i)\|_F^2 \\ &+\frac{\lambda_{W^+}}{2}\sum_{k}^{|W_k^+|}\|W_k^+\|_2^2 + constant \end{aligned} \tag{30}$$

$$\begin{aligned} \Phi(W) = &\frac{\lambda_v}{2}\sum_{j=1}^{N}\|V_j - cnn(W, P_j)\|_F^2 \\ &+\frac{\lambda_W}{2}\sum_{k}^{|W_k|}\|W_k\|_2^2 + constant \end{aligned} \tag{31}$$

so we adopt the back propagation algorithm to optimize them. Finally, Algorithm 1 shows the detailed work of the *DBPMF*.

C. DCBPMF FOR OCF

Once a *PMF* model has been trained, the users or items feature vectors will close to the prior mean when they have few

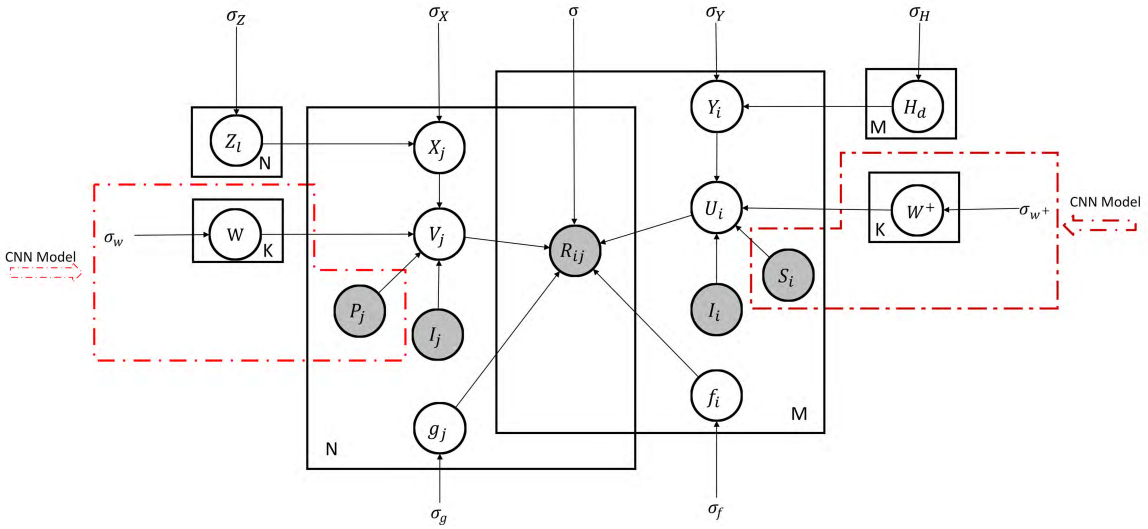


FIGURE 5. The graphical model for DCBPMF.

ratings. So, constraining user-specific and item-specific feature vectors is an effective way to address this problem. This section we proposed deep constrain bias *PMF* to constrain user-specific and item-specific feature vectors.The graphical model for *DCBPMF* shows in Figure 5.

In *DCBPMF*, we defined U_i and V_j as follows:

$$U_i = cnn(W^+, S_i) + \varepsilon_i + Y_i + \frac{\Sigma_{d=1}^{M} I_{id} H_d}{\Sigma_{d=1}^{M} I_{id}} \tag{32}$$

$$V_j = cnn(W, P_j) + \varepsilon_j + X_j + \frac{\Sigma_{l=1}^{N} I_{lj} Z_l}{\Sigma_{l=1}^{N} I_{lj}} \tag{33}$$

where $cnn()$ represents the output of *CNN* architecture and $\varepsilon_i \sim N(0, \sigma_u^2 I), \varepsilon_j \sim N(0, \sigma_v^2 I)$. W^+ and W denote all the weight and bias variables of *CNN*, S_i represents one-hot vector of user i side information. P_j denotes one-hot vector of item j side information. H_d, the dth item of user i having rated, is used to track the effect of the user having rated a particular item on the prior mean of his feature vector. Y_i can be seen as the offset added to the mean of the prior distribution to get the feature vector U_i for the user i. Z_l, the lth user has rated the item j, is used to track the effect of the item having rated by a particular user on the prior mean of the its feature vector. X_j can be seen as the offset added to the mean of the prior distribution to get the feature vector V_j for the item j. And I_{ij} is the indicator function.

The objective function of *DCBPMF* defined as follows:

$$\begin{aligned} E = & \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{N} I_{ij} (R_{ij} - U_i^{\mathrm{T}} V_j - f_i - g_j)^2 \\ & + \frac{\lambda_u}{2} \sum_{i=1}^{M} \| U_i - cnn(W^+, S_i) - Y_i - \frac{\Sigma_{d=1}^{M} I_{id} H_d}{\Sigma_{d=1}^{M} I_{id}} \|_F^2 \\ & + \frac{\lambda_v}{2} \sum_{j=1}^{N} \| V_j - cnn(W, P_j) - X_j - \frac{\Sigma_{l=1}^{N} I_{lj} Z_l}{\Sigma_{l=1}^{N} I_{lj}} \|_F^2 \\ & + \frac{\lambda_{W^+}}{2} \sum_{k}^{|W_k^+|} \| W_k^+ \|_2^2 + \frac{\lambda_W}{2} \sum_{k}^{|W_k|} \| W_k \|_2^2 \\ & + \frac{\lambda_f}{2} \sum_{i=1}^{M} f_i^2 + \frac{\lambda_g}{2} \sum_{i=1}^{N} g_j^2 \\ & + \frac{\lambda_Y}{2} \sum_{j=1}^{M} \| Y_i \|_F^2 + \frac{\lambda_H}{2} \sum_{d=1}^{M} \| H_d \|_F^2 \\ & + \frac{\lambda_X}{2} \sum_{j=1}^{N} \| X_j \|_F^2 + \frac{\lambda_Z}{2} \sum_{l=1}^{N} \| Z_l \|_F^2 \end{aligned} \tag{34}$$

in which $\lambda_Y = \frac{\sigma^2}{\sigma_Y^2}, \lambda_X = \frac{\sigma^2}{\sigma_X^2}, \lambda_H = \frac{\sigma^2}{\sigma_H^2}, \lambda_Z = \frac{\sigma^2}{\sigma_Z^2}$, other parameters are same to *DBPMF*.

We use *SGD* to optimize Eq. 34:

$$\begin{aligned} U_i = & (1 - \eta\lambda_u) \cdot U_i + \eta e_{ij} \cdot V_j \\ & + \eta\lambda_u \cdot (cnn(W^+, S_i) + Y_i + \frac{\Sigma_{d=1}^{M} I_{id} H_d}{\Sigma_{d=1}^{M} I_{id}}) \end{aligned} \tag{35}$$

$$\begin{aligned} V_j = & (1 - \eta\lambda_v) \cdot V_j + \eta e_{ij} \cdot U_i \\ & + \eta\lambda_v \cdot (cnn(W, P_j) + X_j + \frac{\Sigma_{l=1}^{N} I_{lj} Z_l}{\Sigma_{l=1}^{N} I_{lj}}) \end{aligned} \tag{36}$$

$$\begin{aligned} Y_i = & (1 - \eta\lambda_Y + \eta\lambda_u) Y_i \\ & - \eta\lambda_u (U_i - cnn(W^+, S_i) - \frac{\Sigma_{d=1}^{M} I_{id} H_d}{\Sigma_{d=1}^{M} I_{id}}) \end{aligned} \tag{37}$$

$$\begin{aligned} X_j = & (1 - \eta\lambda_X + \eta\lambda_v) X_j \\ & - \eta\lambda_v (V_j - cnn(W, P_j) - \frac{\Sigma_{l=1}^{N} I_{lj} Z_l}{\Sigma_{l=1}^{N} I_{lj}}) \end{aligned} \tag{38}$$

$$H_d = (1 - \eta\lambda_H) H_d$$

$$+\frac{\eta e_{ij}}{\Sigma_{d=1}^{M} I_{id}}(U_i - cnn(W^+, S_i) - Y_i - \frac{\Sigma_{d=1}^{M} I_{id} H_d}{\Sigma_{d=1}^{M} I_{id}}) \tag{39}$$

$$Z_l = (1 - \eta\lambda_Z)Z_l + \frac{\eta e_{ij}}{\Sigma_{l=1}^{N} I_{lj}}(V_j - cnn(W, P_j) - X_j - \frac{\Sigma_{l=1}^{N} I_{lj} Z_l}{\Sigma_{l=1}^{N} I_{lj}}) \tag{40}$$

$$f_i = (1 - \eta\lambda_f)f_i + \eta e_{ij} \tag{41}$$

$$g_j = (1 - \eta\lambda_g)g_j + \eta e_{ij} \tag{42}$$

When receiving a new rating, the model will make prediction:

$$\hat{R}_{ij} = U_i^{\mathrm{T}} V_j + f_i + g_j \tag{43}$$

Similar to *DBPMF*, we adopt the back propagation algorithm to optimize W^+, W. Finally, Algorithm 2 shows the detailed work of the *DCBPMF*.

Algorithm 2 DCBPMF for OCF

Input: a sequence of rating pairs (i, j, R_{ij}), user i side information S_i and item side j information P_j

Output: a predicted rating $\hat{R}_{ij}$;

1: Initialize random matrix for CNN weight W, W^+, and $Y \in R^{k\times m}, X \in R^{k\times n}, H \in R^{k\times m}, Z \in R^{k\times n}$ and $f \in R^{k\times m}, g \in R^{k\times n}$ respectively
2: Encode user side information $S$and item side information P as one-hot vector. Input S and P into CNN model and output latent user features $U \in R^{k\times m}$ and latent item features $V \in R^{k\times n}$.
3: **for** $t = 1, 2, \ldots, T$ **do**
4: Receive rating prediction request of user i on item j
5: Make prediction $\hat{R}_{ij} = U_i^{\mathrm{T}} V_j + f_i + g_j$
6: The algorithm suffers a loss $l(U_i, V_j, f_i, g_j, R_{ij})$. The true rating R_{ij} is revealed
7: Update $U_i, V_j, Y_i, X_j, H_d, Z_l, f_i$ and g_j according to: (35) to (42) respectively
8: Update W and W^+ according to the back propagation algorithm
9: **end for**

IV. EXPERIMENTS

In this section, we perform several experiments to compare the quality of our proposed method to that of the baseline approaches. We then analyze the experimental results in detail.

A. BASELINE APPROACHES AND EXPERIMENTAL SETTING

To evaluate our proposed methods, we compared them with several baselines, which are listed as follows:

- **OLR**: online low-rank approximation, which learns a rank-k MF by using online gradient descent to optimize the loss function directly [15];
- **ConvMF**: convolution PMF, which combines CNN with PMF to capture contextual information of item via word embedding and convolutional kernels [11];
- **SOCF_II**: second-order sparse OCF, which adds an absolute term to the objective function [29];

TABLE 1. The detail information of datasets.

Datasets	#Ratings	#Users	#Items	density
MovieLens100K	100,000	943	1,682	6.3%
MovieLens1M	1,000,209	6,040	3,900	4.2%
HetRec2011	855,598	2,113	10,109	4.0%

TABLE 2. The performance on MovieLens100K.

Method \ k	3	5	7	9	11	13
OLR	1.2355	1.1238	1.0482	1.0126	1.0243	1.0499
ConvMF	1.0496	1.0187	1.0006	1.0101	1.0253	1.0475
SOCF_II	1.0979	1.0398	0.9952	0.9997	1.0212	1.0445
DBPMF	1.0404	1.0028	0.9901	0.9982	1.0206	1.0434
DCBPMF	**1.0289**	**0.9916**	**0.9856**	**0.9938**	**1.0128**	**1.0388**

We conducted experiments on three public datasets: MovieLens100K, MovieLens1M and HetRec2011. All datasets are available from the MovieLens website (https://grouplens.org/datasets/movielens/) [30]. The detail information of those datasets shows in Table 1. We choose *RMSE* as our metric to evaluate the performance of our methods.

In *CNN* architecture, the user side information consists of users' ID, gender, age and occupation. Users' gender is encoded as 0 and 1. 0 represents male and 1 represents female. Users' age is divided into five age groups: 'under 18', '18-24', '25-34', '35-44', '45-49', '50-55' and '56+'. Users' occupation is chosen from the 20 options, which is described in MovieLens website [30]. Users' age and occupation also be encoded as one-hot. Item side information consists of movie ID, director, main actors, genres and year of release, which are also encoded by one-hot. We set the output layer dimension k from 3 to 15. To make a fair comparison, we set the learning rate $\eta = 0.005$, and k from 3 to 15 for other algorithms. Since bias factors are added in our methods, the initial values of f_i and g_j from 0 to 1. All the experiments run randomly ten times on each dataset, and the average performance is reported.

B. RESULTS AND ANALYSIS

1) COMPARISON OF DIFFERENT ALGORITHMS

Table 2 to Table 4 show the performance of all algorithms on three datasets. Based on the results, we can make several observations. In Table 2 to Table 4, the bold values are the best performance in each column.

First, our methods (*DBPMF* and *DCBPMF*) outperform other baselines on the MovieLens100K and movieLens1M. On HetRec2011, *DCBPMF* performance is better than other

TABLE 3. The performance on MovieLens1M.

Method \ k	3	5	7	9	11	13
OLR	1.2342	1.1164	1.0384	0.9909	0.9770	0.9917
ConvMF	1.0225	1.0013	0.9808	0.9698	0.9899	1.0098
SOCF_II	1.0603	1.0058	0.9698	0.9509	0.9603	0.9775
DBPMF	0.9727	0.9523	0.9455	0.9488	0.9593	0.9756
DCBPMF	**0.9682**	**0.9495**	**0.9439**	**0.9470**	**0.9588**	**0.9712**

TABLE 4. The performance on HetRec2011.

Method \ k	3	5	7	9	11	13
OLR	0.9595	0.9076	0.8764	0.8790	0.8995	0.9214
ConvMF	0.9222	0.8901	0.8764	0.8790	0.8996	0.9225
SOCF_II	0.9060	0.8706	0.8612	**0.8598**	**0.8673**	**0.8789**
DBPMF	0.8822	0.8686	0.8616	0.8641	0.8788	0.8958
DCBPMF	**0.8701**	**0.8604**	**0.8599**	0.8688	0.8826	0.8889

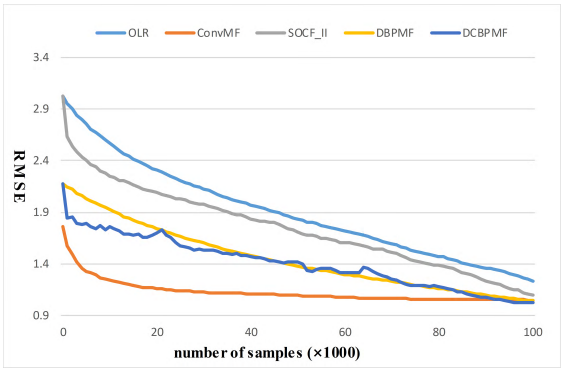


FIGURE 6. The performance of all methods on Movielens100K and k = 3.

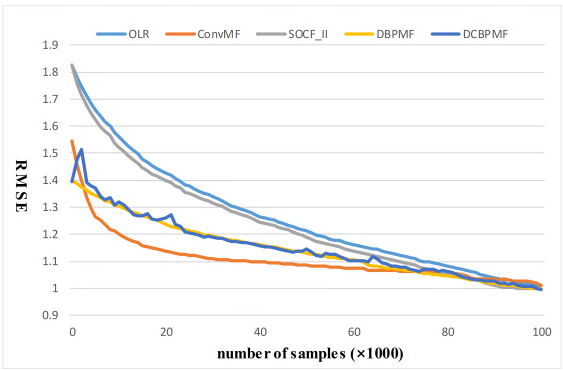


FIGURE 7. The performance of all methods on Movielens100K and k = 9.

methods when the $k < 9$. In most cases, we found that our method (*DBPMF*) outperformed the other online collaborative filtering methods, i.e. *OLR*, *ConvMF* and *SOCF_II*. This shows that adding *CNN* architecture and the bias into *PMF* is effective for the *OCF*.

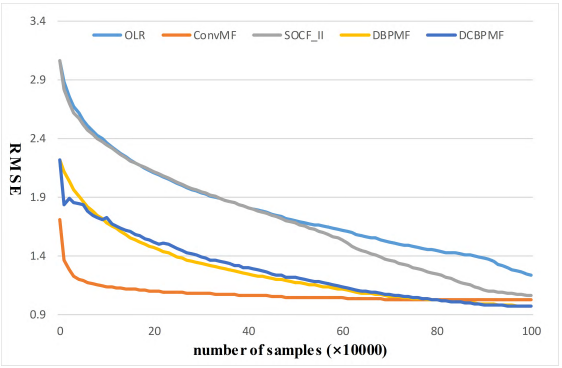


FIGURE 8. The performance of all methods on Movielens1M and k = 3.

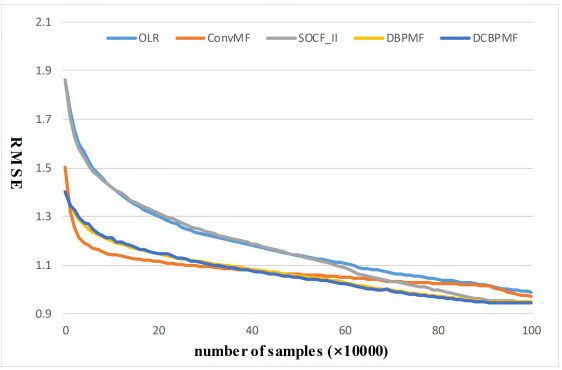


FIGURE 9. The performance of all methods on Movielens1M and k = 9.

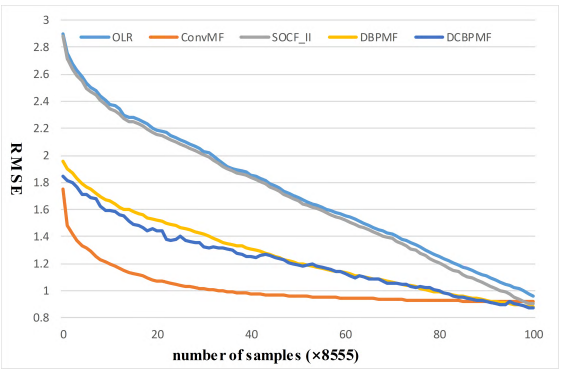


FIGURE 10. The performance of all methods on HetRec2011 and k = 3.

Second, the Figure 6 to 11 depict the changes of *RMSE* with the numbers of samples about all methods when the $k = 3$ and $k = 9$ on MovieLens100K, MovieLens1M and HetRec2011. We observed that the *DBPMF* and *DCBPMF* converged faster than *OLR* and *SOCF_II*, especially when the algorithm started running. Although our methods converged slowly than *ConvMF*, but the final *RMSE* of our methods is better than *ConvMF*. This is due to the *ConvMF* adopts batch learning approach to improve convergence rate. The Figure 6 to 11 show that our methods achieve best precise prediction in faster convergent rate.

Third, comparing the *DBPMF* and *DCBPMF*, the *DCBPMF* algorithm achieved lower *RMSE* values. This also

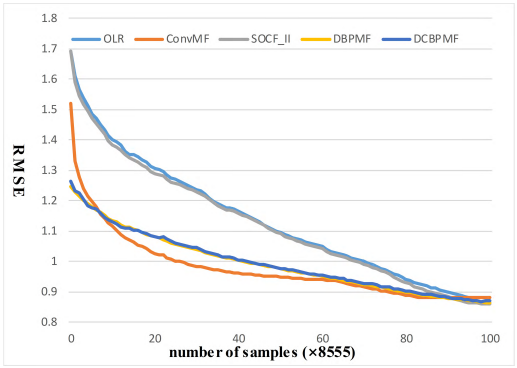


FIGURE 11. The performance of all methods on HetRec2011 and k = 9.

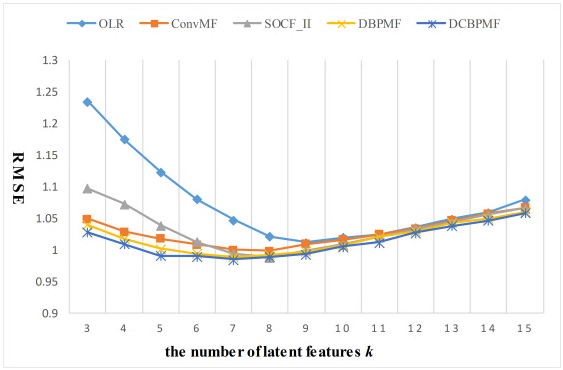


FIGURE 12. The performance of all methods on MovieLens100k with different latent factor k.

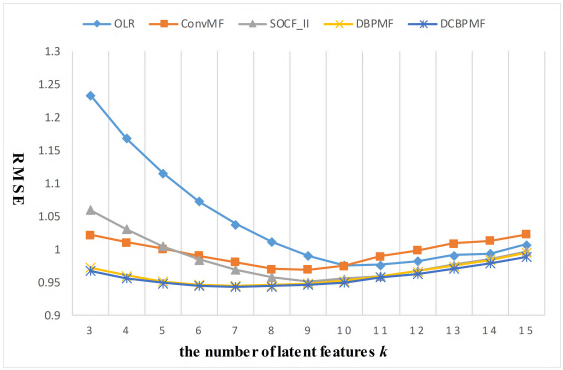


FIGURE 13. The performance of all methods on MovieLens1M with different latent factor k.

proves that constraining user-specific and item-specific feature vectors to *DBPMF* algorithm can improve the performance.

2) THE IMPACT OF THE NUMBER OF LATENT FEATURE

Figure 12 and Figure 13 show the performance of all methods on MovieLens100k and MovieLens1M with different latent factor k. We found that our algorithms (*DBPMF* and *DCBPMF*) were stable when the latent factor k varied from 3 to 15. *OLR* was the most sensitive to the changing of the number of latent features. More delightfully, our algorithms achieved the best performance when the latent factor k is very

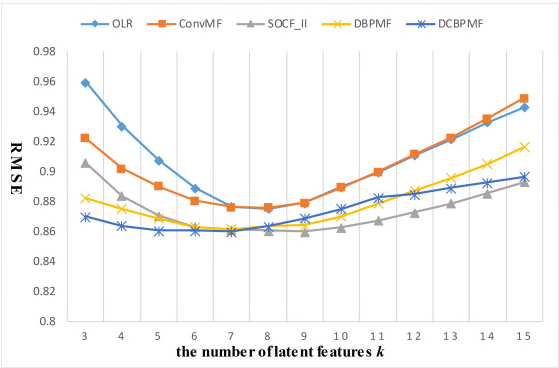


FIGURE 14. The performance of all methods on HetRec2011 with different latent factor k.

TABLE 5. The results of DBPMF with different $\lambda_u, \lambda_v, \lambda_f, \lambda_g$ on Movielens1M when k = 6.

λ_f, λ_g \ λ_u, λ_v	0.002	0.02	0.2	2
0.002	0.96712	0.95035	0.94745	0.96869
0.02	0.96868	0.95570	0.94805	0.96981
0.2	0.97543	0.96809	0.96327	0.97792
2	0.98497	0.98097	0.97464	0.99297

TABLE 6. The parameter $\lambda_Y, \lambda_X, \lambda_H, \lambda_Z, \lambda_f, \lambda_g$ of DCBPMF.

situations	λ_Y	λ_X	λ_H	λ_Z	λ_f	λ_g
S1	0.1	0.1	0.1	0.1	0.1	0.1
S2	0.01	0.01	0.1	0.1	0.1	0.1
S3	0.1	0.1	0.01	0.01	0.1	0.1
S4	0.1	0.1	0.1	0.1	0.01	0.01
S5	0.01	0.01	0.1	0.1	0.01	0.01
S6	0.1	0.1	0.01	0.01	0.01	0.01
S7	0.01	0.01	0.01	0.01	0.1	0.1
S8	0.01	0.01	0.01	0.01	0.01	0.01

small. It proves that our algorithms can track the user's rating behavior and the item's popularity when the value of k is very small (even less than 9). Figure 14 presents the impact of the number of latent features on HetRec2011. When the number of latent features k less than 8, our methods have comparative advantage. But *SOCF_II* perform well as the value of k increases. it is probably that the *SOCF_II* can capture more features on the HetRec2011 by calculating the second order. Also, maybe the dataset is small and the distribution of the item's popularity is more divergent. Overall, our methods are stable and k has little effect on our algorithms.

3) THE IMPACT OF PARAMETERS

Table 5 presents results of *DBPMF* on different parameters when $k = 6$. Because λ_u, λ_v impose the same effect to the

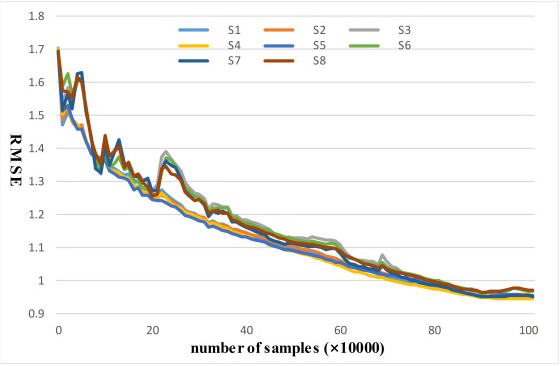


FIGURE 15. The convergence effects of DCBPMF in different situations on Movielens1M when k = 6.

algorithm *DBPMF*. So we adjust the parameters in pairs. So are λ_f, λ_g.We observed that when a parameter is determined, the value of *RMSE* decreases first and then increases as another parameter increases.

Table 6 shows the parameters $\lambda_Y, \lambda_X, \lambda_H, \lambda_Z, \lambda_f, \lambda_g$ setting of *DCBPMF* when we set $\lambda_u = 0.2, \lambda_v = 0.2$ as fixed value, and Figure 15 presents the corresponding the value *RMSE* with the number of samples. We observed that the curve of *DCBPMF* declines more smoothly when the $\lambda_Y, \lambda_X, \lambda_f, \lambda_g$ are same. And when λ_H, λ_Z are equal to 0.1, the *DCBPMF* obtains less value of *RMSE* with an increasing number of samples. Therefore, parameters are essential to constrain the stability and convergence rate of the algorithms and the parameters.

V. CONCLUSIONS

In this paper, we proposed an *OCF* method called *DBPMF* which combines *CNN* and the bias factors in *PMF* to improve the accuracy of *OCF*. Furthermore, we develop the *DBPMF* by constraining user-specific and item-specific feature vectors. We conduct extensive experiments to evaluate our methods' performance. In term of *RMSE*, experimental results show that the methods we proposed in this paper are superior to *OLR*, *ConvMF* and *SOCF_II*. Compared with the baselines, our methods converge quickly. Also, our methods have lower *RMSE* values and have high prediction accuracy. Therefore, *CNN* and bias factors are useful to improve the prediction accuracy of *OCF*. In future, we will try to accelerate the deep model update rate. In addition, we want to use commentary information and movie descriptions to extract user and item latent features.

REFERENCES

[1] R. Mu, "A survey of recommender systems based on deep learning," *IEEE Access*, vol. 6, pp. 69009–69022, 2018.

[2] A. van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Lake Tahoe, NV, USA, vol. 2, 2013, pp. 2643–2651.

[3] Y. Jin, N. Tintarev, and K. Verbert, "Effects of personal characteristics on music recommender systems with different levels of controllability," in *Proc. 12th ACM Conf. Recommender Syst. (RecSys)*, Vancouver, BC, Canada, 2018, pp. 13–21.

[4] B. Yang, Y. Lei, J. Liu, and W. Li, "Social collaborative filtering by trust," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 8, pp. 1633–1647, Aug. 2016.

[5] R. Chen, Q. Hua, Y. Chang, B. Wang, L. Zhang, and X. Kong, "A survey of collaborative filtering-based recommender systems: From traditional methods to hybrid methods based on social networks," *IEEE Access*, vol. 6, pp. 64301–64320, 2018.

[6] S. Zhang, L. Yao, and X. Xu, "AutoSVD++: An efficient hybrid collaborative filtering model via contractive auto-encoders," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Tokyo, Japan, vol. 2017, pp. 957–960.

[7] F. Ortega, D. Rojo, P. Valdiviezo-Díaz, and L. Raya, "Hybrid collaborative filtering based on users rating behavior," *IEEE Access.*, vol. 6, pp. 69582–69591, 2018.

[8] Y. Li, Z. Li, F. Wang, and L. Kuang, "Accelerated online learning for collaborative filtering and recommender systems," in *Proc. IEEE Int. Conf. Data Mining Workshop*, Shenzhen, China, Dec. 2014, pp. 879–885.

[9] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst*, 2008, pp. 1257–1264.

[10] C. Musto, G. Semeraro, M. de Gemmis, and P. Lops, "Learning word embeddings from wikipedia for content-based recommender systems," in *Advances in Information Retrieval*. Cham, Switzerland: Springer, 2016, pp. 729–734.

[11] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proc. 10th ACM Conf. Recommender Syst. (RecSys)*, New York, NY, USA, 2016, pp. 233–240.

[12] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 2015, pp. 1235–1244.

[13] 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 Int. Conf. World Wide Web*, Geneva, Switzerland, 2015, pp. 278–288.

[14] J. Liu, D. Wang, and Y. Ding, "PHD: A probabilistic model of hybrid deep collaborative filtering for recommender systems," in *Proc. 9th Asian Conf. Mach. Learn.*, vol. 77, 2017, pp. 224–239.

[15] S. C. H. Hoi, D. Sahoo, J. Lu, and P. Zhao. (Feb. 2018). "Online learning: A comprehensive survey." [Online]. Available: http://arxiv.org/abs/1802.02871

[16] G. Li, S. C. H. Hoi, K. Chang, W. Liu, and R. Jain, "Collaborative online multitask learning," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1866–1876, Aug. 2014.

[17] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2012.

[18] J. Wang, S. C. H. Hoi, P. Zhao, and Z.-Y. Liu, "Online multi-task collaborative filtering for on-the-fly recommender systems," in *Proc. 7th ACM Conf. Recommender Syst.*, New York, NY, USA, 2013, pp. 237–244.

[19] J. Lu, S. Hoi, J. Wang, and P. Zhao, "Second order online collaborative filtering," in *Proc. Asian Conf. Mach. Learn.*, Canberra, ACT, Australia, vol. 29, 2013, pp. 325–340.

[20] X. Zhou, W. Shu, F. Lin, and B. Wang, "Confidence-weighted bias model for online collaborative filtering," *Appl. Soft Comput.*, vol. 70, pp. 1042–1053, Sep. 2018.

[21] B. Pal and M. Jenamani, "Kernelized probabilistic matrix factorization for collaborative filtering: exploiting projected user and item graph," in *Proc. 12th ACM Conf. Recommender Syst. (RecSys)*, Vancouver, BC, Canada, 2018, pp. 437–440.

[22] Y. Gong and Q. Zhang, "Hashtag recommendation using attention-based convolutional neural network," in *Proc. 25th Int. Joint Conf. Artif. Intell. (IJCAI)*, New York, NY, USA, 2016, pp. 2782–2788.

[23] S. Kant and T. Mahara, "Merging user and item based collaborative filtering to alleviate data sparsity," *Int. J. Syst. Assur. Eng. Manag.*, vol. 9, no. 1, pp. 173–179, Feb. 2018.

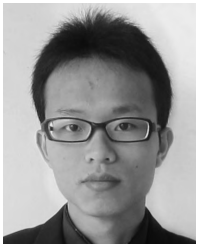
[24] S. Hao, J. Lu, P. Zhao, C. Zhang, S. C. H. Hoi, and C. Miao, "Second-order online active learning and its applications," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 7, pp. 1338–1351, Jul. 2018.

[25] C. Liu, T. Jin, S. C. H. Hoi, P. Zhao, and J. Sun, "Collaborative topic regression for online recommender systems: An online and bayesian approach," *Mach. Learn.*, vol. 106, pp. 651–670, May 2017.

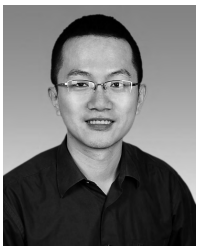
[26] G. Li, S. C. H. Hoi, K. Chang, and R. Jain, "Micro-blogging sen-timent detection by collaborative online learning," in *Proc. IEEE 10th ICDM*, Sydney, NSW, Australia, Dec. 2010, pp. 893–898.

[27] G. Li, K. Chang, S. C. H. Hoi, W. Liu, and R. Jain, "Collaborative online learning of user generated content," in *Proc. 20th ACM Int. CIKM*, 2011, pp. 285–290.
[28] J. Wang, P. Zhao, and S. C. H. Hoi, "Exact soft confidence-weighted learning," in *Proc. 29th Int. Conf. Int. Conf. Mach. Learn. (ICML)*, Edinburgh, Scotland, 2012, pp. 107–114.
[29] F. Lin, X. Zhou, and W. Zeng, "Sparse online learning for collaborative filtering," *Int. J. Comput. Commun. CONTROL*, vol. 11, no. 2, pp. 248–258, Apr. 2016.
[30] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, p. 19, Jan. 2016.
[31] J. Wang, C. H. Steven Hoi, P. Zhao, and Z.-Y. Liu, "Online multi-task collaborative filtering for on-the-fly recommender systems," in *Proc. 7th ACM Conf. Recommender Syst. (RecSys)*, New York, NY, USA, 2013, pp. 237–244.
[32] C. Liu, T. Jin, S. C. Hoi, P. Zhao, and J. Sun, "Collaborative topic regression for online recommender systems: An online and bayesian approach," *Mach. Learn*, vol. 106, no. 5, pp. 651–670, 2017.

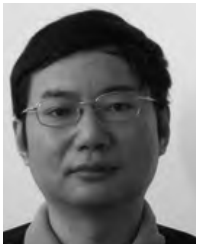
KANGKANG LI received the B.S. degree in education technology from Shanxi Datong University, in 2013, and the M.S. degree from Jiangsu Normal University, in 2016. He is currently pursuing the Ph.D. degree with the School of Software, Xiamen University, Xiamen, China. His current research interests include online collaborative filter, recommender systems, and deep learning techniques.



XIUZE ZHOU received the bachelor's degree from the Zhejiang University of Science and Technology, Hangzhou, China, in 2012, and the master's degree from the Department of Automation, Xiamen University, Xiamen, China, in 2016. His current research interests include online collaborative filter, machine learning, and recommender systems.



FAN LIN received the M.S. and Ph.D. degrees from Xiamen University, in 2003 and 2013, respectively, where he is currently an Associate Professor with the Software Engineering Department. His major research interests include online collaborative filter, machine learning, and recommender systems.



WENHUA ZENG received the M.S. and Ph.D. degrees in automation profession from Zhejiang University, in 1986 and 1989, respectively. He is currently a Professor with the Software Engineering Department, Xiamen University. His current research interests include machine learning, data mining, and evolutionary algorithms.

GIL ALTEROVITZ received the B.S. degree from Carnegie Mellon University and the S.M. degree in electrical engineering and computer science and the Ph.D. degree in electrical and biomedical engineering from MIT. He is currently an Assistant Professor with the Harvard Medical School. His major research interests include Bayesian methods and network models.