IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 5, 2019, accepted July 5, 2019, date of publication July 18, 2019, date of current version October 3, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2929789

An Efficient Hybrid Recommendation Model With Deep Neural Networks

ZHENHUA HUANG[1], CHANG YU[2], JUAN NI[3], HAI LIU[1], CHUN ZENG[1], AND YONG TANG[1]
[1]School of Computer Science, South China Normal University, Guangzhou 510631, China
[2]Department of Computer Science, Tongji University, Shanghai 201804, China
[3]School of Politics and Administration, South China Normal University, Guangzhou 510631, China

Corresponding author: Juan Ni (nijuan@m.scnu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772366, in part by the National Natural Science Foundation of China (Key Program) under Grant U1811263, and in part by the Natural Science Foundation of Shanghai under Grant 17ZR1445900.

ABSTRACT Recently, deep learning has gained great popularity in the area of recommender systems. Various combinations of deep learning, collaborative recommendation and content-based recommendation have occurred. However, as one of the three most significant recommendation techniques, hybrid recommendation has little cooperation with deep learning. Besides, most current deep hybrid models only incorporate two simple recommendation methods together in post-fusion, leaving massive space for further exploration of better combinations. In this paper, we apply deep learning to hybrid recommendation, proposing a deep hybrid recommendation model DMFL (Deep Metric Factorization Learning). In DMFL, we combine deep learning with improved machine learning models to learn the interaction between users and items from multiple perspectives. Such deep hybrid learning helps to reflect the user preference more comprehensively and strengthen model's ability of generalization. We also propose a more accurate method of user feature representation, taking both long-term static characteristics and short-term dynamic interest changes of users into consideration. Furthermore, thorough experiments have been conducted on real-world datasets, which strongly proves the effectiveness and efficiency of the proposed model.

INDEX TERMS Recommender system, deep learning, hybrid recommendation, comparative learning, performance evaluation.

I. INTRODUCTION

In a world with increasing scale of information, recommender systems have become an indispensable part in people's everyday life. Great convenience has been provided by recommender systems as they can help individuals seek their interested items among the surrounding overwhelming information fast and accurately. Recommender systems have now gained great popularity in many areas such as electronic commerce like Amazon, Alibaba; social network like Facebook, Twitter; and news like Google News, Toutiao.

Hybrid recommendation [1]–[3] is one of the most widely used conventional recommendation methods. As every single recommendation algorithm has its own shortcomings, hybrid recommendation combines different recommendation algorithms together to achieve a better recommendation quality, absorbing the advantages of both and inheriting the disadvantages of neither. There are three basic strategies in hybrid recommendation: post-fusion [4], middle-fusion [5] and former-fusion [6]. Post-fusion provides the final recommendation results by combining the results produced by two or more recommendation algorithms through voting mechanism, linear combination, or confidence level. Middle-fusion employs one recommendation algorithm as framework and incorporates another algorithm in it. Former-fusion mixes different kinds of recommendation algorithms into a unified recommendation model and then trains it with different kinds of samples.

Traditional hybrid recommender systems require heavy manual work for feature engineering and can hardly extract deep-dimensional features for users and items, which greatly limits the effectiveness and extensibility of hybrid recommendation. However, with the popularization of deep learning techniques [7]–[10], the new breakthrough of hybrid recommendation has occurred. Due to its nonlinear

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

architecture, the deep neural network can automatically extract the latent features for users and items from overwhelming data, free of heavy manual work. Growing number of studies [11]–[14] have adopted deep learning to hybrid recommender systems and the results proved that deep learning techniques can help improve the recommendation quality significantly. In these studies, deep learning is mainly applied to automatically learning deep representations from users' and items' information, while its ability to explore the deep connection between users and items is ignored. Meanwhile, the design of user representations in current studies is not accurate enough. Most studies only consider the short-term characteristics of users, ignoring the long-term preference characteristics of users contained in the user information.

In this paper, we propose a novel deep learning hybrid recommendation model DMFL (Deep Metric Factorization Learning). DMFL optimizes three mainstream recommendation frameworks - deep learning, factorization machine and metric learning – and combines them to achieve final recommendation results from multiple perspectives. We employ deep learning to extract the latent features of users and items, and explore the deep nonlinear relationship between them. We incorporate FM (Factorization Machine) [15] with SDAE (Stacked Denoising Autoencoder) [7] to learn the feature inter-actions between users and items, catching the linear relationship between them. We implement metric learning with pseudo-Siamese networks [16] to measure the relationship between users and items in the perspective of distance and catch user preference more effectively. DMFL generates the final recommendation results from three different perspectives, which helps to enhance the recommendation performance greatly.

In DMFL, we also provide a more comprehensive method of user feature representation. We divide the user feature vector into two parts: 1) the vector of static user basic information feature which represents users' long-term characteristics, and 2) the vector of dynamic user history preference feature which represents users' dynamic changes in interests. The combination of static and dynamic features could express user features more accurately.

The main contributions of our work are as follows:

- We proposed a novel deep hybrid recommendation model DMFL, combining deep learning with improved machine learning models to learn the interaction between users and items from multiple perspectives, which compensates the shortcomings of individual methods and improves the overall recommendation quality effectively.

- We designed a more accurate method of user feature representation, taking both long-term static characteristics and short-term dynamic interest changes of users into consideration.

- We implemented and evaluated DMFL on real-world datasets. The experimental results show the effectiveness and efficiency of DMFL in comparison with state-of-the-art baselines.

The rest of this paper is organized as follows: Section 2 summarizes the related work in this area. Section 3 describes the proposed model DMFL in detail. Section 4 presents the experiments and analysis. Section 5 concludes the paper.

II. RELATED WORKS

In this section, we summarize the related work from two aspects. We first introduce the relevant work about conventional hybrid recommendation, and then introduce the works that employ deep learning to recommender systems.

A. CONVENTIONAL HYBRID RECOMMENDATION

Hybrid recommendation was first mentioned in 1997 [1], proposing the idea of combining content-based recommendation [17] and collaborative recommendation [18] together to build the innovative recommendation models. Hybrid recommendation then caught researchers' attention in the following years and became one of the three most popular recommendation methods. Various studies [19]–[21] about hybrid recommendation models have occurred [22] proposes a unified probabilistic framework that merges collaborative and content-based recommendations. A three-way co-occurrence data among users, items, and item content are constructed, achieving collaboration data and content data. Global probabilistic models coupled with standard EM (Expectation Maximization) learning algorithms are used to train the model. [23]enriches the item-based collaborative filtering models by incorporating additional item meta-data to generate a HIN (Heterogeneous Information Network), which helps ease data sparsity issue greatly. Reference [6] takes the relationship between content features into account, extracting the content feature relationship matrix through history data and then integrating it to the final model using collaborative filtering.

Though conventional hybrid recommender systems can achieve great recommendation performance, it is always confronted with a challenge - feature learning. Hybrid recommendation requires heavy artificial feature engineering and can only catch the superficial relationship between users and items. Hence, deep learning can be applied to recommender systems to learn features automatically and capture deep interactions between users and items.

B. DEEP LEARNING RECOMMENDATION

The first attempt to combine deep learning with recommender systems was made by [24] using RBM (Restricted Boltzmann Machine) in 2007. It trained a single RBM for each user and improved the recommendation performance. With the popularization of deep learning techniques, many more deep learning models have been applied to the recommender systems, including autoencoder, MLP (Multi-Layer Perceptron), CNN (Convolutional Neural Network), RNN (Recurrent Neural Network) and etc.

In the research of recommendation systems based on autoencoder [25]–[27], Wang *et al.* [28] proposed the CDL (Collaborative Deep Learning) model, using bag of words

to express items' text information and learning items' latent feature through Bayesian SDAE (Stacked Denoising Autoencoders). The learned feature vectors were then fed into the PMF (Probabilistic Matrix Factorization) model, and the inner products of the latent vectors were calculated as user preference scores for items. Jian *et al.* [29] combined the SDAE model with the Time SVD++ [30] algorithm, employing SDAE model to learn latent feature vectors of users and items, applying Time SVD++ algorithm to considering the effect of time in the process of predicting user preferences.

In the research of recommendation systems based on MLPs [11], [31], [32], Covington *et al.* [33] proposed a video recommendation model for YouTube website, which turned the recommendation problem into a multi-classification problem based on MLP. The MLP was used to extract the embedding vectors for users and videos, and the similarities between users and videos were obtained by softmax multi-classifier.

Google proposed the famous Wide & Deep model [34] in 2016, which was applied to mobile app recommendations in Google play. Wide & Deep model consists of a wide linear model and a deep neural network model. It uses its wide component to exploit the features in history data, and deep component to generalize new feature combinations, combining the benefits of memorization and generalization at the same time.

Based on Wide & Deep model, Guo *et al.* [35] proposed the DeepFM model, replacing the wide linear model in the Wide & Deep model with FM, performing feature combination directly through FM, and skipping the complex process of manual feature combination in the Wide & Deep model. He *et al.* [12] further proposed the NFM (Neural Factorization Machine) model, combining FM with MLP in a pipeline way. NFM first performed order-2 feature pooling combination by FM, and then fed the combined feature vectors into MLPs for predictive output.

In the study of recommendation systems based on CNNs [36]–[38], Dieleman and Schrauwen [39] used CNNs to learn the latent representations of songs and calculated the similarity between user latent vectors and song latent vectors to ease the cold start problem of new songs in the recommender systems. Kim *et al.* [40] proposed ConvMF (Convolutional Matrix Factorization) model, using texts as auxiliary information. ConvMF adopts CNNs to extract the feature vectors of items from their text description, effectively solving the problem that bag of model cannot effectively exploit internal communication of words. It then uses the matrix factorization algorithm to calculate the scores that users give to items, achieving good performance on sparse data. Wang *et al.* [14] used CNNs to extract latent features from images and incorporate the image latent features into the recommendations of the Place of Interest (POI)to improve the recommendation quality.

In the research of recommendation systems based on RNNs [41]–[43], Hidasi *et al.* [44] exploited the serialization feature of GRU (Gated Recurrent Unit) to session recommendation, predicting the next item the user might choose. Okura *et al.* [45] used RNNs to learn users' historical behavior feature vector, used the DAE (denoising autoencoder) to learn the feature representation vector of news from news content, and then calculated the similarity between user vectors and news vectors to obtain user preferences for news. Li *et al.* [46] divided user features into user history preference features and user current session preference features during session recommendation, learning historical preference features with bidirectional RNN models, and learning current session preference features with LSTM (Long Short-Term Memory) model.

In the previous works, though there are various combinations of deep learning techniques and machine learning techniques, we are the first to propose the combination of deep neural networks, factorization machine and metric learning in recommendation models.

III. DEEP METRIC FACTORIZATION LEARNING

This section gives a detailed description of the proposed model DMFL. We first make a brief introduction to the overall structure of DMFL, and then describe the components of the model in detail. Finally, the process of model training is explained.

A. THE ARCHITECTURE OF DMFL

As is shown in Fig. 1, the proposed model DMFL can be divided into two parts: feature learning and preference generation.

The feature learning part is composed of two paralleled deep neural net-works to extract static item latent feature vectors and dynamic user latent feature vectors respectively. The preference generation part consists of three sub-modules, which are the SDAE-FM module, the deep neural network module and the metric learning module. The learned latent feature vectors for users and items in the feature learning part are fed into the three modules as inputs simultaneously, and the results generated by the three sub-modules are combined as the final user preference prediction result for items.

B. FEATURE LEARNING

We assume that there are m users and n items in the target recommendation scenario of this paper. We record the user set as $U = \{u_1, u_2, \ldots, u_m\}$, and user i is represented as u_i, i$=$ $1, 2, \ldots, m$. X denotes the basic information set of users and the basic information set of u_i is denoted as X_i. The item set is recorded as $V = \{v_1, v_2, \ldots, v_n\}$, and item j is represented as v_j, $j = 1, 2, \ldots, n$. Y denotes the basic information set of items and the basic information set of v_j is denoted as Y_j. The user-item rating matrix is denoted as $\boldsymbol{R} = \{r_{ij}\} \in \mathbb{R}^{m \times n}$, where r_{ij} represents the rating that u_i gives to v_j.

The feature learning process can be divided into two parts: item feature learning and user feature learning. In order to fully exploit the deep hidden information contained in items and users, we employ deep learning techniques to extract deep latent feature vectors. Moreover, considering the different characteristics of items and users, two different methods,

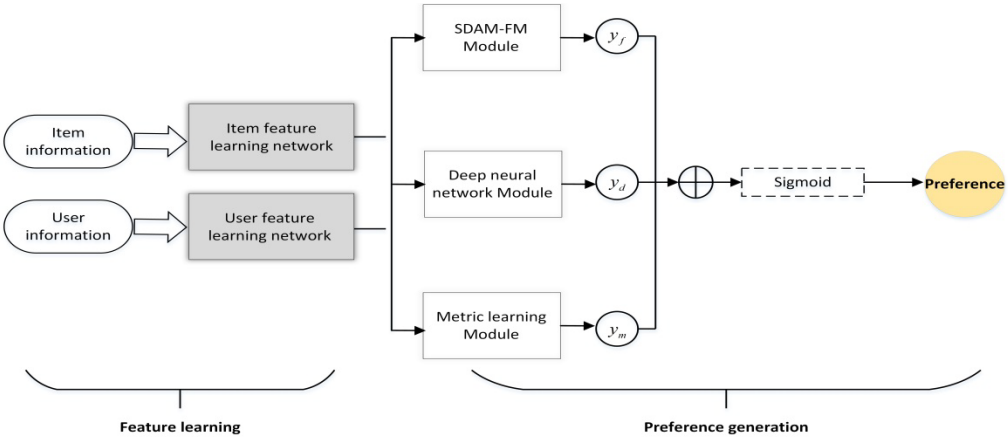


FIGURE 1. The architecture of DMFL.

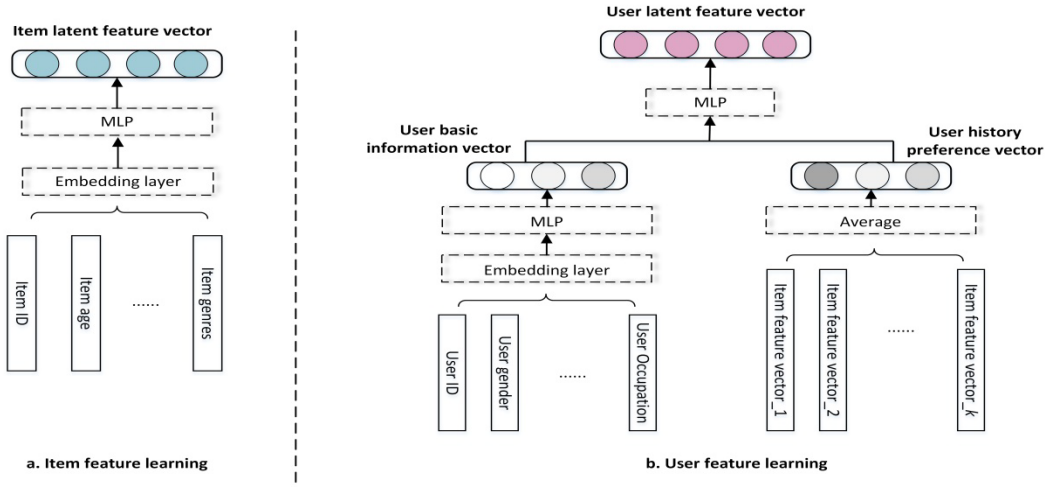


FIGURE 2. Process of feature learning.

static and dynamic, are designed to extract the features of items and users, respectively. The specific feature learning networks are shown in Fig. 2, and the part *a* indicates the item feature learning network, part *b* represents the user feature learning network.

We first introduce the static item feature learning process. Normally the nature of items to be recommended is stable and rarely changes, so we could directly extract the item latent feature vectors from static item basic information. The general item basic information Y_j of v_j includes the item ID, item name, item category and other information of description. As is shown in Fig.2.a, we feed the item basic information into the item feature learning network to obtain the deep latent feature vectors of items. The item feature learning process can be expressed as:

$$\boldsymbol{y}_j = net\_v\left(\vartheta_v, Y_j\right) \tag{1}$$

where $net\_v(\cdot)$ represents the computation process of item feature learning network, ϑ_v represents all the parameters in item feature learning network, $\boldsymbol{y}_j \in \mathbb{R}^d$ denotes the obtained feature vector of v_j and d represents its dimension.

We then introduce the dynamic user feature learning process. Unlike static item features, user features are dynamic in the actual recommendation scenario for users' interests could change frequently in very short intervals. Therefore, the user feature vectors should be updated and transformed in time as the user's interests change, to represent the characteristics of users more accurately. At the same time, due to the basic personal information such as individual gender, age, experience, etc., users also have some fixed long-term characteristics. For example, women are more interested in clothing, makeup, and men prefer sports, machinery.

Considering the above two aspects, we proposed a comprehensive user feature representation method to fully and accurately represent user characteristics. We divide the user feature vector into two parts: user basic information feature vector and user history preference feature vector. As is shown in Figure 2.b, we use deep neural network to extract u_i's user basic information feature vector $\boldsymbol{x}_i^c$ from its basic information

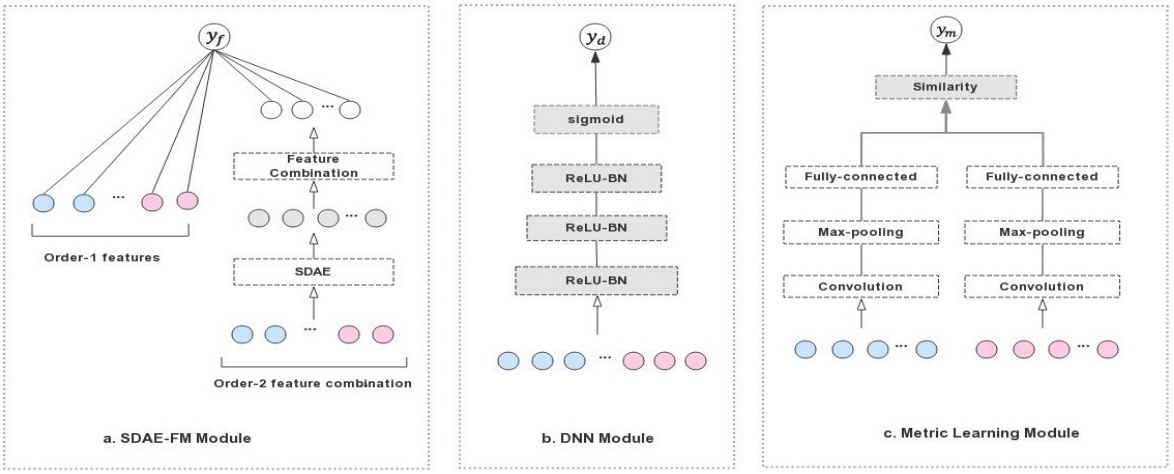


FIGURE 3. Structures of three sub-modules.

X_i, including the user ID, age, occupation, region, etc. In the meantime, we extract the item feature vectors of the k items that u_i likes most recently (or the k items with the highest user history ratings) from its history rating record, and denote it as u_i's history preference matrix $\boldsymbol{X}_i^h = \{\boldsymbol{y}_1^i, \boldsymbol{y}_2^i, \ldots, \boldsymbol{y}_k^i\} \in \mathbb{R}^{k \times d}$. The feature vectors of the k items are summed and averaged, achieving the user history preference feature vector $\boldsymbol{x}_i^h$:

$$\boldsymbol{x}_i^h = \frac{1}{k} \sum_{t=1}^{k} \boldsymbol{y}_t^i. \tag{2}$$

We then feed $\boldsymbol{x}_i^c$ and $\boldsymbol{x}_i^h$ into a layer of neural network to obtain the final user feature vector $\boldsymbol{x}_i \in \mathbb{R}^d$, d represents the dimension of $\boldsymbol{x}_i$:

$$\boldsymbol{x}_i = f\left(\boldsymbol{W}_\alpha \left[\boldsymbol{x}_i^c : \boldsymbol{x}_i^h\right] + \boldsymbol{b}_\alpha\right), \tag{3}$$

where $f(\cdot)$ is the activation function of the layer, $\mathbf{W}_\alpha$, $\mathbf{b}_\alpha$ are the weights and bias of the layer respectively.

We record the user feature learning process as function $net\_u(\cdot)$, and all the parameters in the user feature extraction network as ϑ_u.The user feature learning process can be expressed as:

$$\boldsymbol{x}_i = net\_u\left(\vartheta_u, \boldsymbol{X}_i, \boldsymbol{X}_i^h\right). \tag{4}$$

C. PREFERENCE GENERATION

After obtaining the user and item feature vectors, we feed the obtained feature vectors into preference generation part to achieve user preference prediction. Inspired by the Wide & Deep model, we found that combining different models simultaneously could analyze and catch the characteristics of data from multiple perspectives, making up for the shortcomings and deficiencies of individual models, absorbing the advantages of all the models, and thus obtaining more accurate results. Hence, in the preference generation part, we design three different sub-modules from three perspectives and combine the results of the three sub-modules as the final user preference prediction results. The detailed structure of the preference generation part is shown in Fig. 3. The three sub-modules are trained jointly under the unified objective function—predicting user preference, and such deep hybrid helps improve the generalization ability of the proposed model significantly.

1) SDAE-FM MODULE

The part *a* in Fig. 3 represents the SDAE-FM module. We improved the traditional FM model by introducing SDAE model to reduce feature dimension and extract deep latent features, comprehensively considering both linear features and deep nonlinear features. We utilize this module to learn the importance of each feature component and the relationship between feature components more accurately, obtaining user preferences from the perspective of feature combination.

The traditional FM model directly uses the original order-1 feature components to perform the order-2 feature combination. Although matrix factorization is used to reduce the computational complexity, the number of combinations between feature components is still large when the feature dimension is large. To alleviate this problem, we introduce SDAE model to optimize the process of order-2 feature combination. In the SDAE-FM module, the original features are first reduced by SDAE, then the reduced features are employed to perform feature combination, which reduces the complexity of feature combination effectively. Moreover, as the features reduced by SDAE are the deep high-dimensional representation of original features, the combination of deep order-2 nonlinear features and shallow order-1 linear features can represent the user preferences more comprehensively.

We construct a three-layer SDAE model and train the parameters of each layer of SDAE in order. Each SDAE layer performs the same computation and we take the first layer of SDAE as an example: we first splice $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ as $\boldsymbol{z}_{ij} = [\boldsymbol{x}_i : \boldsymbol{y}_j]$, the $\boldsymbol{z}_{ij}$ added with random noise is set as the input. And the

calculation process of first layer of SDAE is as follows:

$$z_{ij}^{'} = Noise\left(z_{ij}\right), \tag{5}$$

$$z_{ij}^{1} = \text{encode}\left(\vartheta_e^1, z_{ij}^{'}\right), \tag{6}$$

$$\widehat{z}_{ij}^{'} = \text{decode}\left(\vartheta_d^1, z_{ij}^{1}\right), \tag{7}$$

$$\mathcal{L}_{S_1}\left(\vartheta_e^1, \vartheta_d^1, z_{ij}^{'}\right) = \frac{1}{2mn}\sum\nolimits_{i,j}\left\|z_{ij}^{'} - \widehat{z}_{ij}^{'}\right\|^2, \tag{8}$$

where *Noise* $(\cdot)$ denotes the function of adding random noise to the feature vectors, *encode* $(\cdot)$ denotes the operation of the encoder part, *decode* $(\cdot)$ denotes the operation of the decoder part, $\vartheta_e^1, \vartheta_d^1$ denotes the parameters of the decoding and encoding part respectively, $\mathcal{L}_{S_1}\left(\vartheta_e^1, \vartheta_d^1, z_{ij}\right)$ represents the reconstruction loss function of the first layer, z_{ij}^{h1} is the hidden vector learned by the first SDAE layer, and we set it as the input of the next layer of SDAE.

We record the reconstruction loss function of the SDAE model as the sum of the reconstruction losses of the three layers:

$$\mathcal{L}_S(\vartheta_{f\_s}, \boldsymbol{x}_i, \boldsymbol{y}_j) = \mathcal{L}_{S_1}\left(\vartheta_e^1, \vartheta_d^1, z_{ij}^{'}\right) + \mathcal{L}_{S_2}\left(\vartheta_e^2, \vartheta_d^2, z_{ij}^{1'}\right) + \mathcal{L}_{S_3}\left(\vartheta_e^3, \vartheta_d^3, z_{ij}^{2'}\right), \tag{9}$$

For the sake of simplicity, we record all the parameters of the SDAE part as ϑ_{f_s}, $\vartheta_e^t, \vartheta_d^t$ represents the parameters of the t-th layer of SDAE, $z_{ij}^{1'}$ represents the input of the second layer of SDAE, and $z_{ij}^{2'}$ represents the input of the third layer of SDAE.

We denote the deep-dimensional vector obtained after three layers as z_{ij}^h. Then the final output of SDAE-FM module is:

$$y_{ij}^f = \omega_0 + \sum\nolimits_{t=1}^{2d}\omega_t z_{ij,t} + \sum\nolimits_{t=1}^{2d}\sum\nolimits_{k=t+1}^{2d} < \boldsymbol{V}_t, \boldsymbol{V}_k > z_{ij,t}^h z_{ij,k}^h, \tag{10}$$

where $\omega_0 \in \mathbb{R}$ is the bias, $\omega_i \in \mathbb{R}^n$ models the importance of order-1 features; $\boldsymbol{V}_t = (v_{t,1}, v_{t,2} \ldots, v_{t,p})^T \in \mathbb{R}^p$ is the latent support vector of $z_{ij,t}^h$. $< \boldsymbol{V}_t, \boldsymbol{V}_k >$ is the inner product of $\boldsymbol{V}_t$ and $\boldsymbol{V}_k$, $< \boldsymbol{V}_t, \boldsymbol{V}_k >= \sum_{q=1}^{p} v_{t,q} \cdot v_{k,q}$, and it denotes the impact of order-2 feature interactions. $2d$ is the dimension of z_{ij}^h.

We use ϑ_f to represent all the parameters in the SDAE-FM module, $h_f(\cdot)$ to represent the entire calculation process of SDAE-FM module, and then rewrite Eq. (10) as:

$$y_{ij}^f = h_f\left(\vartheta_f, \boldsymbol{x}_i, \boldsymbol{y}_j\right). \tag{11}$$

2) DEEP NEURAL NETWORK MODULE

The part *b* in Fig. 3 represents the deep neural network module. We use deep neural network to further explore the deep nonlinear relationship between users and items, and generate more accurate user preferences from the perspective of deep structure.

We designed a four-layer neural network, and the computation performed in each layer is shown in the formula:

$$\boldsymbol{a}^{(l+1)} = f\left(\boldsymbol{W}^l\boldsymbol{a}^l + \boldsymbol{b}^l\right), \tag{12}$$

where l represents the number of network layers, $f(\cdot)$ represents the activation function used by each layer of the network, where ReLUs(Rectified Linear Units) [47], [48] is used as the activation function here. $\boldsymbol{a}^l, \boldsymbol{W}^l, \boldsymbol{b}^l$ represent the input, weights and bias of the l-th layer respectively.

In order to prevent the over-fitting problem, we employ Batch Normalization [49] technology to standardize the input of each layer of network, so that the average value of each layer's input is 0 and the variance is 1.

$$\mu = \frac{1}{m}\sum\nolimits_{t=1}^{N} a_t^l, \tag{13}$$

$$\sigma^2 = \frac{1}{N}\sum\nolimits_{t=1}^{N}\left(a_t^l - \mu\right)^2, \tag{14}$$

$$\boldsymbol{a}_t^{l-norm} = \frac{a_t^l - \mu}{\sqrt{\sigma^2 + \varepsilon}}, \tag{15}$$

where N represents the number of input samples, a_t^l represents the t-th input of the first layer, and ε takes 10^{-8}.

The final output of deep neural network module is:

$$y_{ij}^d = sigmoid\left(\boldsymbol{W}^f\boldsymbol{a}^f + \boldsymbol{b}^f\right), \tag{16}$$

where *sigmoid* $(\cdot)$ is the sigmoid function, a^f, W^f, b^f are the input, model weights and bias of the final layer respectively.

We use ϑ_d to represent all the parameters in the neural network module, and $h_d(\cdot)$ to represent the calculation process of the neural network module. The entire preference generation process of the neural network module is:

$$y_{ij}^d = h_d\left(\vartheta_d, \boldsymbol{x}_i, \boldsymbol{y}_j\right). \tag{17}$$

3) METRIC LEARNING MODULE

The *c* part in Fig. 3 represents the metric learning module. In this module, we measure the relationship between users and items from the perspective of distance. Obviously, our goal is to ensure that the distance between users and the items they like is closer than that they dislike.

We first designed a pair of pseudo-Siamese networks to further mine the features of users and items, and then calculated the distance between items and users with the learned feature vectors. The pseudo-Siamese networks part are two parallel convolutional neural network models with different structures and parameters. The computation process of the two networks is similar, where we take the user network as an example.

We use $\boldsymbol{k}_t$ to denote the t-th filter, and the computation in convolutional layer can be denoted as:

$$\boldsymbol{h}_t = f(\boldsymbol{x}_i * \boldsymbol{k}_t + \boldsymbol{b}_t), \tag{18}$$

where "*" denotes the convolution operation, $\boldsymbol{b}_t$ is a bias, and $f(\cdot)$ is the activation function RELUs.

Then the output is connected to a global max-pooling layer to capture the feature with the highest value as the higher the

value is, the more important the feature is. The computation in this layer is:

$$\boldsymbol{m}_t = max\{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_{(d-c+1)}\}, \tag{19}$$

where d is the length of $\boldsymbol{x_i}$, and c is the length of feature map.

We use l to represent the number of kernels, and the final convolution vector is:

$$\boldsymbol{m} = \{\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_l\}. \tag{20}$$

We then feed m to a fully connected layer to achieve the final learned feature vector:

$$\boldsymbol{x}_i^{'} = f(\boldsymbol{W}_u\boldsymbol{m} + \boldsymbol{b}_u). \tag{21}$$

We could obtain $\boldsymbol{y}_j^{'}$ similarly. Considering the accuracy of metric computation, we normalize the learned vectors to make sure that $\| \boldsymbol{x}_i^m \|^2 \leq 1$, $\| \boldsymbol{y}_j^m \|^2 \leq 1$:

$$\boldsymbol{x}_i^m = norm(\boldsymbol{x}_i^{'}), \tag{22}$$

$$\boldsymbol{y}_j^m = norm(\boldsymbol{y}_j^{'}). \tag{23}$$

Then we use the product similarity to measure the distance between users and items as the output of the metric learning module:

$$y_{ij}^m = \boldsymbol{x}_i^m \cdot \boldsymbol{y}_j^m. \tag{24}$$

We represent all the calculations of the metric learning module with the function $h_m(\cdot)$. ϑ_{m_u}, ϑ_{m_v} respectively represent the parameters of the user and item networks, where $\vartheta_m = \{\vartheta_{m_u}, \vartheta_{m_v}\}$, and then the output of the metric learning module is:

$$y_{ij}^m = h_m\left(\vartheta_m, \boldsymbol{x}_i, \boldsymbol{y}_j\right). \tag{25}$$

The three sub-modules generate user preferences from the perspectives of linearity, deep nonlinearity and distance respectively. They mutually restrain and promote each other, which can effectively improve the accuracy of prediction. The final preference output of the model is:

$$\widehat{y}_{ij} = sigmoid(y_{ij}^f + y_{ij}^d + y_{ij}^m). \tag{26}$$

D. MODEL TRAINING

We train the three sub-modules jointly and the final loss function of the DMF model consists of the cross-entropy prediction loss and the SDAE reconstruction loss $\boldsymbol{\mathcal{L}_S}$.The final loss function of the DMF model is:

$$\begin{aligned}\mathcal{L}\left(\vartheta_v, \vartheta_u, \vartheta_f, \vartheta_d, \vartheta_m, X, Y, \boldsymbol{R}\right) \\ = min \sum\nolimits_{i,j} -\left(y_{ij}\, log\, \widehat{y}_{ij}\right) - \left(1 - y_{ij}\right) log\left(1 - \widehat{y}_{ij}\right) + \mathcal{L}_S \\ + \lambda_l(\|\vartheta_v\|^2 + \|\vartheta_u\|^2 + \left\|\vartheta_f\right\|^2 + \|\vartheta_d\|^2 + \|\vartheta_m\|^2)\end{aligned} \tag{27}$$

where $(\|\vartheta_v\|^2 + \|\vartheta_u\|^2 + \left\|\vartheta_f\right\|^2 + \|\vartheta_d\|^2 + \|\vartheta_m\|^2)$ is the l_2 regularization terms of all parameters in the model to prevent over-fitting. λ_l represents the coefficient of regulation term to control the influence of model regulation.

Algorithm 1 DFML

Input: User information set X, item information set Y, rating matrix $\boldsymbol{R}$, batch size N
Output: Model parameters $\{\vartheta_v, \vartheta_u, \vartheta_f, \vartheta_d, \vartheta_m\}$
Initialize $\vartheta_v, \vartheta_u, \vartheta_f, \vartheta_d, \vartheta_m$
repeat
  Randomly choose N samples from the training set;
  for sample from 1 to N **do**
    calculate $\boldsymbol{y}_j, \boldsymbol{x}_i$ according to Eq. (1), Eq. (4)
    calculate $y_{ij}^f, y_{ij}^d, y_{ij}^m$ according to Eq. (11), Eq. (17), Eq. (25)
    calculate $\widehat{y}_{ij}$ according to Eq. (26)
    calculate $\mathcal{L}$ according to Eq. (27)
  calculate the average gradients of N samples of all parameters
  update the parameters according to Eq. (28)
  endfor
until the model is convergent
return $\{\vartheta_v, \vartheta_u, \vartheta_f, \vartheta_d, \vartheta_m\}$

We employ Mini-Batch Stochastic Gradient Descent algorithm to optimize the objective function, and train N ($2 \leq N \leq 100$) samples at one time to improve training efficiency. We choose AdaGrad [50] algorithm to control the value of learning rating automatically. The process of AdaGrad algorithm is as Eq.(28):

$$\theta_{t+1} \leftarrow \theta_t - \left(\frac{\alpha}{\sqrt{\sum_{i=1}^{t}(\frac{\partial \mathcal{L}}{\partial \vartheta_i})^2 + \epsilon}}\right)\frac{\partial \mathcal{L}}{\partial \vartheta_t}, \tag{28}$$

where $\frac{\partial \mathcal{L}}{\partial \vartheta_t}$ is the gradient of ϑ_t, α is the original learning rate, and ϵ is 10^{-8}.

The detailed process of model training is as follows:

E. COMPLEXITY ANALYSIS

We assume that there are m users, n items, every user has rated $\bar{n}$ items ($\bar{n} \ll n$), k items are selected as the user history preference items, the dimension of user information is d_u, the dimension of user information is d_v, and the dimension of extracted latent feature vectors is d.

The main time complexity of DFML model is derived from the $m\bar{n}$ times calculation of user preference, including feature learning and preference generation. The overall time complexity is $O(m\bar{n})$. The space complexity of the model is related to the batch size N during training, the user history number k, the information dimensions d_u, d_v, and d, since N, k, d_u, d_v, and d are fixed constants, and N is much smaller than the total number of samples. Therefore, the space complexity of the model is $O(1)$.

IV. EXPERIMENTS

This section conducts comprehensive evaluation of the proposed model on three real-world datasets.

TABLE 1. The statistics of three real-world datasets.

	Users	Items	Ratings	Ratings per user	Ratings per item
Movielens-20M	138,493	27,278	20,000,263	144.4	733.2
Movielens-1M	6,040	3,900	1,000,209	165.6	256.5
BookCrossing	278,858	271,379	1,149,780	4.1	4.2

A. DATASETS AND EVALUATION METRICS

We conduct our experiments on three real-world datasets, Movielens-20M,[1] Movielens-1M[2] and BookCrossing.[3] Table 1 shows the statistics of the three datasets.

• MovieLens-20M: MovieLens-20M dataset is one of the most widely used datasets in the recommendation area. It is a movie dataset collected from MovieLens website, containing the information about users, movies and ratings that users give to movies. In our experiments, according to previous works, we regard the ratings greater than or equal to 4 as positive feedback, ratings less than 4 as negative feedback, and only those users with more than 20 ratings are included in the experiments.

• MovieLens-1M: MovieLens-1M dataset is another version of Movielens dataset, containing the information about users, movies and ratings that users give to movies. In our experiments, we also regard the ratings greater than or equal to 4 as positive feedback, ratings less than 4 as negative feedback, and only include the users with more than 20 ratings in the experiments.

• BookCrossing: BookCrossing dataset is a classic book dataset crawled from the Book-Crossing community from August to September in 2004, containing the information about users, books and ratings. As it is much sparser compared with the MovieLens dataset, we adopt it to evaluate our model's ability to ease the data sparsity problem. Also, according to previous works, we regard the ratings greater than or equal to 5 as positive feedback, ratings less than 5 as negative feedback.

We evaluate our model on two well-known metrics: Recall [51] and AUC [52] which are widely used for Top-k recommendation evaluation. AUC represents the area under ROC (Receiver Operating Characteristic) curve. Recall represents the percentage of correctly predicted true positive items in the samples. The formulation is as follows:

$$Recall = TP/(TP + FN) \tag{29}$$

where TP denotes the number of positive items that are correctly predicted to be true, and FN denotes the number of positive items that are falsely predicted to be false.

B. STATE-OF-ART BASELINES

We compare the proposed model with six state-of-the-art baselines to evaluate their performances.

[1]https://grouplens.org/datasets/movielens/20m
[2]https://grouplens.org/datasets/movielens/1m
[3]https://grouplens.org/datasets/book-crossing

• FM [15]: Factorization Machine, a generalized MF (Matrix Factorization) model that effectively captures feature interactions as inner product of respective feature latent vectors.

• Wide & Deep [34]: Wide& deep model combines deep neural network with wide linear regression and achieves the benefits of both memorization and generalization.

• CML [53]: Collaborative Metric Learning combines collaborative recommendation with metric learning. It extracts the latent features for items with MLP and then employs LMNN (Large Margin Nearest Neighbor) method to learn user's preference for items.

• DeepRec [54]: DeepRec adopts feedforward deep neural network learning to generate recommendation item vectors, and proposes weighted loss function to balance the popularity and novelty of recommended items.

• CDL[1] [47]:Comparative Deep Learning learns latent feature vectors for users and items by deep learning and then calculates the relative distance among users and a pair of items to make recommendation decisions.

• CDL[2] [28]: Collaborative Deep Learning extracts items' latent feature vectors from text data using Bayesian SDAE and generates predictions through PMF model.

C. EXPERIMENTAL SETTINGS

In our experiments, the datasets are randomly divided into training (70%), validation (10%), and test (20%) sets, the training set for model training, the validation set for hyper-parameters tuning and the test set for performance evaluation.

We perform grid search for all the compared model to find parameters which can achieve best recommendation performance in the validation set. A two-layer MLP with a 256-dimensional hidden layer, 50% dropout, ReLUs as the activation function is used as the feature extractor for CDL[1], CDL[2], DeepRec, and CML.

In DMFL, both the item feature extraction network and the user's feature extraction network are three-layer neural networks, including one layer of embedding layer and two layers of fully connected layer. We set the feature vector dimension $d = 150$ for both items and users. In the prediction generation part, the SDAE-FM module uses a three-layer SDAE model for feature dimension reduction, with the number of neurons per layer being 300, 128, and 64. The deep neural network model uses a four-layer neural network with 300,128, 32, 1 neurons per layer. The former three layers adopt ReLUs function as activation function and batch normalization to prevent overfitting, and the last layer uses sigmoid function as activation function. The metric learning module uses a pair

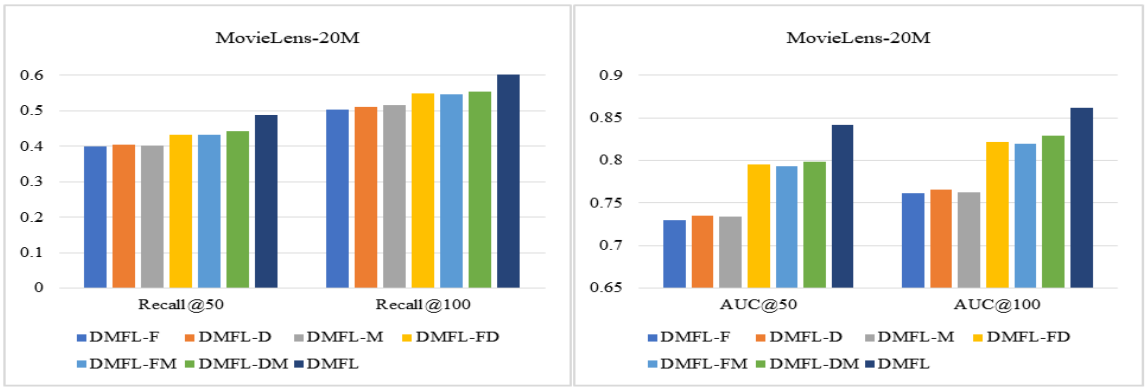


FIGURE 4. Recall and AUC on MovieLens-20M.

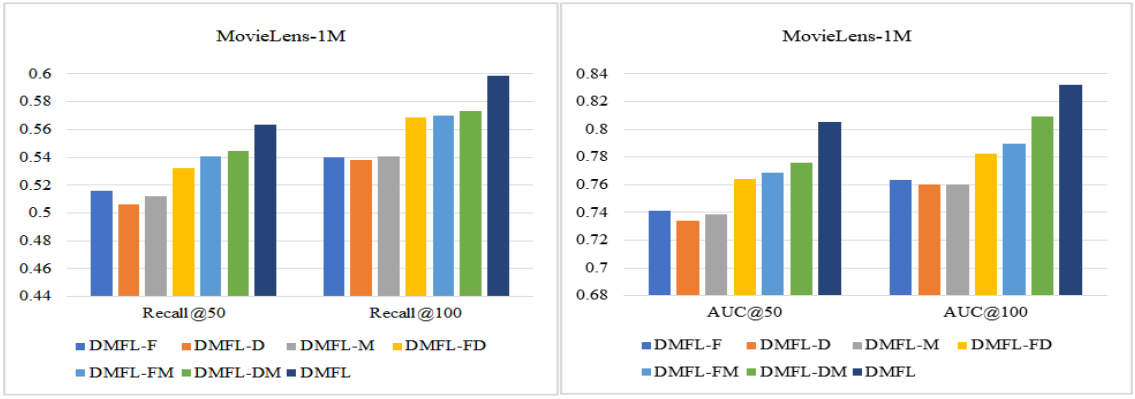


FIGURE 5. Recall and AUC on MovieLens-1M.

of pseudo-Siamese networks. The network for user is a CNN with 128 kernels whose size is 1×3, and the network for item is a CNN with 128 kernels whose size is 1×5. The regularization coefficient $\lambda_l = 0.01$, the batch size $N = 32$, and the initial learning rate $\alpha = 0.01$.

We implement our model on Tensorflow. All the models are trained with GPU acceleration.

D. PERFORMANCE EVALUATION

We conduct three groups of experiments to thoroughly evaluate the performance of our model.

1) EFFECTTIVENESS OF MODEL STRUCTURE

In this part, we compare the performance of DMFL with its partial versions to evaluate the effectiveness of combining three sub-modules together.

We compare the DMFL model with its six partial combinations, which are:

- DMFL-F: Only SDAE-FM module is kept. The learned latent feature vectors are fed into the SDAE-FM module only.
- DMFL-D: Only the deep neural network module is kept. The learned latent feature vectors are fed into the deep neural network module only.
- DMFL-M: Only the metric learning model is kept. The learned latent feature vectors are fed into the metric learning part only.
- DMFL-FD: SDAE-FM and deep neural network modules are kept. The learned latent feature vectors are fed into the two modules.
- DMFL-FM: SDAE-FM and metric learning modules are kept. The learned latent feature vectors are fed into the two modules.
- DMFL-DM: Deep neural network and metric learning modules are kept. The learned latent feature vectors are fed into the two modules.

Figures 4 $\sim$ 6 demonstrate the Recall and AUC performance of each combination. As is shown in the figures, DMFL achieves best performance on recall@50, recall@100, AUC@50, AUC@100 among all three datasets. DMFL-FD, DMFL-FM and DMFL-DM perform a little worse than DMFL, while DMFL-F, DMFL-D and DMFL-M perform the worst. The experimental results prove that the combination of different sub-modules can help to efficiently improve recommendation quality since the hybridization absorbs the advantages of each component and compensates for the disadvantages of either.

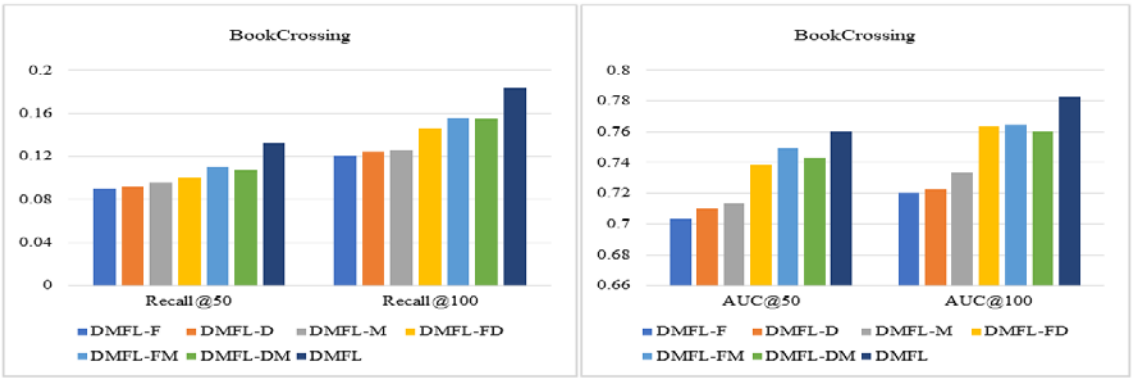


FIGURE 6. Recall and AUC on BookCrossing.

TABLE 2. Recall of baselines and the proposed model.

Dataset		FM	Wide &Deep	CML	DeepRec	CDL[1]	CDL[2]	DMFL
MovieLens-20M	Recall@50	0.4387	0.4535	0.4599	0.4489	0.4469	0.4603	**0.4885**
	Recall@100	0.5573	0.5884	0.5940	0.5610	0.5827	0.5951	**0.6102**
MovieLens-1M	Recall@50	0.5123	0.5532	0.5301	0.5317	0.5349	0.5423	**0.5632**
	Recall@100	0.5340	0.5695	0.5501	0.5612	0.5589	0.5601	**0.5984**
BookCrossing	Recall@50	0.1017	0.1039	0.1103	0.1001	0.0953	0.0995	**0.1424**
	Recall@100	0.1585	0.1613	0.1652	0.1559	0.1524	0.1568	**0.1902**

TABLE 3. AUC of baselines and the proposed model.

Dataset		FM	Wide &Deep	CML	DeepRec	CDL[1]	CDL[2]	DMFL
Movie Lens-20M	AUC@50	0.8104	0.8242	0.8301	0.8221	0.8169	0.8297	**0.8415**
	AUC@100	0.8301	0.8482	0.8511	0.8369	0.8465	0.8517	**0.8621**
Movie Lens-1M	AUC@50	0.7021	0.7632	0.7214	0.7345	0.7432	0.7521	**0.8054**
	AUC@100	0.7245	0.7834	0.7430	0.7580	0.7669	0.7790	**0.8323**
BookCrossing	AUC@50	0.7430	0.7471	0.7534	0.7373	0.7312	0.7363	**0.7635**
	AUC@100	0.7630	0.7661	0.7671	0.7602	0.7512	0.7573	**0.7902**

2) MODEL COMPARISON

In this part, we compare our model with six state-of-the-art baselines to evaluate the effectiveness of our model.

Tables 2 ∼ 3 illustrate the recommendation performance of all models in recall@50, recall@100 and AUC@50, AUC@100. It can be observed that DMFL almost outperforms all other state-of-the-art baselines on MovieLens-20M dataset, MovieLens-1M dataset and BookCrossing dataset.

We can see that FM performs the worst among these eight models, proving the effectiveness of learning deep-dimensional features since other seven models all both adopt deep learning to learn the latent feature vectors for users and items. According to the experimental results, DMFL achieves the second best recommendation results on recall@50, recall@100, AUC@50 and AUC@100 in all the dataset. In particular, DMFL outperforms Wide& Deep, CML, DeepRec, CDL^2, proving the combination of deep learning, metric learning and SDAE-FM can benefit the recommendation accuracy effectively.

3) THE IMPACT OF USER NUMBER

In this part, we selected 30%, 50%, and 100% of the user data on the three datasets, and conducted experiments under different user numbers to observe the impact of use number on the models. We choose recall@50 and AUC@50 as evaluation metrics.

Figures 7∼9 show the performance of all the models on three datasets respectively. It can be seen that as the number of users increases, the recall rate and AUC value of the DFML also increases slightly. It shows that DFML is overall stable and can achieve excellent recommendation results under different numbers of users. To be specific, it can be observed that the recommendation accuracy of all models is more stable in MovieLens-20M and MovieLens-10M than that in BookCrossing, proving that sparse data does have negative effect on recommendation performance. We can also find that DFML can always achieve better performance than the other six algorithms under the same conditions, indicating that our model have excellent recommendation quality.

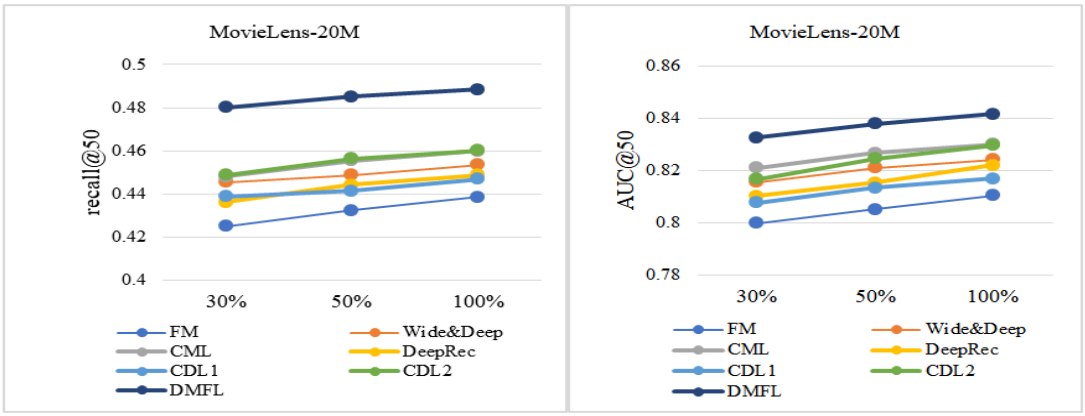


FIGURE 7. Recall@50 and AUC@50 of different number of users on MovieLens-20M.

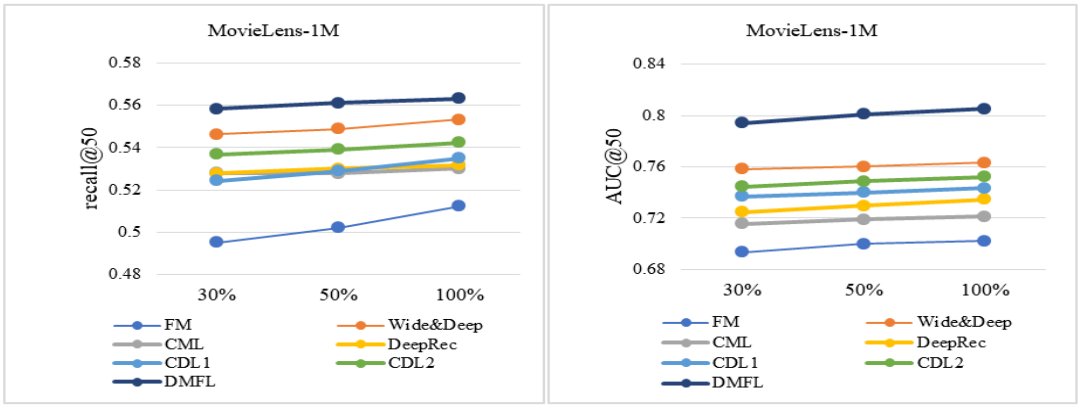


FIGURE 8. Recall@50 and AUC@50 of different number of users on MovieLens-1M.

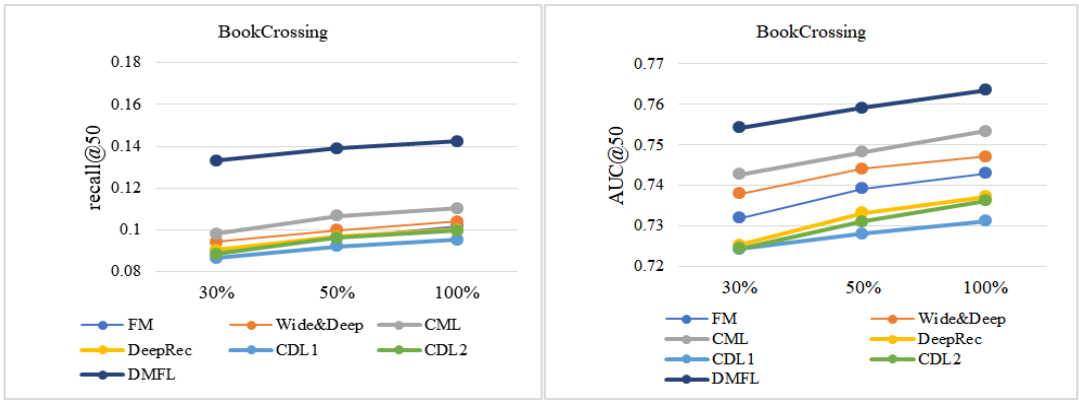


FIGURE 9. Recall@50 and AUC@50 of different number of users on BookCrossing.

V. CONCLUSION

In this paper, we proposed a deep hybrid recommendation model DMFL. DMFL inherits the advantages of deep learning, metric learning and factorization machine, getting rid of heavy manual feature engineering and exploring both linear and nonlinear relationship between users and items. Moreover, we conducted detailed experiments on three different real-world datasets, proving the effectiveness and efficiency of our model.

In the future, we will try to fuse more kinds of contextual information into recommendation, such as text, image and music. And we also want to incorporate other deep learning

methods such as attention mechanism into recommendation model to improve recommendation quality.

REFERENCES

[1] M. Balabanović and Y. Shoham, "Fab: Content-based, collaborative recommendation," *Commun. ACM*, vol. 19, no. 3, pp. 66–72, 1997.

[2] P. Kouki, J. Schaffer, J. Pujara, J. O'Donovan, and L. Getoor, "Personalized explanations for hybrid recommender systems," in *Proc. 24th Int. Conf. Intell. User Interfaces*, 2019, pp. 379–390.

[3] V. R. Kagita, A. K. Pujari, and V. Padmanabhan, "Virtual user approach for group recommender systems using precedence relations," *Inf. Sci.*, vol. 294, pp. 15–30, Feb. 2015.

[4] W. Chen, Z. Niu, X. Zhao, and Y. Li, "A hybrid recommendation algorithm adapted in e-learning environments," *World Wide Web*, vol. 17, no. 2, pp. 271–284, 2014.

[5] H. Wen, L. Fang, and L. Guan, "A hybrid approach for personalized recommendation of news on the Web," *Expert Syst. Appl.*, vol. 39, no. 5, pp. 5806–5814, 2012.

[6] E. Aslanian, M. Radmanesh, and M. Jalili, "Hybrid recommender systems based on content feature relationship," *IEEE Trans. Ind. Informat.*, to be published.

[7] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103.

[8] Z. Huang, J. Tang, G. Shan, J. Ni, Y. Chen, and C. Wang, "An efficient passenger-hunting recommendation framework with multi-task deep learning," *IEEE Internet Things J.*, to be published. doi: 10.1109/JIOT.2019.2901759.

[9] A. Joulin and T. Mikolov, "Inferring algorithmic patterns with stack-augmented recurrent nets," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 190–198.

[10] T. Xiao, S. Liang, and Z. Meng, "Hierarchical neural variational model for personalized sequential recommendation," in *Proc. World Wide Web Conf.*, 2019, pp. 3377–3383.

[11] Y. Shan, T. R. Hoens, J. Jiao, H. Wang, D. Yu, and J. C. Mao, "Deep crossing: Web-scale modeling without manually crafted combinatorial features," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 255–262.

[12] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. Int. Conf. World Wide Web*, 2017, pp. 173–182.

[13] Q. Liu, S. Wu, and L. Wang, "Multi-behavioral sequential prediction with recurrent log-bilinear model," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 6, pp. 1254–1267, Jun. 2017.

[14] S. Wang, Y. Wang, J. Tang, K. Shu, S. Ranganath, and H. Liu, "What your images reveal: Exploiting visual contents for point-of-interest recommendation," in *Proc. Int. Conf. World Wide Web*, 2017, pp. 391–400.

[15] S. Rendle, "Factorization machines," in *Proc. IEEE Int. Conf. Data Mining*, 2010, pp. 995–1000.

[16] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a 'siamese' time delay neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 1994, pp. 737–744.

[17] Z. Huang, G. Shan, J. Cheng, and J. Sun, "TRec: An efficient recommendation system for hunting passengers with deep neural networks," *Neural Comput. Appl.*, vol. 31, pp. 209–222, Jan. 2019. doi: 10.1007/s00521-018-3728-2.

[18] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. 14th Conf. Uncertainty Artif. Intell.*, pp. 43–52, 1998.

[19] A. B. Barragáns-Martínez, E. Costa-Montenegro, J. C. Burguillo, M. Rey-López, F. A. Mikic-Fonte, and A. Peleteiro, "A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition," *Inf. Sci.*, vol. 180, no. 22, pp. 4290–4311, 2010.

[20] C. A. Gomez-Uribe and N. Hunt, "The netflix recommender system: Algorithms, business value, and innovation," *ACM Trans. Manage. Inf. Syst.*, vol. 6, no. 4, p. 13, 2016.

[21] O. Kaššák, M. Kompan, and M. Bieliková, "Personalized hybrid recommendation for group of users: Top-N multimedia recommender," *Inf. Process. Manage.*, vol. 52, no. 3, pp. 459–477, 2016.

[22] A. Popescul, L. H. Ungar, D. M. Pennock, and S. Lawrence, "Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments," pp. 437–444, Jan. 2013, *arXiv:1301.2303*. [Online]. Available: https://arxiv.org/abs/1301.2303

[23] H. Zhang, I. Ganchev, N. S. Nikolov, Z. Ji, and M. O'Droma, "Hybrid recommendation for sparse rating matrix: A heterogeneous information network approach," in *Proc. IEEE 2nd Adv. Inf. Technol., Electron. Automat. Control Conf.*, Mar. 2017, pp. 740–744.

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

[25] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 353–362.

[26] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," in *Proc. 9th ACM Int. Conf. Web Search Data Mining*, 2016, pp. 153–162.

[27] X. Dong, L. Yu, Z. Wu, Y. Sun, L. Yuan, and F. Zhang, "A hybrid collaborative filtering model with deep structure for recommender systems," in *Proc. 21st AAAI Conf. Artif. Intell.*, 2017, pp. 1–7.

[28] 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*, 2015, pp. 1235–1244.

[29] W. Jian, J. He, K. Chen, Y. Zhou, and Z. Tang, "Collaborative filtering and deep learning based recommendation system for cold start items," *Expert Syst. Appl.*, vol. 69, pp. 29–39, Mar. 2017.

[30] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 447–456.

[31] X. He and T. S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2017, pp. 355–364.

[32] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 1059–1068.

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

[34] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide & deep learning for recommender systems," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, 2016, pp. 7–10.

[35] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: A factorization-machine based neural network for CTR prediction," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 1725–1731.

[36] Q. Zhang, J. Wang, H. Huang, X. Huang, and Y. Gong, "Hashtag recommendation for multimodal microblog using co-attention network," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 3420–3426.

[37] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," Jun. 2018, *arXiv:1806.01973*. [Online]. Available: https://arxiv.org/abs/1806.01973

[38] H. Gao, D. Kong, M. Lu, X. Bai, and J. Yang, "Attention convolutional neural network for advertiser-level click-through rate forecasting," in *Proc. World Wide Web Conf.*, 2018, pp. 1855–1864.

[39] A. van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 2643–2651.

[40] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proc. ACM Conf. Recommender Syst.*, 2016, pp. 233–240.

[41] H. Liang and K. Wang, "Top-k route search through submodularity modeling of recurrent POI features," Oct. 2017, *arXiv:1710.03852*. [Online]. Available: https://arxiv.org/abs/1710.03852

[42] J. Manotumruksa, C. Macdonald, and I. Ounis, "A contextual attention recurrent architecture for context-aware venue recommendation," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2018, pp. 555–564.

[43] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, "STAMP: Short-term attention/memory priority model for session-based recommendation," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 1831–1839.

[44] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," Nov. 2015, *arXiv:1511.06939*. [Online]. Available: https://arxiv.org/abs/1511.06939

[45] S. Okura, Y. Tagami, S. Ono, and A. Tajima, "Embedding-based news recommendation for millions of users," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 1933–1942.
[46] Z. Li, H. Zhao, Q. Liu, Z. Huang, T. Mei, and E. Chen, "Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 1734–1743.
[47] C. Lei, D. Liu, W. Li, Z. J. Zha, and H. Li, "Comparative deep learning of hybrid representations for image recommendations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2545–2553.
[48] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
[49] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," Feb. 2015, *arXiv: 1502.03167*. [Online]. Available: https://arxiv.org/abs/1502.03167
[50] 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, Feb. 2011.
[51] J. Rajasegaran, N. Karunanayake, A. Gunathillake, S. Seneviratne, and G. Jourjon, "A multi-modal neural embeddings approach for detecting mobile counterfeit apps," in *Proc. World Wide Web Conf.*, 2019, pp. 3165–3171.
[52] S. A. Khan and Z. A. Rana, "Evaluating performance of software defect prediction models using area under precision-Recall curve (AUC-PR)," in *Proc. 2nd Int. Conf. Advancements Comput. Sci.*, 2019, pp. 1–6.
[53] C.-K. Hsieh, L. Yang, Y. Cui, T.-Y. Lin, S. Belongie, and D. Estrin, "Collaborative metric learning," in *Proc. Int. Conf. World Wide Web*, 2017, pp. 193–201.
[54] W. Zhang, Y. Du, T. Yoshida, and Y. Yang, "DeepRec: A deep neural network approach to recommendation with item embedding and weighted loss function," *Inf. Sci.*, vol. 470, pp. 121–140, Jan. 2019.

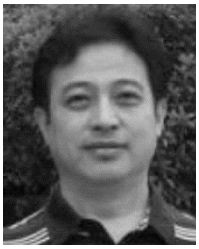
ZHENHUA HUANG was born in Fujian, China, in 1980. He received the Ph.D. degree in computer science from Fudan University, China, in 2008. He is currently a Professor with the School of Computer Science, South China Normal University. Since 2004, he has been published over 70 articles in various journals and conference proceedings. His research interests include database, data mining, big data analysis, recommender systems, natural language processing, and deep learning.

CHANG YU received the B.S. degree from the School of Computer Science and Software Engineering, East China Normal University. She is currently pursuing the master's degree in computer science with Tongji University. Her main research interests include recommendation systems, deep learning, and data mining.

JUAN NI received the master's degree from East China Normal University, Shanghai, China, in 2011. She is currently a Teacher with the School of Politics and Administration, South China Normal University. She has published more than 10 articles in various journals and conference proceedings. Her main research interests include educational data mining, educational big data analysis, and recommendation systems.

HAI LIU received the Ph.D. degree in computer science from Sun Yat-sen University, China, in 2010. He is currently an Associate Professor with the School of Computer, South China Normal University, China. His research interests include recommendation systems, deep learning, and text mining.

CHUN ZENG received the B.E. degree in software engineering from Huaihua University. He is currently pursuing the master's degree in software engineering with South China Normal University, China. His main research interests include database, data mining, machine learning, and recommender systems.



YONG TANG received the B.S. degree in computer science from Wuhan University, in 1985, and the Ph.D. degree from the University of Science and Technology of China, in 2001. He is currently a Professor and the Dean of the School of Computer Science, South China Normal University, where he also serves as the Director of the Services Computing Engineering Research Center of Guangdong Province. He has completed more than 30 research and development projects. He has authored or coauthored more than 100 publications in his research areas. His research interests include database and cooperative software, temporal information processing, social networks, and big data analytics.