Kernelized Deep Learning for Matrix Factorization Recommendation System Using Explicit and Implicit Information

Xiaoyao Zheng, *Member, IEEE*, Zhen Ni, *Senior Member, IEEE*, Xiangnan Zhong, *Member, IEEE*, and Yonglong Luo

***Abstract*— In the current matrix factorization recommendation approaches, the item and the user latent factor vectors are with the same dimension. Thus, the linear dot product is used as the interactive function between the user and the item to predict the ratings. However, the relationship between real users and items is not entirely linear and the existing recommendation model of matrix factorization faces the challenge of data sparsity. To this end, we propose a kernelized deep neural network recommendation model in this article. First, we encode the explicit user—item rating matrix in the form of column vectors and project them to higher dimensions to facilitate the simulation of nonlinear user—item interaction for enhancing the connection between users and items. Second, the algorithm of association rules is used to mine the implicit relation between users and items, rather than simple feature extraction of users or items, for improving the recommendation performance when the datasets are sparse. Third, through the autoencoder and kernelized network processing, the implicit data are connected with the explicit data by the multilayer perceptron network for iterative training instead of doing simple linear weighted summation. Finally, the predicted rating is output through the hidden layer. Extensive experiments were conducted on four public datasets in comparison with several existing well-known methods. The experimental results indicated that our proposed method has obtained improved performance in data sparsity and prediction accuracy.**

***Index Terms*— Deep learning, kernelized network, matrix factorization and data sparsity, recommender system.**

I. Introduction

With the continuous acceleration of the digital process of the physical world, the online world contains a huge amount of information resources. Due to the explosive growth of information, users cannot effectively obtain the information they are interested in, which leads to the problem of information overload [1]. The recommender system can recommend potential items of interest to users and realize the goal of maximizing the utility of information utilization by establishing the binary relationship between users and items. It has become one of the crucial intelligent technologies for understanding users. The binary relationship between users and items is constructed by using explicit information and implicit information. Explicit information mainly includes rating, text reviewers, and so on. Implicit information includes potential neighbor relationship, social trust, and so on. [2]. Therefore, recommender systems have been widely used in e-commerce [3], smart tourism [4], social network [5], market decision [6], microblog [7], app recommendation [8], and so on.

Traditional recommendation methods generally can be divided into three categories: content-based recommendation [9], [10], collaborative filtering recommendation [11], [12], and hybrid recommendation [13], [14]. Matrix factorization recommendation is one of the successful collaborative filtering methods, and it has a high recommendation accuracy in experimental datasets [15]–[17]. By projecting the rating matrix of users and items into the latent space of the same dimension, the latent feature vectors of users and items are obtained by matrix factorization. Then, the ratings between users and target items are calculated in the form of inner product, so as to predict the relationship between users and target items. In order to further enhance the performance of the matrix factorization algorithm, other works have been proposed to improve the matrix factorization technologies by integrating trust [18], time [19], context [20], and other information [21], [22]. It has been proven to be a great help in improving the quality of recommendations by incorporating auxiliary data in matrix factorization recommendation approaches.

However, in practical applications, the recommender system using the matrix factorization technique still needs improvement. The first reason is that the data can be sparse and the number of online users and items is huge in the recommender system. Compared with the number of users and items in the whole network system, the number of items that each user concerned and the number of items consumed by users are extremely small, respectively. Therefore, data sparsity leads to the low accuracy of the recommender system [23], [24]. The second reason is the matrix factorization that employs the

Manuscript received February 4, 2021; revised December 13, 2021 and March 2, 2022; accepted June 9, 2022. This work was supported in part by National Science Foundation under Grant 2047064, Grant 2047010, and Grant 1947419; in part by the Natural Science Foundation of China under Grant 61972439; in part by the Natural Science Foundation of Anhui Province under Grant 1808085MF172; and in part by the Key Program in the Youth Elite Support Plan in Universities of Anhui Province under Grant gxyqZD2019010. *(Corresponding author: Zhen Ni.)*

Xiaoyao Zheng and Yonglong Luo are with the School of Computer and Information, Anhui Normal University, Wuhu 241002, China (e-mail: zxiaoyao_2000@163.com; ylluo@ustc.edu.cn).

Zhen Ni and Xiangnan Zhong are with the Department of Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431 USA (e-mail: zhenni@fau.edu; xzhong@fau.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3182942.

Digital Object Identifier 10.1109/TNNLS.2022.3182942

linear dot product requiring the same dimension of the user and the item [25]. These strict conditions lead to some deficiencies of matrix factorization method. For example, it is difficult to adapt to the semantic interpretation of different types of items and users in real recommendation environments.

In recent years, deep learning has been used in natural language processing, computer vision, and autonomous driving [26]–[30]. The common deep learning methods used in the recommendation system include autoencoder-based collaborative filtering recommendation [31]–[33], multilayer perceptron (MLP)-based recommendation [34], [35], convolutional neural network (CNN)-based recommendation [36], [37], recurrent neural network (RNN)-based recommendation [38], [39], restricted Boltzmann machine (RBM)-based recommendation [40], [41], graph neural networks for recommendation [42], [43], deep reinforcement learning (DRL) for recommendation [44], [45], and other forms of deep learning recommendation [25], [46], [47]. However, when the deep learning recommendation method is used to model the key factor in predicting ratings between the user and item, it applies an inner product on the latent features of users and items. In addition, the size and sparsity of the large dataset will lead to poor learning efficiency of deep learning method and ultimately result in low recommendation accuracy.

Motivated by the above challenges, this article proposes a kernelized deep neural network for matrix factorization (K-DNNMF) method to bring the advantages of deep learning, kernel function, and matrix factorization. This method maps users and items to latent spaces of different dimensions through a kernelized network. It employs association rules to mine the potential relationship between users and items as implicit data and connects with the explicit data of user—item rating matrix. Then, a multilayer perceptual network is designed for training implicit and explicit data to predict the corresponding ratings. The contributions of this article are summarized as follows.

1) A kernelized network named KernelNet proposed in this article projects the latent factor vectors of users and items to a space with higher dimensions. The nonlinear interaction between user and item is simulated by KernelNet in high-dimensional space. It can solve the problem of low prediction accuracy caused by using the dot product linear method in the matrix factorization recommendation. Thus, the performance of the recommender systems can be effectively improved by the nonlinear method using KernelNet.
2) Compared with existing methods, we use association rules to mine the potential relationship between users and items. We then fuse it as implicit data with explicit data through the MLP, rather than using simple user or item feature data directly as implicit data as in the literature [12], [16], [25], [32]. Thus, the problem of data sparsity can be effectively alleviated by mining and integrating the implicit data of user—item relationship in this article.
3) We conduct extensive experiments on four public datasets. The experimental results show that our method has the promising performance compared with several well-known matrix factorization and deep learning recommendation methods.

The remainder of this article is organized as follows. We describe several typical matrix factorization methods and deep learning recommendation model in Section II. We propose a novel kernelized deep neural network recommendation model in Section III. Moreover, we present the details of the kernelized network principle and recommendation framework in this section. We present a performance comparison of our model with several well-known methods and provide the evaluation results in Section IV. In Section V, this article is summarized, and the future research directions are prospected.

II. Related Work

Since this article focuses on deep neural network matrix factorization recommendation method, this section mainly reviews some significant works on matrix factorization recommendation methods and deep learning recommendation models.

A. Matrix Factorization Recommendation Method

One of the most successful collaborating filtering methods for recommender systems is the matrix factorization recommendation [48]. It is generally known that the matrix factorization method focuses on using low-rank approximations to fit the user—item ratings matrix and projects items and users to the same latent factor space. The latent space tries to explain ratings by characterizing both items and users on factors inferred from user feedback automatically, such as ratings and text reviews. The latent factor vectors of user and item can be obtained by minimizing the sum-squared errors objective function. The predicted ratings are then obtained by the inner product of the latent factor vectors of the users and the items, which is a linear form of computation. To start with, we briefly review the basic low-rank matrix factorization approach called probabilistic matrix factorization model (PMF) [49], which does not take any social factors into consideration. A user latent factor vector $p_u \in \mathbb{R}^f$ associated with user u and an item latent factor vector $q_i \in \mathbb{R}^f$ associated with item i can be obtained by using a stochastic gradient descent algorithm. Then, the prediction is made by taking the inner product: $\hat{r}_{\text{ui}} = p_u \cdot q_i^T$. The specific objective function is shown in the following equation:

$$\Psi = \frac{1}{2}\sum_{u}\sum_{i\in I_u}\left(r_{\text{ui}} - p_u q_i^T\right)^2 + \frac{\lambda}{2}\left(\sum_{u}\|p_u\|_F^2 + \sum_{u}\|q_i\|_F^2\right) \tag{1}$$

where λ is a regularized parameter to avoid overfitting and $\|\cdot\|_F^2$ denotes the Frobenius norm. Known that rating values can be calculated easily according to the equation $\hat{r}_{\text{ui}} = p_u \cdot q_i^T$ for any user—item combination (u, i). It can be seen from (a) that PMF constructs the interaction of user and item latent factors. Assume that each dimension of the latent space is independent of each other and linearly combining them with the same weight. Thus, matrix factorization can be regarded as a linear model of latent factors.

In addition, there are a variety of variants based on the basic matrix factorization model [50]–[54]. However, the simple linear prediction of inner product limits the adaptability and performance improvement of matrix factorization method. Therefore, changing the prediction method of linear dot product will play a key role in improving the recommendation performance.

B. Deep Learning Recommendation Model

The impact of deep learning is pervasive, and its effectiveness has recently been demonstrated in research on information retrieval and recommender systems. This article mainly introduces the recommender system based on deep learning technology. One of the most successful applications of deep learning recommendation methods is the use of autoencoders for collaborative filtering recommendations. Sedhain *et al.* [31] proposed autoencoder-based recommendation method, named AutoRec. This model employed user-based ratings or item-based ratings as the input, and the collaborative filtering recommendation was realized by constructing the minimum error between the encoded input and the decoded output as the optimization objective function. Strub *et al.* [32] proposed item-based collaborative filtering neural network (I-CFN) and user-based collaborative filtering neural networks (U-CFN) recommendation methods by extending AutoRec, which incorporated such as user profiles and item descriptions to alleviate the sparsity and cold start influence. However, the autoencoder-based collaborative filtering recommendation method has two shortcomings: one is that the rating should be an integer and the other is that the decomposition of the rating aggravates the data sparsity. MLP is an accurate and efficient network that approximates any measurable function. Therefore, it can be used to replace the linear inner product calculation in matrix factorization. He *et al.* [34] presented a general framework named neural collaborative filtering (NCF) method by replacing the inner product with a multilayer perceptron that can learn an arbitrary function from data. Xue *et al.* [35] proposed a deep matrix factorization model by using MLP architecture, which constructed a user—item matrix with explicit ratings and nonpreference implicit feedback. By constructing a multilayer, nonlinear, layer-to-layer interconnection network structure, MLP can use the network structure to approach a complex multivariate function as a training target and to learn the original features of the dataset from many unlabeled training datasets. A graph neural network can be well applied to represent user and item features and social relationship, which is helpful to improve the performance of recommender system. Guo and Wang [42] proposed a deep graph neural network-based social recommendation (GNN-SoR) framework, which used two graph neural networks to abstract user and item features and fused the encoded the user and item space into a matrix factorization method to complete missing rating values. Other recommendation method-based deep learning has different advantages and disadvantages in different application fields. This article mainly focuses on the accuracy of rating prediction based on user—item rating's matrix. We propose a novel kernelized deep neural network recommendation model combined autoencoder and MLP. To overcome the linear inner product, we make use of kernel function to project the user and item latent factor to high-dimensional space. Moreover, we employ MLP to integrate explicit and implicit data, instead of simple weighted sum to achieve rating prediction, so as to improve the performance of the recommender system.

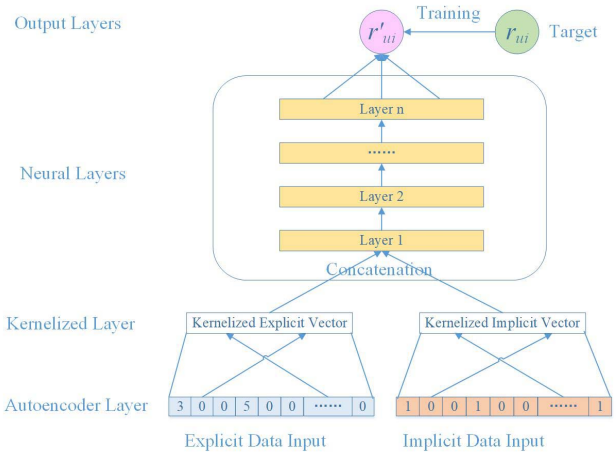


Fig. 1. Proposed K-DNNMF structure.

III. Proposed Deep Neural Network Matrix Factorization Recommendation Method

We first propose a K-DNNMF recommendation model based on KernelNet. It takes user—item ratings and implicit information mined by association rules from user and item features as input. Latent factor vectors based on users and items are generated through autoencoders and mapped into another high space by kernelized networks, which are used as inputs to MLP networks. Then, user and item interaction functions are learned through MLP, and thus, the ratings of users to items are obtained through output layers.

A. Deep Learning Structure With KernelNet

In this article, a recommendation structure for deep learning based on kernelized network is constructed by combining autoencoder, kernelized network, and multilayer perceptron, as shown in Fig. 1, in order to provide an alternative solution direction for improving recommendation performance under the framework of deep learning. The structure is mainly composed of four parts.

1) *Autoencoder Layer:* It is responsible for decomposing the user—item rating matrix into vectors based on the user or item (i.e., rows or columns). Because the rating vector is reconstructed by autoencoder according to the rating range, thus the sparsity is multiplied.
2) *Kernelized Layer:* It projects the coding of the autoencoder layer to a high-dimensional space, so as to improve the accuracy of feature expression of users and items and reduce the impact of data sparsity.
3) *Neural Layers:* The rating vector after encoding and kernelization is then passed through the MLP to transform the linear model decomposed by matrix factorization

into a nonlinear model, which improves the plasticity and robustness of the whole model.

4) *Output Layer:* It outputs the interaction value between the hidden layer's user and the item's latent factor vector through the activation function, namely, the predicted rating.

This structure is a general framework, which can be used in different combinations according to different recommendation tasks. Based on this structure, the entire recommendation process can be divided into four steps, as shown in Fig. 2.

1) *Data Processing:* The first step is to decompose the rating matrix into row or column vectors according to the original dataset, that is, explicit data. In addition, based on the user and item features' data, the relationships between the items and users are mined to generate implicit data.
2) *Encoding and Kernelized Processing:* The second step is to encode the explicit data and implicit data through the autoencoder. Then, we project the two pieces of information to a high-dimensional vector space with the kernel function to complete the feature representation of user and item data.
3) *Concatenation and Training:* The third step is to train the interaction function of the latent factor vector between the user and the item by means of the nonlinear model through the MLP. It avoids the simple linear weighted sum of the explicit and implicit data, which can simulate the user and item interactions effectively.
4) *Prediction and Evaluation:* The fourth step is to predict the target user's rating of the item based on the trained user—item interaction function and to realize the item recommendation.

B. Deep Learning Matrix Factorization Recommendation Method by Using Explicit Information

Autoencoder is generally composed of a three-layer network, in which the number of neurons in the input layer and output layer is the same. The number of neurons in the middle layer is smaller than that in the input layer and output layer. In the process of network training, for each training sample, the autoencoder tries to make the output signal and the input signal similar and uses a reconstruction error to express. Moreover, autoencoder can form a deep structure through layers and layers of training.

Assume that the user set and item set in the recommendation system are $U = \{u_1, u_2, \ldots, u_m\}$ and $I = \{i_1, i_2, \ldots, i_n\}$, respectively. The rating matrix is denoted as $R = \{r_{ij}\}^{m\times n}$, and each item can be regarded as a column vector, namely, $r_i = \{r_{1i}, r_{2i}, \ldots r_{mi}, \}$. Then, the rating matrix can be expressed as n 1-D vectors with m dimensions, i.e., $R = \{r_1, r_2, \ldots, r_n\}$. Similarly, each user can be expressed as a row vector. This article takes item vector as an example to describe it. Therefore, the objective function of the autencoder can be expressed in the following equation:

$$\mathcal{L} = \underset{\theta}{\operatorname{argmin}} \sum_{r_i \in R} \|r_i - h(r_i;\theta)\|_{\mathcal{O}}^2 + \frac{\lambda}{2} \cdot \left(\|W\|_F^2 + \|V\|_F^2\right) \tag{2}$$

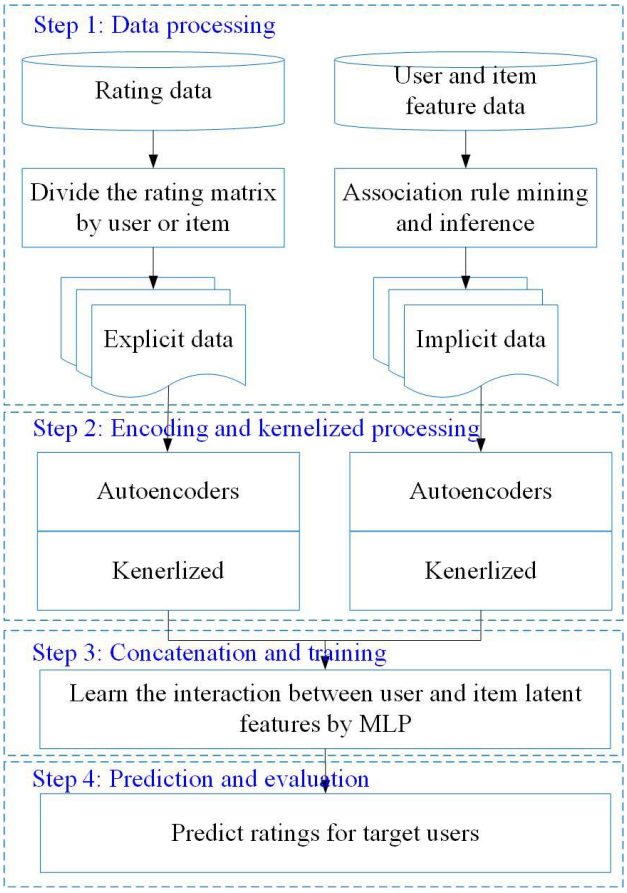


Fig. 2. Recommendation process based on the proposed structure.

where $h(r_i;\theta) = f(W\cdot g(Vr_i+\mu)+b), \theta = \{W, V, \mu, b\}$, and $f(\cdot)$ and $g(\cdot)$ are identity and sigmoid activation functions for autoencoder layer, respectively. As the autoencoder increases the data sparsity, it reduces the correlation between the data. In this article, we attempt to project the output information of the autoencoder to the high-dimensional space by the kernel function to enhance the expressiveness of the data. The effect of kernel function can also be seen as enhancing the connection between users or items. Next, W and V in the autoencoder can be reparameterized by the kernel function. Then, we get $V_{ij} = \alpha^{(l)} K(v_i^{(l)}, v_j^{(l)})$ and $W_{ij} = \beta^{(l)} K(w_i^{(l)}, w_j^{(l)})$, where $K(\cdot,\cdot)$ is a kernel function. It can project a d-dimensional space vector to $\hat{d}$ dimension by inner product, i.e., $K(u,v) =< \psi(u), \psi(v) >=< \hat{u}, \hat{v} >$. ψ is an embedded function that projects d dimensions onto $\hat{d}$. Through the above reparameterization, we redefine a kernelized neural network named KernelNet, as shown in Fig. 3. The specific mathematical form is defined as follows:

$$\begin{aligned} h\left(r_i^{(l)};\theta\right) &= f\left(\beta^{(l)} \sum_{i,j} K\left(w_i^{(l)}, w_j^{(l)}\right)\right. \\ &\quad \left. \cdot\, g\left(\sum_{i,j} \alpha^{(l)} K\left(v_i^{(l)}, v_j^{(l)}\right) r_i^{(l-1)} + \mu_i^{(l)}\right) + b_i^{(l)}\right). \end{aligned} \tag{3}$$

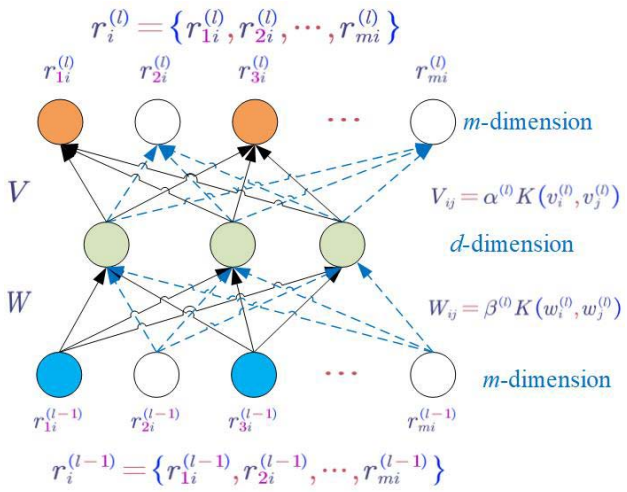


Fig. 3. Schematic of kernelized neural network.

The kernelized vector dimension $\hat{d}$ in KernelNet is determined by the embedded kernel function $K(.,.)$ or ψ. Therefore, the kernel function has a important influence on the performance of KernelNet. According to the optimal kernel function search method proposed in the literature [55], we choose four radial basis function (RBF) as the candidate Kernel, including Gaussian kernel, Laplacian kernel, log kernel, and generalized T-student kernel. The specific definition of the four kernel functions is $K(u, v) = \exp(-\gamma \|u - v\|^2)$, $K(u, v) = \exp(-(\|u - v\|/\sigma))$, $K(u, v) = -\log(\|u - v\|^d + 1)$, and $K(u, v) = (1/1 + \|u - v\|^d)$. Because the RBF kernel can map the sample data to a higher dimensional space, it can deal with the nonlinear feature's expression of the sample data. Moreover, we select the optimal kernel function as KernelNet's kernel function through experiments, which are detailed in Section IV.

The kernel functions can be abstracted into the mathematical description of RBF kernel function $K(u, v) = \alpha\Psi(D(u, v))$, where $D(\cdot, \cdot)$ is a distance function and $\alpha \in R$. Its essence is to map the d-dimensional space of u and v vectors to the $\hat{d}$-dimensional space of Ψ function. This is to realize the nonlinear expression of high-dimensional space. After determining the kernel function, according to (3), we get the parameters $\theta = \{K(w_i^{(l)}, w_j^{(l)}), K(v_i^{(l)}, v_j^{(l)}), \alpha^{(l)}, \beta^{(l)}, \mu^{(l)}, b^{(l)}\}$ after reparametrization. Therefore, according to the embedded KernelNet definition, the objective function becomes (4), which is specifically updated as follows:

$$\begin{aligned}\mathcal{L} = \underset{\theta}{\operatorname{argmin}} & \sum\nolimits_{r^{(i)} \in R} \|r_i - h(r_i; \theta)\|_{\mathcal{O}}^2 \\ & + \lambda_1 \left(\|\alpha\|_F^2 + \|\beta\|_F^2\right) \\ & + \lambda_2 \left(\left\|K\left(w_i, w_j\right)\right\|_F^2 + \left\|K\left(v_i, v_j\right)\right\|_F^2\right). \end{aligned} \tag{4}$$

The objective function can be realized by the limited-memory Broyden–Fletcher-Goldfarb–Shanno (L-BFGS) optimizer.

C. Mining Implicit Information to Enhance Recommendation

In this section, we first introduce the details of association rules algorithm to mine the hidden relation between users and items, namely, implicit data. Then, we propose to fuse the implicit data with the explicit data as a new input to train the deep learning model as introduced as above.

1) Building Implicit Data Using Association Rules: Data sparsity is one of the toughest challenges of a recommender system. Recommendation performance can be improved by increasing implicit information of users and items. In this article, we mainly use an association rule algorithm to mine the rules between user features and item features and build an implicit user—item relationship matrix based on the above rules. Then, we combine the implicit data obtained from mining with the explicit rating data to further improve the recommendation accuracy.

Suppose that the user's feature attribute set is denoted as F_U, and the item's feature attribute set is denoted as F_I. We first add a user and an item feature values to the dataset $D = \{F_U, F_I\}$ if the user and the item exist rating in the rating matrix R and the rating level is greater than the median. For example, if user u_i gives item i_j a rating of 4 on a scale of 1–5 so that the median is 3 and the rating 4 is greater than the median. Then, we add a feature record of user u_i and item i_j to D. For dataset D to be mined, feature rules of items of potential interest to users are given, which are specifically defined as follows:

$$\text{Support}(F_U, F_I) = P(F_U F_I) = \frac{\text{Freq}(F_U F_I)}{\text{Freq}(\text{all(samples)})}. \tag{5}$$

Accordingly, we give the confidence definition of the feature rules of the items that users are potentially interested in as follows:

$$\text{Confidence}(F_I \Rightarrow F_U) = P(F_U|F_I) = \frac{P(F_U F_I)}{P(F_I)}. \tag{6}$$

After the feature rules of items of potential interest to users are obtained by mining D, we set $s_{ij} = 1$ if the features of user u_i and item i_j meet the feature rules. Then, we get the implicit relation matrix $S = \{s_{ij}\}^{m \times n}$ of the item and user by traversing the dataset of the user and item. Furthermore, we use an autoencoder and KernelNet to get $h(s_i; \theta_s) = f(W_s \cdot g(V_s s_i + \mu_s) + b_s)$ for implicit dataset S in the same way as for explicit data, where $\theta_s = \{W_s, V_s, \mu_s, b_s\}$.

The implicit data mined in this article are different from the traditional implicit data that directly take user and item features data as implicit data. We focus on generating implicit data through feature data mining and discover the potential relationship between user and item. Therefore, the implicit data obtained in this article have stronger internal relations and plasticity.

2) Fusing Explicit and Implicit Data Through MLP: Many existing recommendation models are essentially linear methods. MLP can change the existing recommendation methods and transform it into a more reasonable nonlinear method, which can be interpreted as neural extension. This section focuses on the integration of rating data and implicit auxiliary data through MLP to achieve the rating prediction of the item. However, a simple vector concatenation cannot explain any interaction between the explicit data and the implicit data. To solve this problem, we add a hidden layer over the concatenated vectors using a standard MLP to learn the

interaction between the explicit data and the implicit data, as shown in the following equation:

$$\begin{aligned} z_1 &= \phi_1(h(r_i,\theta), h(s_i,\theta_s)) = \begin{bmatrix} h(r_i,\theta) \\ h(s_i,\theta_s) \end{bmatrix} \\ z_2 &= \phi_2(z_1) = \phi_2(Q_2 z_1 + c_2) \\ &\cdots\cdots \\ z_n &= \phi_2(z_{n-1}) = \phi_2(Q_n z_{n-1} + c_{n-1}) \\ \hat{r}_{ij} &= \sigma(H\phi_n(z_{n-1})) \end{aligned} \tag{7}$$

where Q_i denotes the weight matrix of MLP layer i, with $i = \{2, 3, \ldots, n\}$, c_i denotes the bias vector of layer i, and ϕ_i denotes the activation function of layer i. For activation functions ϕ_i, sigmoid, tanh, ReLU, or other activation functions can be selected depending on the recommendation task. H represents the edge weight of the output layer and σ is the sigmoid activation function of the output layer. ReLU is a nonlinear function, and it can fit nonlinear mapping and reduce overfitting. Therefore, the ReLU function is selected as the activation function in this article.

Since initialization plays an important role in the convergence and performance of the deep learning model, the K-DNNMF model proposed in this article consists of the explicit data and implicit data through three kinds of networks, namely, the autoencoder, KernelNet, and MLP. We propose a two-stage training to realize the initialization of the K-DNNMF. First, we train the autoencoder and KernelNet until it converges. Then, the parameters of autoencoder and KernelNet are used as the initialization parameters of K-DNNMF model, and the MLP network is trained on this basis. We use the Adam (adaptive moment estimation) optimizer because it is a method to calculate the adaptive learning rate of each parameter and has a fast convergence speed. In addition, it is generally recommended that users select items to focus on a limited number of features, so the data are considered to be interpreted in a low-dimensional model, which is the ideal for KernelNets setup and also for K-DNNMF model training. In general, the user will not pay too much attention to the number of features when selecting an item, so the data are considered to be able to be interpreted in a lower dimensional model, so this is very suitable for the KernelNets setup and also beneficial for the K-DNNMF model training.

IV. Experiment of Studies

In this section, we will report the extensive experiments that we have conducted to evaluate the performance of our proposed method. We will start with a discussion on the experimental setup of our evaluation, which will cover the information about four public datasets, two performance metrics, six benchmark models to be used for comparative study, and the parameter settings. The detailed experimental results are reported in the second half of this section.

A. Experimental Setup

1) Experimental Datasets: In order to test the model performance and comparative experiments thoroughly, four public datasets with different ratings and sparseness were selected for verification, including Yahoo Music,[1] Douban,[2] Flixster,[3] and MoiveLens.[4] The details of the datasets are shown in Table I. First, the sparsity of these four datasets is different. Among them, Movielens has the highest data density, while Yahoo Musci has the lowest. In addition, these four datasets have different ratings and sizes, such as Yahoo Music, which is on a 100-point scale. All these problems challenge the universality of the comparative methods.

2) Performance Metrics: In our work, we randomly designate 10% of the given ratings as validation data and the remaining 90% as the training data in the above dataset. The performance measures we use in our experiments are root mean square error (RMSE) and mean absolute error (MAE) that are the most popular accuracy measures. RMSE and MAE are defined as follows:

$$\text{RMSE} = \sqrt{\frac{\sum_{(u,i)\in T}\left(R_{\text{ui}} - \hat{R}_{\text{ui}}\right)^2}{|T|}} \tag{8}$$

$$\text{MAE} = \frac{\sum_{(u,i)\in T}\left|R_{\text{ui}} - \hat{R}_{\text{ui}}\right|}{|T|} \tag{9}$$

where T is the training dataset and $|T|$ is the number of samples. The smaller the errors, the better the performance.

3) Benchmark Models: The following recommendation models are involved in our experimental evaluation for performance comparative study. We benchmark K-DNNMF with six recent matrix factorization models and deep learning recommendation approaches, which are listed as follows.

1) *PMF [49]:* This model uses basic matrix factorization techniques without considering any social factors, which is a well-known probabilistic linear model with Gaussian observation noise. PMF model is one of the most commonly used recommendation methods of matrix factorization.
2) *Singular Value Decomposition Model Integrated With Explicit and Implicit Feedback (SVD++) [51]:* This model takes both the explicit and implicit influence of user—item ratings into account, based on the singular value decomposition model, to generate predictions, which yields greater prediction accuracy. However, its essence is still a linear prediction model.
3) *Inductive Graph-Based Matrix Completion (IGMC) Model [29]:* This model is an IGMC model, which trains a deep graph neural network based on one-hop subgraphs around (user and item) pairs generated from the rating matrix and maps these subgraphs to their corresponding ratings.
4) *U-CFN [32]:* This method is deep learning method based on an extension autoencoders. The U-CFN model performs a factorization of the matrix of ratings by using user-oriented vectors. It deploys the denoising techniques, which makes the U-CFN more robust and incorporates the side information such as user profiles

[1] https://webscope.sandbox.yahoo.com/
[2] https://www.douban.com/note/415374609/
[3] https://figshare.com/articles/Flixster-dataset_zip/
[4] https://grouplens.org/datasets/movielens/

TABLE I
STATISTICS OF ORIGINAL DATASETS

Datasets	User numbers	Item numbers	Rating numbers	Sparsity	Rating levels
YahooMusic	3,000	3,000	5,335	0.9994	0.1,2,,100
Douban	3,000	3,000	136,891	0.9848	1,2,,5
Flixster	3,000	3,000	26,173	0.9971	0.5,1,,5
Movielens	6,040	3,706	1,000,209	0.9553	1,2,,5

TABLE II
COMPARISON OF RECOMMENDATION ACCURACY ON YAHOO MUSIC

Models	PMF	SVD++	U-CFN	I-CFN	IGMC	GNN-SoR	K-DNNMF(EX)	K-DNNMF(EI)
RMSE	39.567	35.776	19.436	18.258	19.100	18.725	**18.491**	**17.891**
MAE	26.598	24.935	13.033	12.419	13.391	12.557	**12.632**	**11.780**

TABLE III
COMPARISON OF RECOMMENDATION ACCURACY ON DOUBAN

Models	PMF	SVD++	U-CFN	I-CFN	IGMC	GNN-SoR	K-DNNMF(EX)	K-DNNMF(EI)
RMSE	0.815	0.809	0.705	0.691	0.721	0.712	**0.704**	**0.679**
MAE	0.617	0.614	0.497	0.466	0.510	0.502	**0.494**	**0.462**

and item descriptions to mitigate the sparsity and cold start influence.

5) *I-CFN [32]:* This model is consistent with the U-CFN model, except that the input is an item-oriented rating vector.
6) *Deep Graph Neural Network-Based Social Recommendation Model [42]:* The GNN-SoR model uses two deep graph neural networks to represent the user and item features and fuse the encoded the user and item space into a matrix factorization method to complete missing rating values.
7) *Kernelized Deep Neural Network for Matrix Factorization Method:* This is the model proposed in this article, which sets up a kernelized network named KernelNet based on an autoencoder and then constructs an MLP network to realize rating prediction. The input data in this article are divided into two forms. If only the explicit data such as the rating matrix are used as the input, we call it K-DNNMF(EX). If implicit information between the item and the user is mined through association rules algorithms and is used as input along with the explicit data, we call it K-DNNMF(EI).

4) Experimental Setting: We implemented our proposed methods based on TensorFlow2.1 and Python 3.7. To determine hyperparameters of K-DNNMF model, we randomly sampled 10% data from datasets as the validation data and tuned hyperparameters on it. The two-stage optimization is adopted. In the first stage, the hyperparameter θ and θ_s learning of autoencoder and KernelNet is trained according to (4). In the second stage, for MLP networks that are trained from scratch, we initialized model parameters with a Gaussian distribution and optimize the model with Adam optimizer. We set the batch size at 512 and the learning rate at 0.001 as the default settings in MLP. Meanwhile, we set the architecture MLP layers number to 3. Since the last hidden layer of MLP determines the model capability, we term it as predictive factors and evaluated the factors of 32. For the KernelNet, we use latent factors vector dimension to 32 and all hidden layers have a size of 512. Parameters in the kernel function are set by default, i.e., $\gamma = 1$, $\sigma = 1$, and $d = 2$. As for the optimization of kernel functions, the experiment in this article focuses on the influence of different types of kernel functions on the recommendation performance. The setting of kernel function parameters is not within the scope of this article due to space limitations.

B. Experimental Results

This section is mainly carried out from three aspects: sensitivity of model parameters, a recommendation performance comparison between different models, and performance comparison in the case of sparse data.

1) Performance Comparison:

a) Recommendation accuracy: In this section, we mainly analyze the recommendation accuracy of the model proposed in this article and the other six models on the four datasets. It can be seen from Tables II to V that the KDNNMF(EI) model with integrated explicit and implicit data has the best performance on both RMSE and MAE metrics. In the recommendation models compared with this article, the inputs of U-CFN, I-CFN, GNN-SoR, and the K-DNNMF(EI) model employ rating data and auxiliary information. PMF, SVD++, IGMC, and K-DNNMF(EX) proposed in this article only use the rating matrix as input data. From the above comparison experimental results, it can be concluded that from the perspective of input data, the K-DNNMF(EX) proposed in this article basically improves the recommendation accuracy by more than 5% compared with other models that only use the rating matrix, and it has the optimal recommendation performance. Similarly, in the approaches using both explicit data and auxiliary data, the K-DNNMF(EI) model proposed in this article shows more superior recommendation accuracy, with its accuracy basically increased by more than 3%. Especially in the case of sparse datasets, the method proposed in this article has more obvious advantages. In general, the following conclusions can be drawn from the above experimental results. First, the recommendation model using deep learning is more competitive than traditional PMF and SVD++. Second, adding auxiliary information or implicit data can improve the recommendation performance. Compared

TABLE IV
COMPARISON OF RECOMMENDATION ACCURACY ON FLIXSTER

Models	PMF	SVD++	U-CFN	I-CFN	IGMC	GNN-SoR	K-DNNMF(EX)	K-DNNMF(EI)
RMSE	0.901	0.889	0.835	0.811	0.872	0.851	**0.824**	**0.807**
MAE	0.712	0.689	0.574	0.539	0.679	0.663	**0.566**	**0.538**

TABLE V
COMPARISON OF RECOMMENDATION ACCURACY ON MOIVELENS

Models	PMF	SVD++	U-CFN	I-CFN	IGMC	GNN-SoR	K-DNNMF(EX)	K-DNNMF(EI)
RMSE	0.883	0.875	0.857	0.832	0.855	0.847	**0.845**	**0.827**
MAE	0.697	0.689	0.677	0.634	0.676	0.660	**0.659**	**0.630**

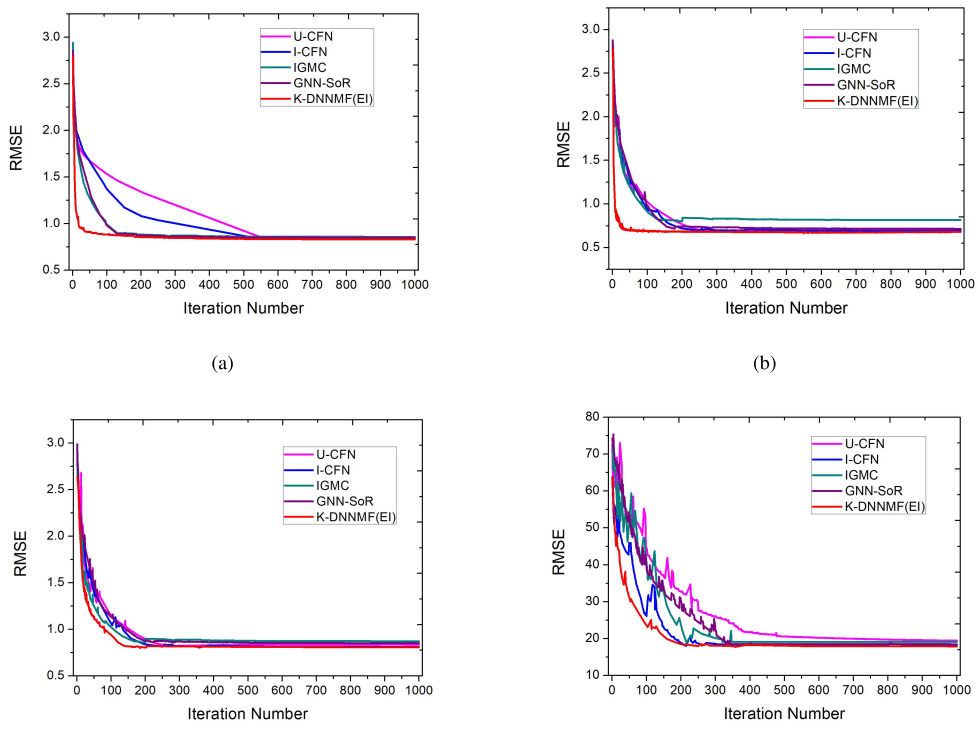


Fig. 4. Comparative analysis of the convergence speed of each model. (a) Convergence speed on MovieLens. (b) Convergence speed on Douban. (c) Convergence speed on Flixster. (d) Convergence speed on Yahoo Music.

with the K-DNNMF(EX) without auxiliary information, our proposed K-DNNMF(EI) further improved the recommendation performance after adding implicit data. Finally, both from the perspective of input data and the recommendation theoretical method, the kernelized method proposed in this article has better recommendation performance. In addition, the two models proposed in this article, K-DNNMF(EX) and K-DNNMF(EI), are not isolated. If the recommender system can provide auxiliary information, the model can be automatically changed into K-DNNMF(EI). If not, the system can still be trained as K-DNNMF(EX) without the need for manual adjustment. Therefore, the method proposed in this article has good adaptability.

b) Speed of convergence: In this section, we investigate both the convergence speed of different models on the four datasets. Since the recommendation method-based deep learning is different from the traditional matrix factorization method, we only compare the convergence speed on the four deep learning methods with our K-DNNMF(EI) model.

The experimental results in Fig. 4(a) show that when the number of iterations reaches 200, our method is close to the optimal value. I-CFN and U-CFN can reach the optimal value only after about 600 iterations, and IGMC and GNN-SoR methods using the graph neural network are about 300 iterations. The experiments of convergence on Douban, Flixster, and Yahoo Music in Fig. 4(b)–(d) show that the K-DNNMF(EI) method proposed in this article also has the fastest convergence speed. When the data are sparse, the convergence rate increases rapidly. However, these methods also produce fluctuation in the process of convergence. Because the kernelized network proposed in this article can map the input column vectors to a higher dimensional space, the nonlinear problem can be transformed into a higher dimensional to solve a linear problem, and the efficiency is higher. I-CFN and U-CFN methods add side information (item or user feature) in each layer of the network to increase the training time. In addition, IGMC and GNN-SoR methods use the graph neural network method to build subgraphs of users or

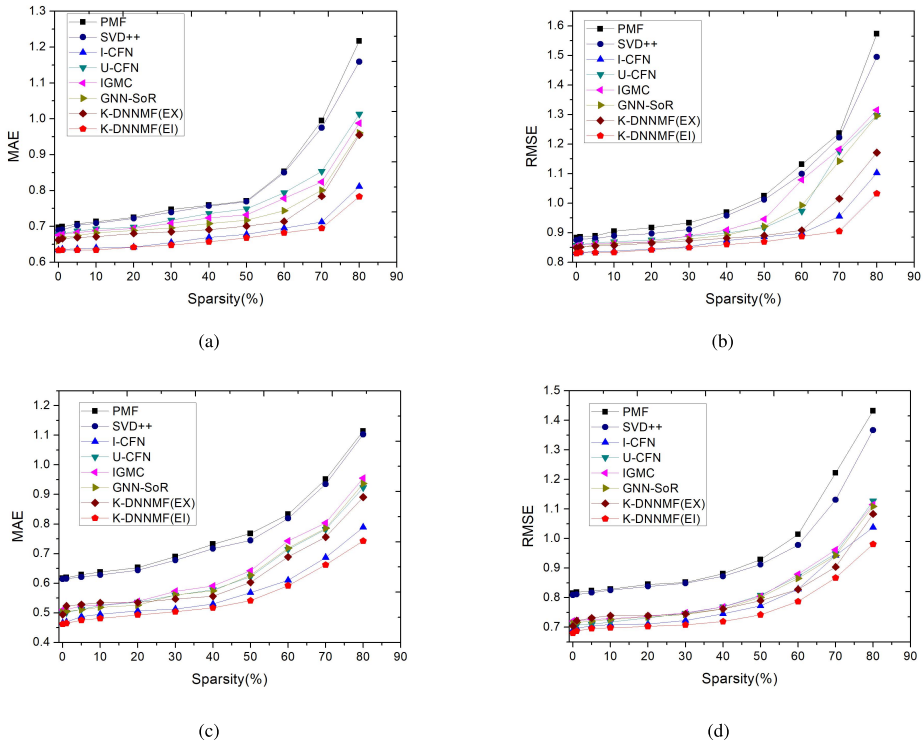


Fig. 5. Comparison experiments under different data sparsities. (a) Sparsity comparison experiment on MAE under MovieLens. (b) Sparsity comparison experiment on RMSE under MovieLens. (c) Sparsity comparison experiment on MAE under Douban. (d) Sparsity comparison experiment on RMSE under Douban.

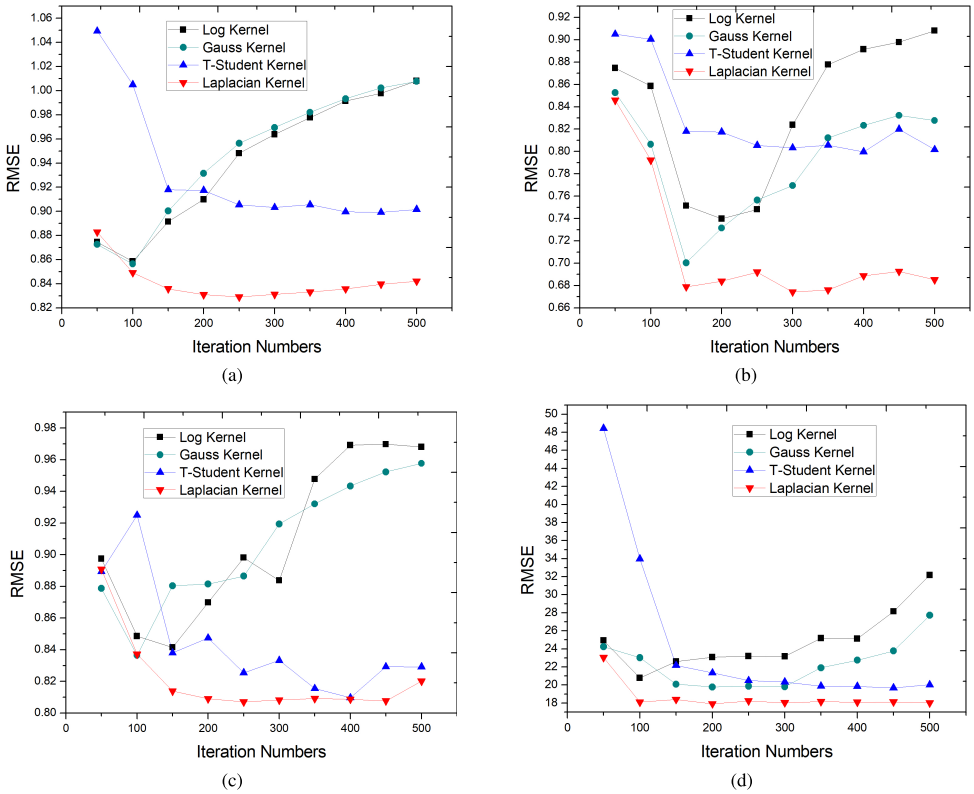
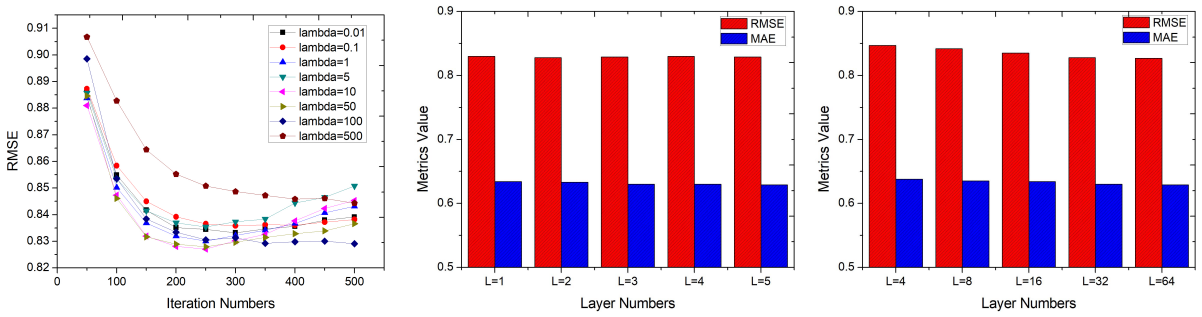


Fig. 6. Influence of kernel function on recommendation accuracy. (a) Testing different kernel functions on MovieLens. (b) Testing different kernel functions on Douban. (c) Testing different kernel functions on Flixster. (d) Testing different kernel functions on Yahoo Music.

items, so the network scale increases exponentially with the scale of users and items, and the optimization efficiency is low. In general, compared with the other four deep learning recommended models, the K-DNNMF(EI) method proposed in this article has faster convergence speed and relatively less fluctuation phenomenon.



(a) (b) (c)

Fig. 7. Influence of hyperparameters on recommendation accuracy. (a) Influence of regularized parameter on recommendation accuracy. (b) Influence of KernelNet layers on recommendation accuracy. (c) Influence of MLP layers on recommendation accuracy.

c) Performance comparison under data sparsity: Data sparsity is a serious challenge to the recommender system. Therefore, the performance test of the recommendation models in different sparse data environments can effectively distinguish the competitiveness of various methods. In this section, we adopt random data extraction to simulate the adaptability of various comparison models under various sparse conditions. We first randomly selected a certain proportion of rating data from the dataset and then deleted it. Then, we select 10% of the data from the dataset as the test data and the rest as training data. Because Flixster and Yahoo Music data are already sparse, many users do not have ratings after data random extraction, so data sparse experiments cannot be carried out. Therefore, we only simulated on MovieLens and Douban datasets. The experimental results in Fig. 5 show the variation trend of MAE and RMSE randomly extracted from the MovieLens and Douban datasets by 1%–80%. From the experimental results, it can be concluded that the K-DNNMF(EI) model proposed in this article has the optimal recommendation accuracy under different data sparseness conditions. As the data in the dataset become sparse, the recommendation accuracy gradually decreases. Especially after 50% data are extracted from the dataset, the performance of each comparative model decreases rapidly. In addition, the experimental results show that implicit information helps mitigate the impact of data sparsity on recommendation performance. The reasons are mainly as follows. First, the IGMC model searches for the nearest neighbor relationship by building subgraphs of users or items. When the data sparseness is enhanced, the data correlation is weakened rapidly, resulting in the rapid decline of IGMC recommendation performance. Second, I-CFN, U-CFN, and GNN-SoR methods integrate item feature or user profile to their models directly to alleviate the data sparsity. On the contrary, our K-DNNMF(EI) model uses an association rules algorithm to deeply mine the internal relations between users and items, so as to generate implicit information. This strengthens the connection between the user and the item and improves the recommendation performance. Moreover, we add an MLP network on top of the KernelNet, which can well fuse the explicit data and implicit data and enhance the robustness of the recommended method based on deep learning.

2) Sensitive Analysis for Hyperparameters:

a) Impact of kernel functions on recommendation performance: This section focuses on the effect of different types of kernel functions on the recommendation performance of our proposed model. We tested the influence of four different kernel functions on recommendation accuracy on four datasets, as shown in Fig. 6. The experimental results show that if the Laplacian kernel function is selected, the overall recommendation model has more stable and optimal recommendation accuracy. The log, Gauss, and T-student kernel functions fluctuate greatly. The experimental results demonstrate that the kernel function has a great influence on the model recommendation accuracy, so in practical applications, the recommender system can select the optimal kernel function in advance through training.

b) Optimization of network layers and regularized parameters: We mainly analyze the effect of the regularized parameter λ on the recommendation performance, as well as sets the number of layers in KernelNet and MLP networks in this section. Since these four datasets obtain the same conclusion, we only present the results of the MovieLens dataset. To simplify parameter setting, λ_1 and λ_2 in (4) are set to the same value, so in this experiment, λ is used for unified representation. The specific experimental results of normalized parameter λ are shown in Fig. 7(a). The result in Fig. 7(a) shows the influence of nine different values of regularization parameters from 0.01 to 500 on our proposed model. The experimental results show that too large or too small regularization parameters have a great influence on recommendation accuracy, and when λ is 10, it has the optimal recommendation accuracy. It can also be drawn from Fig. 7(a) that overfitting exists, but the overfitting phenomenon is good when λ is 100. Fig. 7(b) shows that different KernelNet layers have no obvious influence on the recommendation accuracy. Therefore, if the recommender system needs to improve training efficiency, we can use fewer network layers. Fig. 7(c) shows that the number of MLP network layers also has a certain impact on the recommendation performance. When the number of MLP network layers increases, the recommendation model has a small improvement, but it is not significant.

V. Conclusion

In this article, we presented a deep neural network matrix factorization recommendation model, namely, K-DNNMF. First, the rating matrix and implicit information mined by association rules are processed by KernelNet. Second, we propose to fuse the explicit data and implicit data through the MLP

network instead of simple linear weighted sum. Finally, the rating prediction is completed by a hidden layer. Extensive experiments on four public datasets show that the K-DNNMF model proposed in this article is very promising compared with several well-known recommendation methods for deep learning. From the experimental results, it can be concluded that the latent factor vectors of users and items can be projected to higher dimensions through kernelized networks so that the nonlinear interaction between users and items can be well simulated in higher dimensions. Moreover, adding implicit information can effectively improve the recommendation accuracy, especially in the case of sparse data. The MLP network is a promising way to combine explicit data with implicit information.

We intend to explore several directions in future work, including employing the ResNet to deal with the overfitting problem in deep learning networks. Furthermore, since the interaction between the user and the item is temporal, we are going to try some DRL approaches to solve the temporal recommendation problem.

REFERENCES

[1] X. Luo, M. Zhou, S. Li, Z. You, Y. Xia, and Q. Zhu, "A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 3, pp. 579–592, May 2015.

[2] C.-C. Hsu, and M.-Y. Yeh, "A general framework for implicit and explicit social recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 12, pp. 2228–2241, Mar. 2018.

[3] C. Wu, and M. Yan, "Session-aware information embedding for e-commerce product recommendation," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 2379–2382.

[4] X. Zheng, Y. Luo, L. Sun, J. Zhang, and F. Chen, "A tourism destination recommender system using users' sentiment and temporal dynamics," *J. Intell. Inf. Syst.*, vol. 51, no. 3, pp. 557–578, 2018.

[5] L. Wu, P. Sun, Y. Fu, R. Hong, X. Wang, and M. Wang, "A neural influence diffusion model for social recommendation," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2019, pp. 235–244.

[6] J. Wu, X. Li, F. Chiclana, and R. Yager, "An attitudinal trust recommendation mechanism to balance consensus and harmony in group decision making," *IEEE Trans. Fuzzy Syst.*, vol. 27, no. 11, pp. 2163–2175, Jan. 2019.

[7] R. Ma, X. Qiu, Q. Zhang, X. Hu, Y.-G. Jiang, and X. Huang, "Co-attention memory network for multimodal microblog's hashtag recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 2, pp. 388–400, Aug. 2021.

[8] C. Liu, J. Cao, and S. Feng, "Leveraging kernel-incorporated matrix factorization for app recommendation," *ACM Trans. Knowl. Discovery Data*, vol. 13, no. 3, pp. 1–27, 2019.

[9] L. Boratto, S. Carta, G. Fenu, and R. Saia, "Semantics-aware content-based recommender systems: Design and architecture guidelines," *Neurocomputing*, vol. 254, pp. 79–85, Sep. 2017.

[10] D. Wang, Y. Liang, D. Xu, X. Feng, and R. Guan, "A content-based recommender system for computer science publications," *Knowl.-Based Syst.*, vol. 157, pp. 1–9, Oct. 2018.

[11] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua, "Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention," in *Proc. 40th Int. ACM SIGIR Conf. Res. Development Inf. Retr.*, 2017, pp. 335–344,

[12] L. Baltrunas, B. Ludwig, and F. Ricci, "Matrix factorization techniques for context aware recommendation," in *Proc. 5th ACM Conf. Recommender Syst.*, 2011, pp. 301–304.

[13] K. Tsukuda, and M. Goto, "DualDiv: Diversifying items and explanation styles in explainable hybrid recommendation," in *Proc. 13th ACM Conf. Recommender Syst.*, 2019, pp. 398–402.

[14] X. Zheng, Y. Luo, L. Sun, X. Ding, and J. Zhang, "A novel social network hybrid recommender system based on hypergraph topologic structure," *World Wide Web*, vol. 21, no. 4, pp. 985–1013, 2018.

[15] M. Jamali, and M. Ester, "A matrix factorization technique with trust propagation for recommendation in social networks," in *Proc. 4th ACM Conf. Recommender Syst.*, 2010, pp. 135–142.

[16] R. Forsati, M. Mahdavi, M. Shamsfard, and M. Sarwat, "Matrix factorization with explicit trust and distrust side information for improved social recommendation," *ACM Trans. Inf. Syst.*, vol. 32, no. 4, pp. 1–38, 2014.

[17] Y. Xiao, G. Wang, C.-H. Hsu, and H. Wang, "A time-sensitive personalized recommendation method based on probabilistic matrix factorization technique," *Soft Comput.*, vol. 22, no. 20, pp. 6785–6796, 2018.

[18] X. Xu, and D. Yuan, "A novel matrix factorization recommendation algorithm fusing social trust and behaviors in micro-blogs," in *Proc. IEEE 2nd Int. Conf. Cloud Comput. Big Data Anal.*, Apr. 2017, pp. 283–287.

[19] D. F. Gurini, F. Gasparetti, A. Micarelli, and G. Sansonetti, "Temporal people-to-people recommendation on social networks with sentiment-based matrix factorization," *Future Gener. Comput. Syst.*, vol. 78, pp. 430–439, Jan. 2018.

[20] X. Ren, M. Song, E. Haihong, and J. Song, "Context-aware probabilistic matrix factorization modeling for point-of-interest recommendation," *Neurocomputing*, vol. 241, pp. 38–55, Jun. 2017.

[21] D. Lian, K. Zheng, Y. Ge, L. Cao, E. Chen, and X. Xie, "GeoMF++ scalable location recommendation via joint geographical modeling and matrix factorization," *ACM Trans. Inf. Syst.*, vol. 36, no. 3, pp. 1–29, 2018.

[22] Z. Xu, L. Chen, Y. Dai, and G. Chen, "A dynamic topic model and matrix factorization-based travel recommendation method exploiting ubiquitous data," *IEEE Trans. Multimedia*, vol. 19, no. 8, pp. 1933–1945, Mar. 2017.

[23] L. Wan, F. Xia, X. Kong, C.-H. Hsu, R. Huang and J. Ma, "Deep matrix factorization for trust-aware recommendation in social networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 1, pp. 511–528, Dec. 2021.

[24] X. He, J. Tang, X. Du, R. Hong, T. Ren and T.-S. Chua, "Fast matrix factorization with nonuniform weights on missing data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 8, pp. 2791–2804, Jan. 2020.

[25] J. Han *et al.*, "Adaptive deep modeling of users and items using side information for recommendation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 3, pp. 737–748, Jun. 2019.

[26] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–38, 2019.

[27] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173-182.

[28] Y. Liu, S. Wang, M. S. Khan, and J. He, "A novel deep hybrid recommender system based on auto-encoder with neural collaborative filtering," *Big Data Mining Anal.*, vol. 1, no. 3, pp. 211–221, 2018.

[29] M. Zhang, and Y. Chen, "Inductive matrix completion based on graph neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–14.

[30] S. Deng, L. Huang, G. Xu, X. Wu, and Z. Wu, "On deep learning for trust-aware recommendations in social networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 5, pp. 1164–1177, Feb. 2016.

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

[32] F. Strub, R. Gaudel, and J. Mary, "Hybrid recommender system based on autoencoders," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, 2016, pp. 11–16.

[33] F. Zhuang, Z. Zhang, M. Qian, C. Shi, X. Xie, and Q. He, "Representation learning via dual-autoencoder for recommendation," *Neural Netw.*, vol. 90, pp. 83-89, Jun. 2017.

[34] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.

[35] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems," in *Proc. IJCAI*, vol. 17, 2017, pp. 3203–3209.

[36] W.-T. Chu, and Y.-L. Tsai, "A hybrid recommendation system considering visual information for predicting favorite restaurants," *World Wide Web*, vol. 20, no. 6, pp. 1313–1331, 2017.

[37] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 974–983.

[38] S. Wu, W. Ren, C. Yu, G. Chen, D. Zhang, and J. Zhu, "Personal recommendation using deep recurrent neural networks in NetEase," in *Proc. IEEE Int. Conf. Data Eng. (ICDE)*, May 2016, pp. 1218–1229.
[39] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, "Personalizing session-based recommendations with hierarchical recurrent neural networks," *Proc. 11th ACM Conf. Recommender Syst.*, 2017, pp. 130–137.
[40] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 791–798.
[41] Z. Chen, W. Ma, W. Dai, W. Pan, and Z. Ming, "Conditional restricted Boltzmann machine for item recommendation," *Neurocomputing*, vol. 385, pp. 269–277, Apr. 2020.
[42] Z. Guo and H. Wang, "A deep graph neural network-based mechanism for social recommendations," *IEEE Trans. Ind. Inform.*, vol. 17, no. 4, pp. 2776–2783, Apr. 2021.
[43] H. Xu, C. Huang, Y. Xu, L. Xia, H. Xing and D. Yin, "Global context enhanced social recommendation with hierarchical graph neural networks," *IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2020, pp. 701–710.
[44] S.-Y. Chen, Y. Yu, Q. Da, J. Tan, H.-K. Huang, and H.-H. Tang, "Stabilizing reinforcement learning in dynamic environment with application to online recommendation," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 1187–1196.
[45] G. Zheng *et al.*, "DRN: A deep reinforcement learning framework for news recommendation," in *Proc. World Wide Web Conf.*, 2018, pp. 167–176.
[46] Z. Zhao *et al.*, "Social-aware movie recommendation via multimodal network learning," *IEEE Trans. Multimedia*, vol. 20, no. 2, pp. 430–440, Aug. 2017.
[47] Z. Huang, X. Xu, H. Zhu, and M. Zhou, "An efficient group recommendation model with multiattention-based neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 4461–4474, Jan. 2020.
[48] Z. Li, X. Fang, and O. R. L. Sheng, "A survey of link recommendation for social networks: Methods, theoretical foundations, and future research directions," *ACM Trans. Manage. Inf. Syst.*, vol. 9, no. 1, pp. 1–26, 2017.
[49] A. Mnih, and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1257–1264.
[50] Y. He, C. Wang, and C. Jiang, "Correlated matrix factorization for recommendation with implicit feedback," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 3, pp. 451–464, May 2018.
[51] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008,pp. 426–434,
[52] T. Tran, K. Lee, Y. Liao, and D. Lee, "Regularizing matrix factorization with user and item embeddings for recommendation," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 687–696.
[53] L. Shi, W. X. Zhao, and Y.-D. Shen, "Local representative-based matrix factorization for cold-start recommendation," *ACM Trans. Inf. Syst.*, vol. 36, no. 2, pp. 1–28, 2017.
[54] G. Guo, J. Zhang, and N. Yorke-Smith, "A novel recommendation model regularized with user trust and item ratings," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1607–1620, Feb. 2016.
[55] C. A. Micchelli, and M. Pontil, "Learning the kernel function via regularization," *J. Mach. Learn. Res.*, vol. 6, pp. 1099–1125, Jul. 2005.

Xiaoyao Zheng (Member, IEEE) received the M.S. degree in computer science from the School of Computer Science, Hefei University of Technology, Hefei, China, in 2005, and the Ph.D. degree in human geography from Anhui Normal University, Wuhu, China, in 2018.

Since 2021, he has been a Professor with the School of Computer and Information, Anhui Normal University. His research interests include data mining, information security, and big data analysis.

Dr. Zheng is a member of China Computer Federation (CCF).

Zhen Ni (Senior Member, IEEE) is currently an Associate Professor with the Department of Electrical Engineering and Computer Science (EECS), Florida Atlantic University (FAU), Boca Raton, FL, USA. Before that, he was an Assistant Professor with the Department of Electrical Engineering and Computer Science (EECS), South Dakota State University (SDSU), Brookings, SD, USA, from 2015 to 2019. He has authored and coauthored more than 100 peer-reviewed research papers mostly on IEEE TRANSACTIONS journals and major conferences. His research interests mainly include artificial and computational intelligence, and machine learning.

Dr. Ni received the prestigious NSF CAREER Award in 2021. He has been an Associate Editor of IEEE INTERNET OF THINGS JOURNAL since 2021, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS since 2019, and *IEEE Computational Intelligence Magazine* since 2018.

Xiangnan Zhong (Member, IEEE) is currently an Assistant Professor with the Department of Electrical Engineering and Computer Science, Florida Atlantic University (FAU), Boca Raton, FL, USA. Her research interests include computational intelligence, reinforcement learning, cyber-physical systems, networked control systems, neural networks, and optimal control.

Prof. Zhong received the National Science Foundation (NSF) Faculty Early Career Development (CAREER) Award in 2021 and the NSF CRII Award in 2019. She was a recipient of the International Neural Network Society (INNS) Aharon Katzir Young Investigator Award in 2021 and the INNS Doctoral Dissertation Award in 2019. She has been serving as the Chair for IEEE Computational Intelligence Society (CIS) Adaptive Dynamic Programming and Reinforcement Learning (ADPRL) Technical Committee since 2021 and an Associate Editor of IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (TNNLS) since 2021.

Yonglong Luo received the Ph.D. degree from the School of Computer Science and Technology, University of Science and Technology of China, Hefei, China, in 2005.

Since 2007, he has been a Professor with the School of Computer and Information, Anhui Normal University, Wuhu, China. He is currently a Ph.D. Supervisor with Anhui Normal University. He is also the Director of the Anhui Provincial Key Laboratory of Network and Information Security. His research interests include information security and spatial data processing.