IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 21 February 2024, accepted 3 March 2024, date of publication 18 March 2024, date of current version 1 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3378521

RESEARCH ARTICLE

Design and Implementation of Attention-Based CR System in the Context of Big Data

ZHEYI WANG[1], YANLI KUANG[2], AND XIN LYU[3]
[1]The University of Hong Kong, Hong Kong
[2]Imperial College London, SW7 2AZ London, U.K.
[3]Chongqing University of Posts and Telecommunications, Chongqing 400065, China
Corresponding author: Zheyi Wang (h1295125@connect.hku.hk)

ABSTRACT The recommendation accuracy of traditional product recommendation systems is insufficient. Therefore, an improved deep factor decomposition machine algorithm combining adaptive regularization and attention mechanisms is proposed, and big data components are integrated to enable the algorithm to support more data input types. The publicly available datasets Criteo and Avazu from the Kaggle competition were selected for testing experiments in the study. The experimental results are as follows. During the training phase, after the convergence of each model, the loss function value of the model designed in this study is 1.26, which is the lowest among all comparison models. Moreover, when the number of recommended products is 7, the overall recommendation effect of each model is the best. The area under the curve of the subject operation characteristic curve of the model designed in this study on the Criteo dataset is 0.809, which is significantly higher than all comparison models. It is believed that this model has higher recommendation accuracy and can be used in application scenarios that require higher recommendation quality.

INDEX TERMS Big data, attention mechanisms, commodity recommendations, neural networks, adaptive regularit.

I. INTRODUCTION

With the advancement of Internet technology and the popularity of Big Data (BD), e-commerce has seen remarkable growth, and Commodity Recommendation (CR) systems have become increasingly important [1]. As the foundation of modern e-commerce, CR systems may incorporate the user's prior behaviors and product qualities and deliver personalized recommendations for the user, improving user experience and buy conversion rate [2], [3]. Three basic paths have emerged since the introduction of recommender systems: content-based recommendation, collaborative filtering recommendation, and hybrid recommendation [4], [5]. Content-based recommendation is based on a thorough comprehension of the objects, and this approach views the recommendation problem as an information retrieval problem [6]. Collaborative filtering approaches, on the other hand, focus mostly on user behavioral data to create a relationship between users and items and produce recommendations based on the idea of ''things are grouped together, people are grouped together.'' Hybrid recommendation combines the aforementioned two strategies, attempting to maximize their respective advantages while minimizing their drawbacks and enhancing suggestion accuracy. However, most existing recommender systems do not make effective use of users' behavioral characteristics, and some users' short-term interests are not effectively exploited, while the stability of long-term interests and the dynamics of short-term interests are difficult to handle concurrently. Furthermore, many recommender systems overlook the user's attention change, i.e., people pay varied attention to different products. Recommender systems that neglect this frequently produce incorrect recommendation results. As a result, this research presents a CR system based on Attention Mechanisms (AMs), with the goal of improving the CR system's suggestion quality. AM is taken from human brain functioning; when confronted with a great amount of information, humans will only pay attention to a subset of it that is most relevant to the job. AM can be

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

used in recommender systems to parse the user's behavior and adaptively compute the user's interest weight for each product.

The first section of this research introduces the present status of development of recommender systems and their function in e-commerce. The second component is to create a CR model based on Deep Factorization Machine (DeepFM), which is also the research's novelty. The third stage is to run online CR experiments with the created model and compare the results to those estimated by other models. The fourth section involves analyzing the findings of the trials. The main contribution of this study is the development of a product recommendation method that outperforms traditional models in terms of recommendation quality. This method can be applied in the business field to enhance transaction completion rates and improve customer satisfaction with their shopping experience. In summary, this study starts with the research background, analyzes the purpose and necessity of carrying out the research, and then carries out the corresponding model optimization design according to the shortcomings of the current product recommendation system and gives the technical details of the optimization content. Finally, the ablation experiment and multi-model comparison experiment were designed to verify the performance of the improved model. The experiment involved the comparison of multiple objective and subjective evaluation indicators. It can be seen that the research under this topic has a good comprehensiveness.

II. RELATED WORKS

Recommender system is an important part of the business and search field, and the quality of its recommendation results can greatly affect the business income of Internet enterprises. Therefore, recommender systems have been studied by a large number of computer experts. Luo F proposed a personalized recommender system for learning the energy-saving appliance usage patterns of large-scale residential users and recommending appropriate appliance usage plans for them while taking their lifestyles into account. The system was based on collaborative filtering recommendation technology, which first classifies users into ''high-responsive users'' and ''low-responsive users'' according to their demand response level. For each low-responsive user, the system inferred their lifestyle from the usage profile of non-transferable appliances. It then identified users from the set of high-responsive users who have similar habits as the target user. The system made appliance usage recommendations to the target user based on the lifestyle similarity between the target user and each target User. Experimental results showed that the system's recommendation results in high user satisfaction [7]. Gwadabe and Liu found that on e-commerce websites, different interaction sequences may lead to the same recommendation results, so the authors proposed a new graph neural network model for learning interactions between sequential and non-sequential items. The model used a graph neural network to learn non-sequential interactions first and then sequential interactions in an end-to-end manner. The model outperformed other state-of-the-art models on both Yoochoose and Diginetica datasets, with a 10% and 11% improvement in recommendation accuracy over the best-performing comparison model [8]. Roozbahani et al. proposed an integrated model based on multilayer networks that can be personalized to recommend research collaborators. In this approach, collaborators are identified through community detection algorithms. The model can serve the purpose of preventing information loss in the network. Recommendation experiments were conducted using research data and the experimental results showed that the model has high accuracy in recommending research collaborators [9].

Zhang found that the uncertainty of implicit feedback data poses a great challenge to recommender systems, so the authors proposed a causal neuro-fuzzy inference algorithm for modeling missing data in implicit recommendations. Specifically, the authors analyzed the influencing factors of a user's exposure to an item from a causal perspective and used fuzzy set theory to represent the influencing factors. Based on the fuzzy representation of the influencing factors, the authors applied a causal neuro-fuzzy inference network to learn the weights of the fuzzy rules. Finally, joint learning was performed by this algorithm and matrix decomposition in order to complete the construction of the recommendation model. Extensive experiments conducted on three real datasets proved the effectiveness and sophistication of the proposed algorithm [10]. Da'U A and other scholars proposed a recommender system model that utilizes neural network's attention technique to learn user/item representations by jointly considering fine-grained semantic information of user-item pairs. The model learnt heterogeneous user/item representations using review-based and interaction-based features. The authors' team then went on to design a co-attention network to capture the semantic information most relevant to user-item pairs. To capture user-item latents more comprehensively, the authors further integrated interaction-based features with comment-based latents through a shared hidden layer. Finally, the attention factoriser was applied to the shared hidden layer of the integrated user-item features to achieve the output of the prediction results. The authors conducted a series of experiments using real-world datasets and the results demonstrated that the proposed method outperforms the baseline method in terms of rating prediction and ranking performance [11]. Sturluson A designed a recommendation system based on covalent organic frameworks (COF), and experimentally demonstrated that this COF recommendation system was able to reasonably rank COF according to most of the adsorption performance metrics, but the predictive effectiveness of the system decreased dramatically when the percentage of missing entries exceeded 60% [12].

In summary, although a lot of previous research has been carried out to improve the recommendation quality of recommender systems in various domains, it is quite rare to design a recommender system by combining AM and deep

learning methods, and the nonlinear feature processing capability of DNN and the selective attention to data by AM are crucial for the recommendation task, so here an improved CR system is designed to try to improve the quality of recommendation results. In addition, this study also contributes to the knowledge system in the field of recommendation systems, because it proves that adaptive regularity, attention mechanism and big data components can be applied to the DeepFM model at the same time, and this improvement is indeed conducive to improving the performance of the model.

III. CR MODEL BASED ON IMPROVED DeepFM NETWORKS

The traditional linear recommendation model has poor ability to mine deep recommendation features, and therefore the recommendation effect is hardly satisfactory, so now based on DeepFM, a mainstream algorithm that is commonly used in the industry and has good effect, the structure of the DeepFM algorithm is improved, the regularization method is improved, and the AM is integrated to design the hybrid DeepFM algorithm. The CR model is then constructed in the study using the hybrid DeepFM algorithm and a series of BD components.

A. DeepFM NETWORK DESIGN FOR HYBRID AM

DeepFM is a standard parallel fusion network architecture that consists of a factoriser and a Deep Neural Network (DNN) that share embedding layer vectors of users and items [13], [14], [15]. The factor decomposer part is responsible for handling the first- and second-order automatic combination of features, which allows the model to have a high memory by learning low-order features. Whereas, the DNN part is responsible for handling the higher order feature extraction to give the model a high generalization ability [16]. The output y' of the whole model is obtained as per equation (1).

$$y' = sigmod\left(y_{FM} + y_{DNN}\right) \tag{1}$$

where $y' \in (0, 1)$ and y_{FM} are the outputs of the FM part and y_{DNN} is the output of the DNN board. The FM model is responsible for dealing with both data sparsity and complexity in the whole network architecture. And FM enables each feature to learn a hidden vector by using the inner product of two vectors instead of a single weight coefficient, and the combined weights between the features are also the hidden vector inner product of the features [17]. The FM module can play a role in calculating the correlation of two features even if they have no interaction data between them. For example, the FM correlation $FM(V, x)$ of the hidden vector V, input data vector x is calculated according to equation (2).

$$FM(V, x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle V_i \cdot V_j \rangle x_i x_j \tag{2}$$

Equation (2) Each feature i has two parameters, i.e., the importance parameter w_i and the hidden vector V_i. where w_i is mainly used for evaluating the first-order importance of the features, while V_i is used for combining features among features, which is used in second-order computation of FM and higher-order feature combinations of DNNs [18]. The structure of the FM module is shown in Fig. 1, where the Field represents the field of features with the same attributes, which form the basis of DeepFM feature representation.

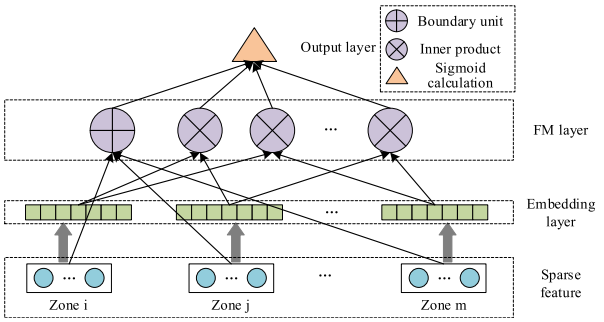

FIGURE 1. Structure of FM Module in DeepFM.

The DNN part is a fully connected feed-forward neural network for learning higher-order feature combinations between users and items, which is shown in Fig. 2.

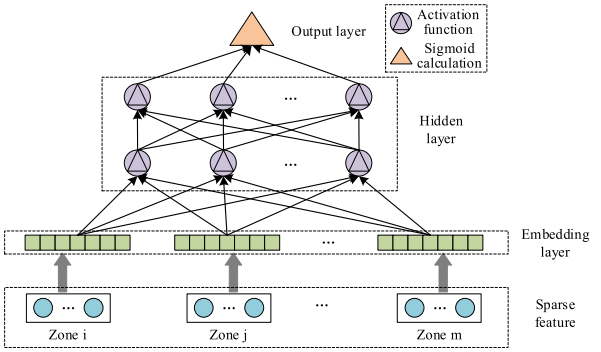

FIGURE 2. DNN network structure.

The original input is a high-dimensional sparse one-hot code, which is converted into a low-dimensional dense vector by the embedding layer for network training. The output $a^{(0)}$ of the embedding layer can be described as in equation (3) [12].

$$a^{(0)} = [e_1, e_2, \ldots, e_m] \tag{3}$$

m in equation (3) denotes the number of feature domains and e_i is the embedding vector of the i th feature domain. By inputting $a^{(0)}$ into the DNN network, the forward propagation process of the DNN network can be described in equation (4):

$$a^{(l+1)} = \sigma\left(W^{(l)} a^{(l)} + b^{(l)}\right) \tag{4}$$

In equation (4) l is the current number of layers, σ is the activation function, $a^{(l+1)}$ is the output of the first layer, $W^{(l)}$

is the model weights and $b^{(l)}$ is the bias term. Then the final output of the DNN module can be described in equation (5).

$$y_{DNN} = \sigma\left(W^{|H|+1}a^{|H|} + b^{|H|+1}\right) \tag{5}$$

In equation (5), A is the number of layers of the implicit layer. This leads to the overall framework structure of DeepFM, as shown in Fig. 3.

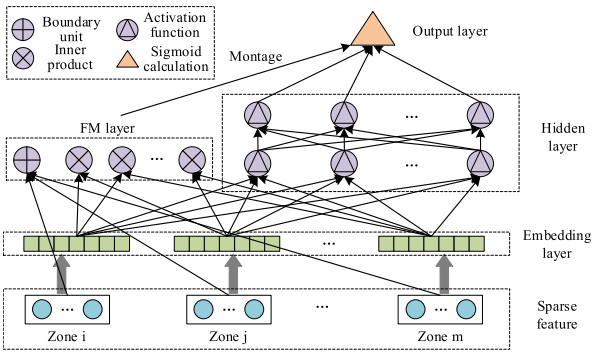

FIGURE 3. DeepFM overall framework.

A good recommendation model requires both memory and generalization capabilities, and the DeepFM model combines both. It uses the FM module to extract shallow features to enhance the memory ability, and the DNN module to extract deep features to enhance the generalization ability. It can be seen that DeepFM is an improvement of the Wide&Deep model, which is unable to combine features automatically.

Consumer's past behavior plays an extremely important role in click prediction. For user interest recognition, the DeepFM model concatenates all embedding vectors in the user's historical behavioral feature strings to form a stable representation vector. For a specific user, the model ensures the consistency and invariance of the user's interest representation due to the application of average pooling, so that the representation vector does not change regardless of the expected term. In other words, it lacks the ability to express interests in a way that reveals the differential impact of historical behavior on users' interests, and therefore consumes a great deal of valuable information. For example, a male user might purchase leather jackets and trainers, mice and headphones, or even dresses for his or her spouse. In reality, when a male user buys a keyboard, he does not need to consider dresses as preference characteristics. The process of purchasing a keyboard is far more influenced by mice and headphones than skirts. At this point, instead of having a positive effect on the recommendation outcome, the skirt data feature attenuates the positive influence of the mouse and headphone features, becoming noise in the recommender system and leading to reduced model performance. In this example, the entire purchase process is: the expected product keyboard has identified that this consumer has purchased mouse and headphones before through a soft search of this user's purchase behavior, thus triggering his related interest. In short, the historical behavior related to the expected item contributes to the user's click.

Considering that AM can enhance the learning ability of the model to focus on the content and reduce the influence of irrelevant features, this study designed an added AM structure - i.e., the attention capsule - for the correlation between consumer user behavior and expected items. By introducing the attention capsule into the DeepFM model, the hybrid model is able to generate a variable, dynamic representation vector to express user interests with limited representation vector dimensions. The structure of the attention capsule is shown in Fig. 4. The role of the attention capsule is to adaptively change the input embedding vectors of the DNN network during historical behaviors by exploiting the different activation levels of the expected items. Each historical behavior of the user is weighted with the expected item to adaptively compute the user interest representation vector of the expected item. The structure of DeepFM network with hybrid attention capsule is shown in Fig. 4. The data labeled as 'User Embedding Data' and 'Embedding data for candidate items' on the far left of Figure 4 are obtained by cleaning and transforming the dataset.

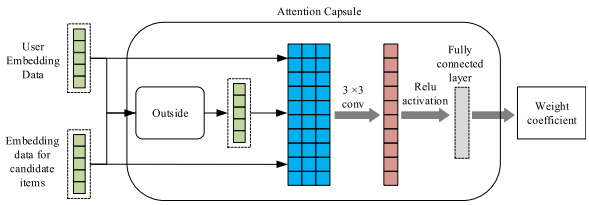

FIGURE 4. Structure diagram of attention capsules.

The attention capsule uses the output data to adjust the attention bias of the other parts of the recommender system, while the attention capsule internally forms the final output by performing multiple convolution, mapping, and fully-connected changes to the input data. Specifically, the output of the attention capsule is calculated according to equation (6).

$$\begin{aligned} v_U(A) &= f(v_A, e_1, e_2, \ldots, e_m) \\ &= \sum_{j=1}^{m} g\left(e_j, v_A\right) e_j \\ &= \sum_{j=1}^{m} weight_j e_j \end{aligned} \tag{6}$$

In equation (6), v_A represents the embedding vector, $g()$ is the feed-forward network, and weight $weight_j$ represents the activation weights. The input to the attention capsule is the embedding vectors of the historical behavior and the candidate products, here the outer product of the two is chosen, and the three are spliced and combined and convolved using a convolution kernel. The output is then connected to a fully connected layer to get the magnitude of the weight values. The attention capsule model used in this study sheds the

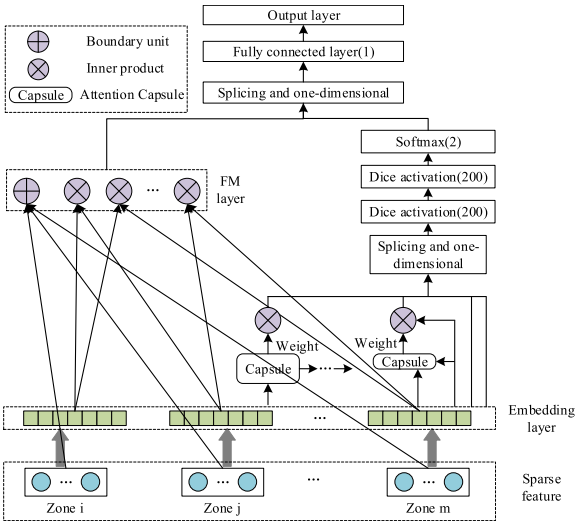

FIGURE 5. Network structure of DeepFM model for mixed attention capsules.

Softmax layer in the traditional attention framework, which means that the obtained sum of weights is not equal to 1. By removing the Softmax layer, the model can maintain the activation level of the items. This means that the correlation between the items and the historical behaviors increases as the sum of weights increases, improving the model's ability to represent interest. The DeepFM model with added attention capsule structure is shown in Fig. 5. As shown in Fig. 5, the hybrid model adds an attention capsule after the embedding layer, so that the user's historical clicking behaviors and the candidate product process need to go through a weighting calculation to get the weight of each clicking behavior with respect to the candidate product. This approach allows the candidate products as well as the products associated in the historical behaviors to be highlighted when forming the representation vectors. Thus, the dynamic representation vectors are adaptively generated to enhance the interest expression capability of the model in a limited number of dimensions. As shown in Fig. 5, the main steps of the hybrid model are as follows: To begin, input the user's historical behaviors into the embedding layer to obtain embedding vectors for various features. Then, provide these vectors to the FM layer to enable the model to learn shallow features. In the next step, the embedding vectors of the historical behaviors are inputted into the attention capsule for the weight calculation with the embedding vectors of the candidate items, and then multiply the computed weight with the original embedding vectors, and input the result into the DNN layer. Finally, the outputs of the FM layer and the DNN layer are spliced and processed in one-dimension, and then inputted into a full-connectivity layer, so as to get the final prediction results. The term 'spliced' refers to the process of combining multiple vectors or matrices into a larger matrix. 'Processing in one dimension' involves concatenating the beginning and end of each row of the matrix to create a one-dimensional vector.

The loss function for the training process uses the log-likelihood loss function, which is calculated in equation (7).

$$L(x, y) = -\sum_{(x,y)\in S} \begin{matrix} ylogp(x) + \\ (1-y)\,log\,(1-p(x)) \end{matrix} \Big/ N \quad (7)$$

In equation (7), y is the actual label of the sample and $p(x)$ is the predicted click probability of input x.

B. DESIGN OF CR MODELS WITH HYBRID ADAPTIVE COLLABORATIVE REGULARITY AND DeepFM

When training a deep learning model, ideally, the model should perform with a small training error in the training set while still maintaining a low error in the unknown test set. Even though, the model should have good generalization performance in the unknown dataset. This generalization performance demonstrates the model's adaptability and transferability. As of now, the industry standard is to describe the model's generalizability using both under-fitting and over-fitting measures. These two states reflect the condition of the model during the training and testing phases. A model may show under-fitting in the early stage of training, but with too much training, the model gradually faces over-fitting, when the generalization error increases accordingly, as shown in Fig. 6. Since the hybrid model contains a CNN structure, the over-fitting problem may also arise, and for this reason, an adaptive regularization method is now devised and this method is fused into the hybrid model.

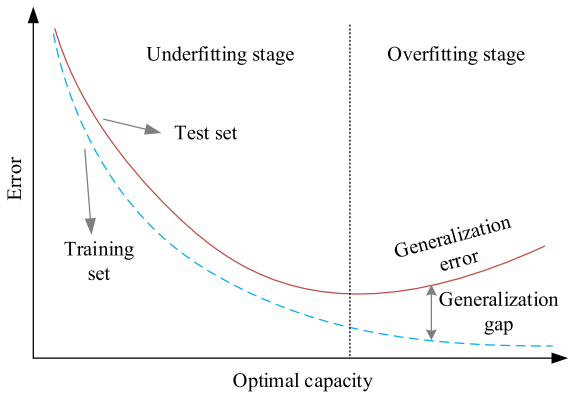

FIGURE 6. Display of over-fitting and under-fitting phenomena.

In an active recommender system, the amount of data may reach billions of levels. Click through rate (CTR) prediction of advertisements in such a huge data system often encounters the problem of sparse and high-dimensional input data. In such a BD environment, it is impractical to use traditional normalization methods directly. In the case of Stochastic Gradient Descent (SGD), for example, if no normalization is performed, only the parameters corresponding to the features that are not zero in the input features need to be updated during training. However, if L2 regularization is introduced, the L2 paradigms of all parameters need to be computed during training, which can greatly increase the burden of

training and reduce the efficiency of the model. Since the L1 norm is sparser than the L2 norm, while the L2 norm does not occur with most of the parameters being zero. Therefore, an adaptive co-regularization equation based on L1 and L2 norms is now designed, i.e., only the L1 and L2 co-paradigms of the parameters corresponding to the non-zero input characteristics are computed and weighted as shown in equation (8).

$$\begin{aligned} &L_{1,2}(W) \\ &= \rho \|W\|_1 + (1-\rho)\|W\|_2^2 \\ &= \sum\nolimits_{(x,y)\in S} \sum_{j=1}^{K} \frac{I\left(x_j \neq 0\right)}{n_j} \left(\rho \left\|w_j\right\|_1 + (1-\rho)\|W\|_2^2\right) \end{aligned} \tag{8}$$

In equation (8) W represents the whole input dictionary, while S represents the size of the training set, K is the feature dimension; and x is the input to the network. $y \in \{0, 1\}$ represents the click labels (0 means not clicked, 1 means clicked), w_j is the j th vector, n_j is the number of occurrences of the j th feature in the whole sample, and ρ is the canonicalised assignment weight coefficient. The judgement function in the improved regularization method is shown in equation (9).

$$I = \begin{cases} 1, & \exists (x, y) \in B, s.t. \left[x_j\right] \neq 0 \\ 0, & other \end{cases} \tag{9}$$

I is the indicator function that discriminates whether e_1 1e i th sample contains feature j. The optimization method iai sed in this research is Mini Batch Gradient Descent MBGD). The total number of samples can be decomposed into multiple small batch samples, so equation (8) can be converted into equation (10). According to the descriptions in equations (8) and (9), after setting ρ, the $L_{1,2}(W)$ loss function can automatically calculate and adjust the coefficients of L1 and L2 norm terms based on whether each sample contains specified features and the click label status of the samples, thereby obtaining an adaptive regularization effect.

$$L_{1,2}(W) = \sum_{j=1}^{K} \sum_{m=1}^{B} \frac{I}{n_j} \left(\rho \left\|w_j\right\|_1 + (1-\rho)\left\|w_j\right\|_2^2\right) \tag{10}$$

B and B_m in equation (10) represent the split small batch and the m th small batch, respectively. Equation (10) is the regular term with adaptive synergistic capability designed in this time, and the derivation using this equation leads to the updating equation of the parameters, as shown in equation (11).

$$w_j \leftarrow w_j - \eta \left[\frac{1}{|B_m|} \sum_{(x,y)\in B_m} \frac{\frac{\partial L(p(x),y)}{\partial w_j} + \frac{\lambda \rho I w_j}{n_j}}{+\frac{\lambda(1-\rho) I \operatorname{sgn}\left(w_j\right)}{n_j}} \right] \tag{11}$$

The $\operatorname{sgn}\left(w_j\right)$ -function in equation (11) is calculated in equation (12).

$$\operatorname{sgn}\left(w_j\right) = \begin{cases} 1, & w_j > 0 \\ 0, & w_j = 0 \\ -1, & w_j < 0 \end{cases} \tag{12}$$

Due to the introduction of the adaptive co-regularization equation, the loss function needs to be adjusted accordingly. Because the adaptive co-regularization equation is improved based on the L1 and L2 regularization, when splicing the loss function, it is necessary to multiply the regularization equation by 1/2 in order to facilitate the subsequent training calculations. Therefore, equation (7), which describes the loss function, becomes equation (13).

$$\begin{aligned} L &= L + \lambda \cdot f(w) \\ &= -\sum\nolimits_{(x,y)\in S} \left[\begin{matrix} ylogp(x) + \\ (1-y)\, log\,(1-p(x)) \end{matrix} \Big/ N \right] \\ &\quad + \frac{\lambda}{2N} \sum_{j=1}^{K} \sum_{m=1}^{B} \frac{2I}{n_j} \left(\rho \left\|w_j\right\|_1 + (1-\rho)\left\|w_j\right\|_2^2\right) \end{aligned} \tag{13}$$

In equation (13), since the output interval of the loss function is $[0, +\infty)$, the smaller its value, the better the classification effect of the model. For each sample, the model's accuracy increases as the predicted value approaches the sample label, resulting in a loss function value closer to 0. The PReLu function $f(x)$ can be converted to equation (14) way.

$$f(x) = p(x)x + (1 - p(x))ax \tag{14}$$

x in equation (14) is the one-dimensional input data representing $f(x)$, a probability function in the interval [0, 1] used to represent the two classification cases, and a very small self-learning parameter that updates itself during the training process. The PReLu function takes 0 as the turning point. However, in this design of the hybrid model, the inputs are different for each layer, so it is clearly inappropriate to use a fixed activation function to map the inputs of the neurons to the outputs. The output distribution of the model keeps changing when the same data distribution is input, resulting in significant and varying changes at the end of each training process. This phenomenon cannot ensure consistent training results. To address this problem, the Dice function is now used, which can adaptively adjust the turning point according to the characteristics of the input, and the calculation process is shown in equation (15).

$$p(x) = \frac{1}{1 + e^{\frac{x - E[x]}{(Var[x]+\varepsilon)^{1/2}}}} \tag{15}$$

$E[x]$ and $Var[x]$ in equation (15) represent the mean and variance of the input information of each batch of samples, respectively, both of which will be updated gradually during the training process, and ε is a constant term, which is taken as 10^{-8} according to the past experience. After completing the construction of the recommendation algorithm, it is necessary to build the overall CR model, taking into account the BD characteristics of the input data, as well as the type of processing tasks, the designed recommendation model framework is shown in Fig. 7. From this figure, in this system, the recommendation framework is mainly composed of two parts: offline recommendation and real-time recommendation. Offline recommendation consists of

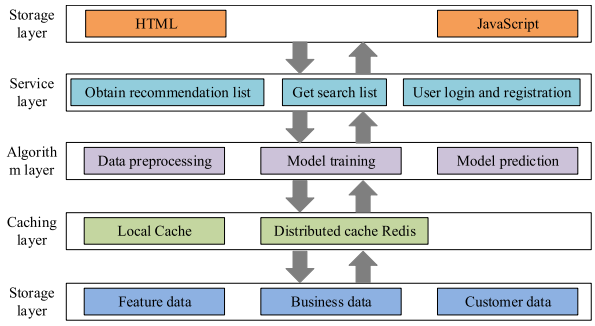

FIGURE 7. Structure of CR model for hybrid improved DeepFM algorithm.

offline statistics service and offline recommendation service. The offline statistics service counts the hot-selling and highly rated products. The offline recommendation service generates a recommendation list based on the user's past behaviors, which is fixed for a short period of time. Additionally, the offline recommendation service generates a list of similar products. When a user views the details of a product, other products similar to that product are displayed. The real-time recommendation service, on the other hand, computes a real-time recommendation list for a user by receiving a stream of user ratings from Kafka, storing the filtered ratings data in Redis, and then submitting the user's ratings queue to the real-time recommendation algorithm. This real-time recommendation list will be stored in MongoDB for business code query and display.

The recommender system of this research is mainly built based on the Spark framework, with the front-end using the Vue framework for page rendering, while the back-end is constructed using the Spring framework. Spark is a fast BD processing framework based on a memory structure and is more compatible than traditional data processing frameworks, as well as being compatible with most of the major databases, which was the reason for choosing the framework to build the recommender model. Support for CR model design based on improved DeepFM network is completed.

IV. EFFECTIVENESS TEST OF CR MODEL APPLICATION BASED ON IMPROVED DeepFM NETWORKS

In this study, a CR experiment is designed and conducted to test the performance of the designed CR model application. The part of designing the experiment mainly contains choosing the experimental dataset selection, the selection of comparison models required for the experiment, the design of the deployment environment for each model, and the selection of evaluation metrics. After designing the experimental program, the optimal deep network structure of the model with DNN structure is first determined experimentally, and then the overall structure of the corresponding model is determined with the optimal DNN structure, and subsequent performance validation experiments are carried out.

A. ANALYSIS OF TRAINING PHASES AND OPTIMAL PROGRAMME SELECTION

The development side environment of the model is as follows: Win10 Professional, 8GB of running memory, i5-7800 processor, and Python version 3.7 programming language is used to experiment with each recommendation model. The public datasets Criteo and Avazu used in the Kaggle competition were selected for the validation experiments, they are also the data source of this paper. Since the data sets selected in this study have been successfully applied to professional open algorithm competitions, and no data errors have been found, they can be considered to have good reliability.DeepCrossing algorithm, DeepFM algorithm, Attentional Factorization Machines (AFM), and CatBoost, an advanced machine learning algorithm, were chosen to construct the comparison models. The parameters of each comparison algorithm are obtained by debugging several times and taking the optimal value. In the experiments, various data sets are divided into training and testing sets in the ratio of 7:3. Various recommendation models were deployed on a 64-bit centos 7.0 server with server memory set to 8GB and hard drive size of 100GB.Area Under Curve (AUC) of receiver operating characteristic curve, average of AUC of each user weighted by number of times the product was displayed (Group AUC, GAUC), and value of loss function were selected for the study, Accuracy, Precision, and Recall as model evaluation indexes.

In this study, 25 experts in the field of recommendation system at home and abroad were invited to score the technical rationality of the improved recommendation model designed in this study on a 10-point scale, and the experts were told that scores above 6 points and above 8 points represented average rationality and good rationality respectively. After evaluation, it was found that the technical rationality score of all the experts in the design model of this study was higher than 8 points, and the average score was 8.62 points. The subjective evaluation results proved that the research technology had good rationality. In addition, since the logic of the core components of the improved recommendation model, such as DeepFM, adaptive regularization and attention module, can be implemented by mainstream programming tools including Python and Java, and the joint relationship between the components has been clearly described, the designed improved recommendation model has good operability and reproducibility [19].

Firstly, the effect of different number of hidden layers on the model performance is analyzed, and the statistical results are shown in Fig. 8. Since there is no hidden layer in the two models AFM and CatBoost, they are not involved in this experiment. In order to improve the accuracy of the experimental results, each scenario is repeated five times here. Figure 8 shows the recommended models for different hidden layer numbers and performance evaluation indexes. The grey dotted line represents the fitting curve between the number of hidden layers and the evaluation index value of each recommended model. Different icons are used to represent different

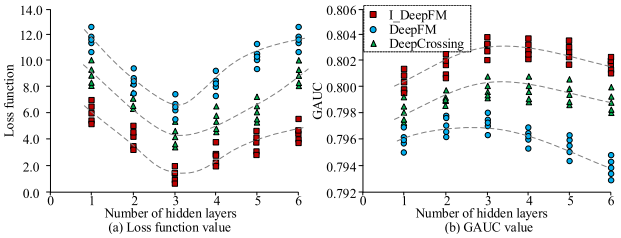

FIGURE 8. The Impact of the Number of Hidden Layers on Model Performance

models. Analysis of Figure 8 shows that when the number of hidden layers is 3, DeepCrossing, DeepFM, DeepFM model loss function mean and GAUC mean are the overall minimum and maximum, respectively, specifically 4.13, 6.27, 1.59 and 0.799, 0.797, 0.803.

Carrying out the experiments again in the same way as in Fig. 8, it was found that the appropriate number of neurons for a single layer structure ranges from 100 to 400. In the following, the neuron scheme for each layer of each model containing a deep network structure is again designed, and the number of neurons in each layer is varied in steps of 50 in order to minimize the amount of computation. According to a single layer of 100 to 400 neurons, the initial number of neurons 100, step size of 50, a number of neuron structure scheme can be generated, each scheme in the training set to test, to get the loss function of these models of the minimum value of the TOP10 scheme, as shown in Table 1. Table 1 shows the number of neurons in the first, second, and third layers of the deep network, in ascending order of the scheme. The scheme with the smaller number of neurons corresponds to the smaller loss function. Therefore, the DeepCrossing, DeepFM, and DeepFM models in the subsequent study form the deep network according to the ascending number #1 scheme of the respective experimental loss function.

TABLE 1. Top 10 statistics of loss function in ascending order under different depth network shapes.

Ascending Scheme Number	DeepCrossing	DeepFM	DeepFM
#1	200,250,200	200,200,200	200,200,200
#2	300,300,300	250,250,250	150,200,150
#3	200,200,200	150,200,150	250,250,250
#4	150,200,150	300,300,300	250,300,250
#5	250,250,250	250,300,250	300,300,300
#6	100,200,300	100,200,300	100,200,300
#7	200,350,200	200,350,200	200,350,200
#8	300,400,300	150,300,150	150,300,150
#9	200,350,300	200,250,200	200,250,200
#10	300,300,200	300,350,300	300,400,300

In the following, the model performance of each contained neural network is compared again when using different activation functions, and the statistical results are shown in

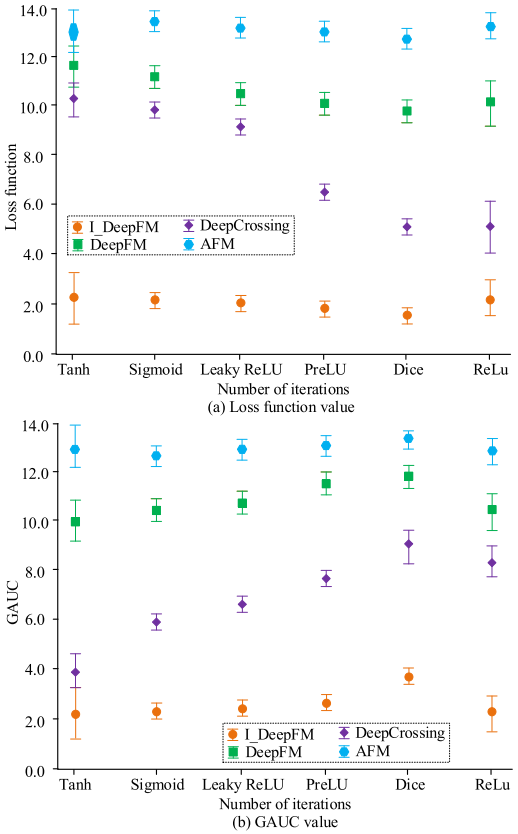

FIGURE 9. Impact of activation function on model performance.

Fig. 9.The horizontal axis in Fig. 9 is the activation function of common various neural network methods, and the vertical axes of subfigures (a) and (b) represent the value of the loss function, and the value of the GAUC, respectively. Here the experiment is still repeated several times and the statistics are plotted according to the mean and standard deviation. In Fig. 9, there are four algorithms that need to use the activation function to participate in the calculation, in terms of the loss function, the average value of the loss function of each model under the Dice activation function is the smallest and the fluctuation degree is also the most minimal, which is when the average value of the loss function of the models of DeepCrossing, DeepFM, DeepFM, and AFM is 5.61, 9.84, and 1.75, respectively, 13.1. From the perspective of GAUC value, the GAUC mean values of DeepCrossing, DeepFM, DeepFM, and AFM models under Dice activation function are the largest, which are 8.77, 11.71, 3.26, and 12.93, respectively.It is evident that algorithms requiring activation functions for calculations perform best under the Dice activation function. This function not only provides the best performance, but also outperforms other activation functions. It is evident that algorithms that require activation functions in their calculations perform best when using the Dice activation function. This function not only solves the problem of gradient explosion during model training, but also significantly alleviates the issue of internal covariate bias [20], [21]. The

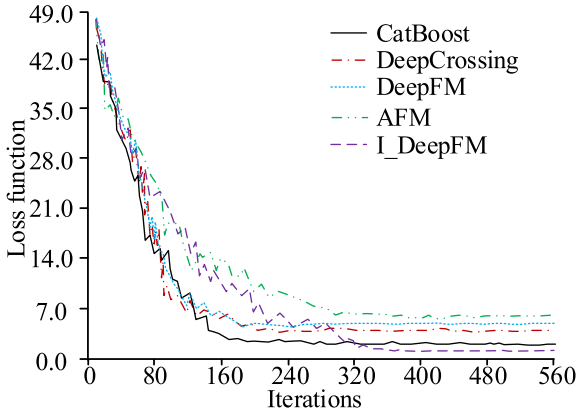

FIGURE 10. Change of loss function of each recommended model during training.

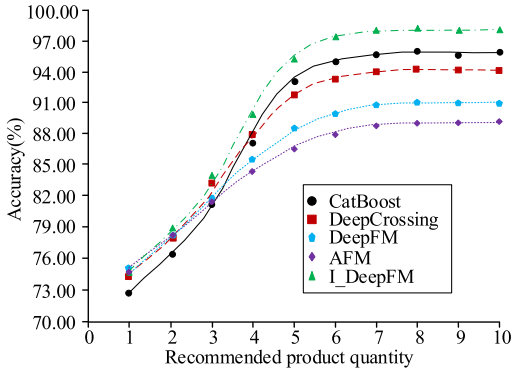

FIGURE 11. Accuracy curve of recommended product quantity.

optimal parameters and structures of some algorithms are obtained from the above research results, and these optimal solutions are used in the subsequent calculations.

The final comparison of the loss function performance of each model on the training set after setting up some of the models according to the optimal scheme is shown in Fig. 10. Note that DeepFM in Fig. 10 represents the recommender system designed for this research. The horizontal and vertical axes in Fig. 10 represent the number of training sessions and loss function values, respectively, using line styles and colors to distinguish different recommendation models. As shown in Fig. 10, when the training times is less than 150 times, the loss function value of each model decreases rapidly with the growth of iterations. When the number of iterations exceeds 150, the decline of the loss function value of each model gradually slows down and eventually converges to a fixed value range. When the number of iterations exceeds 342, the models are trained, and the loss function values of CatBoost, DeepCrossing, DeepFM, AFM, and DeepFM models are 2.48, 4.11, 5.82, 6.28, and 1.26, respectively.

B. COMPARATIVE ANALYSIS OF THE OPTIMAL SOLUTION AND OTHER METHODS

Firstly, the recommendation accuracy of each recommendation model is analyzed under different conditions of the number of recommended goods, and the statistical results are shown in Fig. 11. All the experimental scenarios in Fig. 11 are carried out on the whole test set. The horizontal axis of Fig. 11 represents different CR quantity scenarios, the vertical axis represents the accuracy rate (Accuracy) in %, different signs represent different recommendation algorithms, and different styles of lines represent the regression curves of the recommendation accuracy data of different recommendation algorithms. Figure 11 shows that the accuracy rate increases rapidly with a large number of recommendations when the number of CR is small. However, when the number of CR is greater than 4, the growth rate of the accuracy rate slows down dramatically. Additionally, when the number of CR is greater than 6, the accuracy rate almost converges, and the change becomes negligible. Purely from the perspective of the size of the accuracy rate, the CR here should be set to 10, but the number of recommendations is too much will consume the user's energy to make decisions. Considering all the factors, the number of CR is uniformly set to 7, and the recommendation accuracy rates of CatBoost, DeepCrossing, DeepFM, AFM, and DeepFM models are 96.25%, 94.33%, 90.75%, 88.73%, and 98.49%, respectively.

Calculations on the test sets of the two datasets are performed to obtain Fig. 12. The horizontal axis of Fig. 12 is used to show the evaluation metrics and the recommendation models, while the left vertical axis and the right vertical axis represent the corresponding data of the Criteo and Avazu datasets, respectively. In Fig. 12, the recommendation results of each recommendation model are generally poor due to the more scarce user history information in the Avazu dataset, while the recommendation accuracies of each recommendation model in the Criteo dataset are mostly higher than 0.700.Specifically, the CatBoost, DeepCrossing, DeepFM, AFM, and DeepFM models in the AUC, GAUC of the recommended computed subject operating characteristic curves on the more data-rich Criteo test set are 0.809, 0.734, 0.681, 0.627, 0.842 vs. 0.801, 0.796, 0.785, 0.783, 0.805,

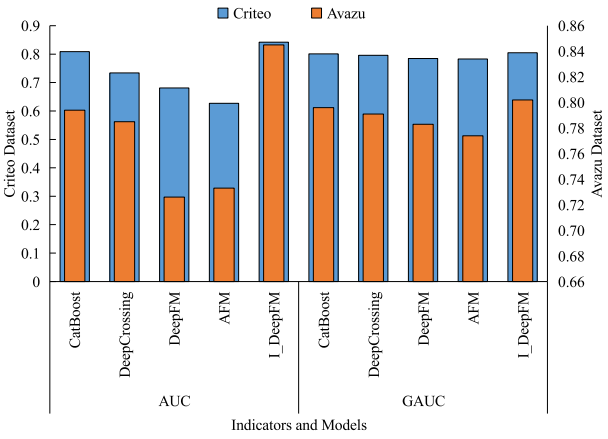

FIGURE 12. Recommendation accuracy performance of each model on two datasets.

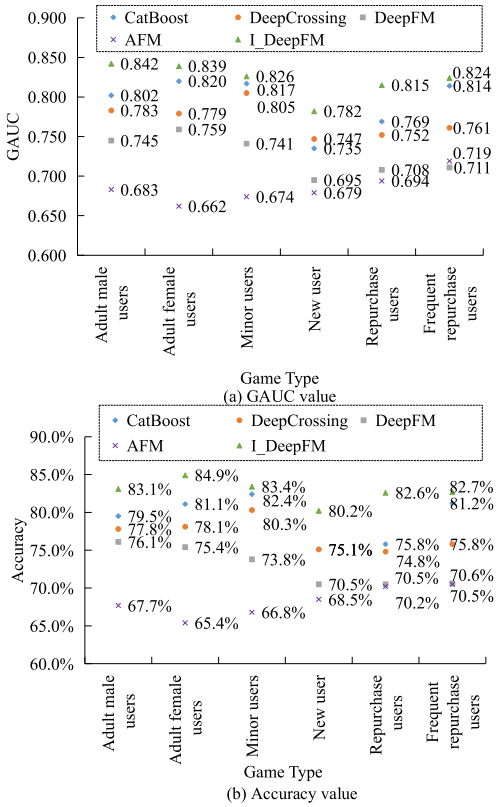

FIGURE 13. Comparison of CR accuracy for different types of users.

respectively. The DeepFM recommender model designed in this study under the same condition The propulsion accuracy is significantly higher than the comparison model.

The recommendation accuracy of each model on different users in the dataset is further detailed in Fig. 13.In Fig. 13, ''Frequent Repurchase Customer'' refers to users who have purchased the product at least three times and the purchase frequency is much higher than the average repurchase frequency of the product, ''Repurchase Customer'' refers to users who have purchased the product at least 1 time and the purchase frequency is not frequent. ''Repurchasing customer'' refers to users who have bought the product at least once and the purchase frequency is not frequent, and ''new customer'' refers to users who do not have any purchase record. Figure 13 shows that recommendation models have higher accuracy in recommending products to adult male users, adult female users, and underage users. However, the accuracy is lower for new users due to the sparse information available to the models. This makes it difficult for the models to accurately predict the purchasing preferences of new users. The DeepFM model designed in this study has a recommendation GAUC and accuracy of 0.842, 0.839, 0.826, 0.782, 0.815, 0.824 with 83.1%, 84.9%, 83.4%, 80.2%, 82.6%, 82.6%, and 82.7% in sexual users, adult female users, underage users, new users, repurchasing users, and frequent repurchasing users, respectively, 82.7%, which was significantly better than the comparison model.

TABLE 2. Comparison of operational efficiency and time consumption of all models.

CR model	data set	Experimental protocol number	Training time (s)	Recommended speed for test set (s/1000 users)
CatBoost	Criteo	#11	8.5	5.3
DeepCrossing	Criteo	#12	45.1	28.3
DeepFM	Criteo	#13	37.8	18.2
AFM	Criteo	#14	21.6	11.6
DeepFM	Criteo	#15	29.4	15.4
CatBoost	Avazu	#21	7.2	5.2
DeepCrossing	Avazu	#22	41.8	30.6
DeepFM	Avazu	#23	31.0	19.4
AFM	Avazu	#24	17.3	10.2
DeepFM	Avazu	#25	16.5	15.0

TABLE 3. Comparative statistics of literature [22], [23], and the methods used in this article.

Product recommendation model	Data set	GAUC	Accuracy	Recommended speed for test set (s/1000 users)
[23] Literature methods	Criteo	0.815	0.822	24.6
[24] Literature methods	Criteo	0.820	0.817	32.6
I_ DeepFM	Criteo	0.839	0.835	15.4
[23] Literature methods	Avazu	0.804	0.812	22.9
[24] Literature methods	Avazu	0.811	0.815	38.1
I_ DeepFM	Avazu	0.862	0.874	15.0

Finally, the comparison of all the models in terms of running efficiency and time consumption is shown in Table 2.

In Table 2, due to the fact that Avazu dataset contains less information, the training of each model on this dataset consumes significantly less time, but as a whole, there is no significant difference in the recommendation speeds of the same model trained using different datasets on the corresponding test sets. Specific analyses show that the DeepFM model designed in this study has 15.4s/1000 users and 15.0s/1000 users on the Criteo and Avazu datasets, respectively, and its computational speed is in the middle of that of the AFM model and the DeepFM model.

To enhance the reliability of the experimental results, the recommended models developed in references [22] and [23] are now selected for comparative testing. shown in Table 3. The results indicate that the DeepFM recommendation model, which was designed in this study, outperforms the models from references [22] and [23] in terms of recommendation results on both GAUC and Accuracy datasets.

V. CONCLUSION

This paper offers a hybrid AM, adaptive regularity improved DeepFM method, and plans and performs a performance test experiment to address the problem of insufficient recommendation accuracy in CR systems. The experimental results showed that when the number of hidden layers is 3, the mean value of the loss function and the mean value of the GAUC of DeepCrossing, DeepFM, and DeepFM models are 4.13, 6.27, and 1.59, and 0.799, 0.797, and 0.803, respectively, and the appropriate number of neurons of their single-layer structure ranges from 100 to 400. The algorithm using the activation function involved in the calculation had the best results. 400. Under the Dice activation function, the methods using the activation function engaged in the computation had the best overall performance. After setting up the algorithms according to the optimal scheme found by the experiments, multi-algorithm comparison experiments were carried out. After the iterations exceeded 342, each model completes training, at which time the loss function values of CatBoost, DeepCrossing, DeepFM, AFM, and DeepFM models were 2.48, 4.11, 5.82, 6.28, and 1.26, respectively. When the CRs exceeded 6, the accuracies of each model practically converge with minor adjustments. When all CRs were added together, the recommendation accuracy of CatBoost, DeepCrossing, DeepFM, AFM, and DeepFM models was 96.25%, 94.33%, 90.75%, 88.73%, and 98.49%, respectively. On the more data-rich Criteo test set, CatBoost, DeepCrossing, DeepFM, AFM, and DeepFM models had AUC and GAUC of 0.809, 0.734, 0.681, 0.627, and 0.842 versus 0.801, 0.796, 0.785, 0.783, and 0.805, respectively, for recommending the construction of subjects' operational characteristic curves. The DeepFM model's recommendation GAUC and accuracy were 0.842, 0.839, 0.826, 0.782, 0.815, 0.824, respectively, compared to 83.1%, 84.9%, 83.4%, 80.2%, 82.6%, 82.7% for sexual users, adult female users, underage users, new users, repurchase users, and frequent repurchase users. The computational speed of the DeepFM model on Criteo and Avazu datasets was 15.4s/1000 users and 15.0s/1000 users, respectively, placing it between the AFM model and the DeepFM model. The recommendation accuracy of the CR system designed in this research has an advantage, but owing to the research settings, it was not possible to test the recommendation effect of this system in other sectors, which is an issue that needs to be worked on in future research.

REFERENCES

[1] Y. Cui, "Intelligent recommendation system based on mathematical modeling in personalized data mining," *Math. Problems Eng.*, vol. 2021, pp. 6672036.1–6672036.11, Feb. 2021, doi: 10.1155/2021/6672036.

[2] M.-C. Chiu, J.-H. Huang, S. Gupta, and G. Akman, "Developing a personalized recommendation system in a smart product service system based on unsupervised learning model," *Comput. Ind.*, vol. 128, Jun. 2021, Art. no. 103421, doi: 10.1016/j.compind.2021.103421.

[3] A. Da'u, N. Salim, and R. Idris, "An adaptive deep learning method for item recommendation system," *Knowl.-Based Syst.*, vol. 213, Feb. 2021, Art. no. 106681, doi: 10.1016/j.knosys.2020.106681.

[4] U. Yadav, N. Duhan, and K. K. Bhatia, "Dealing with pure new user cold-start problem in recommendation system based on linked open data and social network features," *Mobile Inf. Syst.*, vol. 2020, pp. 8912065.1–8912065.20, Jun. 2020, doi: 10.1155/2020/8912065.

[5] S. Akram, S. Hussain, I. K. Toure, S. K. Yang, and H. Jajal, "ChoseAmobile: A web-based recommendation system for mobile phone products," *J. Internet Technol.*, vol. 21, no. 4, pp. 1003–1011, Jul. 2020, doi: 10.3966/160792642020072104010.

[6] T. Chen, "A fuzzy ubiquitous traveler clustering and hotel recommendation system by differentiating travelers' decision-making behaviors," *Appl. Soft Comput.*, vol. 96, Nov. 2020, Art. no. 106585, doi: 10.1016/j.asoc.2020.106585.

[7] F. Luo, G. Ranzi, W. Kong, G. Liang, and Z. Y. Dong, "Personalized residential energy usage recommendation system based on load monitoring and collaborative filtering," *IEEE Trans. Ind. Informat.*, vol. 17, no. 2, pp. 1253–1262, Feb. 2021, doi: 10.1109/TII.2020.2983212.

[8] T. R. Gwadabe and Y. Liu, "Improving graph neural network for session-based recommendation system via non-sequential interactions," *Neurocomputing*, vol. 468, pp. 111–122, Jan. 2022, doi: 10.1016/j.neucom.2021.10.034.

[9] Z. Roozbahani, J. Rezaeenour, A. Katanforoush, and A. J. Bidgoly, "Personalization of the collaborator recommendation system in multilayer scientific social networks: A case study of ResearchGate," *Expert Syst.*, vol. 39, no. 5, pp. e12932.1–e12932.18, Dec. 2021, doi: 10.1111/exsy.12932.

[10] W. Zhang, X. Zhang, and D. Chen, "Causal neural fuzzy inference modeling of missing data in implicit recommendation system," *Knowl.-Based Syst.*, vol. 222, Jun. 2021, Art. no. 106678, doi: 10.1016/j.knosys.2020.106678.

[11] A. Da'u, N. Salim, and R. Idris, "Multi-level attentive deep user-item representation learning for recommendation system," *Neurocomputing*, vol. 433, pp. 119–130, Apr. 2021, doi: 10.1016/j.neucom.2020.12.043.

[12] A. Sturluson, A. Raza, G. D. Mcconachie, D. W. Siderius, X. Z. Fern, and C. M. Simon, "Recommendation system to predict missing adsorption properties of nanoporous materials," *Chem. Mater.*, vol. 33, no. 18, pp. 7203–7216, Sep. 2021, doi: 10.26434/chemrxiv.14205929.

[13] P. Mazumdar, B. K. Patra, and K. S. Babu, "Cold-start point-of-interest recommendation through crowdsourcing," *ACM Trans. Web*, vol. 14, no. 4, pp. 1–36, Aug. 2020, doi: 10.1145/3407182.

[14] L. Li, Z. Zhang, and S. Zhang, "Hybrid algorithm based on content and collaborative filtering in recommendation system optimization and simulation," *Sci. Program.*, vol. 2021, pp. 7427409.1–7427409.11, May 2021, doi: 10.1155/2021/7427409.

[15] B. Peng, "Research and implementation of electronic commerce intelligent recommendation system based on the fuzzy rough set and improved cellular algorithm," *Math. Problems Eng.*, vol. 2021, pp. 6671219.1–6671219.18, Jan. 2021, doi: 10.1155/2021/6671219.

[16] Z. Yang, F. Zhou, L. Yang, and Q. Zhang, "A new prediction method for recommendation system based on sampling reconstruction of signal on graph," *Expert Syst. Appl.*, vol. 159, pp. 113587.1–113587.12, Nov. 2020, doi: 10.1016/j.eswa.2020.113587.

[17] Q. Liang, X. Zheng, Y. Wang, and M. Zhu, "O3ERS: An explainable recommendation system with online learning, online recommendation, and online explanation," *Inf. Sci.*, vol. 562, pp. 94–115, Jul. 2021, doi: 10.1016/j.ins.2020.12.070.
[18] S. E. Morris, L. A. Grohskopf, J. M. Ferdinands, C. Reed, and M. Biggerstaff, "Evaluating potential impacts of a preferential vaccine recommendation for adults 65 years of age and older on U.S. influenza burden," *Epidemiology*, vol. 34, no. 3, pp. 345–352, Feb. 2023, doi: 10.1097/ede.0000000000001603.
[19] N. Nassar, A. Jafar, and Y. Rahhal, "A novel deep multi-criteria collaborative filtering model for recommendation system," *Knowl.-Based Syst.*, vol. 187, Jan. 2020, Art. no. 104811, doi: 10.1016/j.knosys.2019.06.019.
[20] H. Song and N. Moon, "Eye-tracking and social behavior preference-based recommendation system," *J. Supercomput.*, vol. 75, no. 4, pp. 1990–2006, Apr. 2019, doi: 10.1007/s11227-018-2447-x.
[21] Z. Li, B. Cheng, X. Gao, H. Chen, and G. Chen, "A unified task recommendation strategy for realistic mobile crowdsourcing system," *Theor. Comput. Sci.*, vol. 857, pp. 43–58, Feb. 2021, doi: 10.1016/j.tcs.2020.12.034.
[22] C. Choudhary, I. Singh, and M. Kumar, "SARWAS: Deep ensemble learning techniques for sentiment based recommendation system," *Expert Syst. Appl.*, vol. 216, Apr. 2023, Art. no. 119420, doi: 10.1016/j.eswa.2022.119420.
[23] J. Attieh and J. Tekli, "Supervised term-category feature weighting for improved text classification," *Knowl.-Based Syst.*, vol. 261, Feb. 2023, Art. no. 110215, doi: 10.1016/j.knosys.2022.110215.

ZHEYI WANG was born in Chongqing, China, in 1988. He received the bachelor's and first master's degrees from the University of Bristol, U.K., and the second master's degree from The University of Hong Kong. He is currently a Researcher in AI and big data with Citigroup. His research interests include recommender systems, computer vision, and big data analysis.

YANLI KUANG was born in Chongqing, China, in 1996. She received the M.Sc. and M.Eng. degrees from Imperial College London. Her contributing expertise as a healthcare management consultant in Japan, specializing in innovative strategies for optimal healthcare delivery. Her research interests include big data applications in materials, healthcare, and data analysis.

XIN LYU was born in Chongqing, China, in 1991. She received the bachelor's degree from Chongqing University of Posts and Telecommunications, Chongqing. She is currently a Researcher in data mining and data analysis with Southwest University, China. Her research interests include recommender systems, NLP, and AIGC.