A Multi-behavior Recommendation Algorithm Based on Personalized Federated Learning

Zhongqin Bi[1], Yutang Duan[1], Weina Zhang[1(✉)], and Meijing Shan[2]

[1] School of Computer Science and Technology, Shanghai University of Electric Power, Shanghai, People's Republic of China
mszhangwn@mail.shiep.edu.cn

[2] Institute of Information Science and Technology, East China University of Political Science and Law, Shanghai, People's Republic of China

Abstract. Multi-behavior recommendation algorithms comprehensively use various types of interaction behaviors between users and items, such as clicking, collecting, purchasing, and commenting, to model user preferences and item features. It captures high-level interactions between users and items, and effectively alleviates the data sparsity problem in recommendation algorithms. However, most existing multi-behavior recommendation algorithms are mainly centralized learning models. User behavior data is collected and uploaded to the server to train recommendation model parameters, which poses a risk of data leakage and compromises user privacy. To address this problem, a multi-behavior recommendation algorithm based on the federated learning paradigm (FedMB) is proposed. This approach uses the federated learning framework to establish a separate model for each end device and utilizes the data of the end device for user-end model training, which improves the privacy and security of user data. To enhance privacy and security during parameters uploaded, all uploaded parameters will be encrypted, At the same time, the precedence chart is used to optimize the model parameters distributed by the server, thereby improving the recommendation quality of the overall model. Compared with that of the latest methods, our federated model achieves good performance on the three datasets.

Keywords: Multi-behavior recommendation · Privacy security · Federated learning · Personalized model · Parameter encryption

1 Introduction

The purpose of a recommendation system is to analyze the user's personalized preferences and to recommend content to alleviate information overload. Personalized recommendation models acquire users' explicit or implicit information through the interaction between users and products. By doing so, they can obtain user preferences and more accurately learn the embedded expressions of nodes, thus improving the accuracy of recommendations [1].

This work was supported by Project of Shanghai Science and Technology Committee (No. 23010501500).

H. Gao et al. (Eds.): CollaborateCom 2023, LNICST 563, pp. 134–153, 2024.
https://doi.org/10.1007/978-3-031-54531-3_8

However, most traditional recommendation behaviors are based on a single behavior. For instance, in Fig. 1 (a), the recommendation system mainly relies on the user's purchase behavior to collect information and provide product recommendations. Users often generate more than one piece of interactive behavior data during the purchase process. In addition to purchase behavior, there are also actions such as adding items to the shopping cart and browsing, which can be considered interactive behaviors, such as Fig. 1 (b). The recommendation system can better discover user interests and preferences to assist the target behavior (such as purchasing) by using a variety of behavioral interactive information.

To fully extract the information from multi-behavior interactions, various multi-behavior recommendation models have appeared in recent years [2–6]. One approach is to directly build a variety of behaviors and to apply each behavior to a single line as a recommended model, without considering the differences between different behaviors. However, some models provide different learning rights to different behaviors to simulate the importance of various behaviors and to distinguish different behavioral semantics [3].

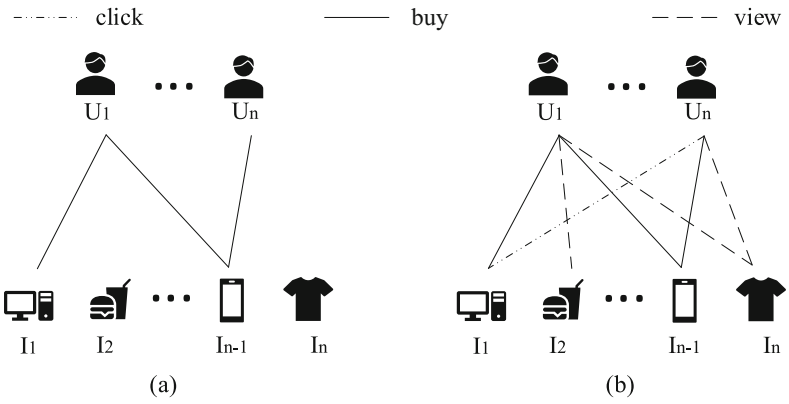

Fig. 1. Single behavior examples and multi-behavior examples in e-commerce scenarios.

Currently, effective integration of multiple types of behavior and capturing differences in multi-behavior are the main research methods recommended in many studies [7]. Commonalities between target and auxiliary behaviors must be identified to improve the target behavior and achieve higher recommendation effects. For example, commodities added to the shopping cart through purchasing behavior must have commonalities, such as the same style, category, or price. However, sparse supervision signals in traditional datasets do not guarantee the quality of graph learning, and buying behaviors are often the target behaviors in most multi-behavior recommendation models. GU S et al. proposed a method to address this issue by dividing different behaviors into two views for two or two comparison learning [2], but this method ignores the impact of auxiliary behavior on the target behavior.

Therefore, the use of user data for separate training models and independent recommendations is proposed to address the issue of personalization of behavior. Traditional

multi-behavior recommendation models often focus on the data from the interaction of the overall user and ignore the differences between different users. By training the model continuously to allow it to learn the differences between different behaviors or the connection between users, the model can effectively improve recommendation accuracy. However, this paper highlights the need to balance the effect and accuracy of the model, which requires further study.

In traditional machine learning models, centralized learning is a mainstream method [8], where the provider of the service collects user data and trains the machine learning model by using interactive information from users and products. However, due to restrictions on regulations and laws such as the GDPR [9], data security and privacy protection have become increasingly important. To address this issue, federated learning (FL) was proposed in 2016 [10], allowing data collection and training to be completed on end devices without the need for transmission. FL greatly improves user data privacy protection performance and has much application space for privacy protection and multiparty computing [11]. Although various privacy protection algorithms and encryption algorithms have been proposed, such as encryption algorithms based on cryptography [12], blurred disturbance methods [13], and the differential privacy [14], the balance between the effect and accuracy of the model must be further studied. FL an effective way to address this problem.

The main contributions of this article are as follows:

- To reduce the risk of privacy leakage in the process of multi-behavior recommendation, we use the federated framework to improve its security, and propose a multi-behavior recommendation algorithm based on federated learning.
- To improve the recommendation effect of the federated model, we adopted the precedence chart method [15]. This method optimizes the iterative parameters through the precedence chart method, and in the experiment, the adoption of the precedence chart method can further enhance the model recommendation effect.
- For the traditional federated aggregation algorithm, there is a risk of privacy leakage during the upload of parameters. In this process, noise data are added, position parameters are established, and real parameters are identified on the server. In addition, the security during the upload of the parameters is improved.
- The effectiveness of our method is verified on three real-world datasets, which demonstrates that our method advances the recommendation performance compared with other baselines.

Organization. The remainder of this article is structured as follows: Sect. 2 describes related work, including introductions to multi-behavior recommendation, discussions on machine learning models of privacy recommendation methods, and introduction to the basic knowledge of federated learning. Section 3 introduces FedMB, which is a multi-behavior recommendation adaptation of FL settings. In Sect. 4, we present a hyperparameter study, an ablation study, and comparative experiments on FedMB. In Sect. 5, we summarize our work and draw conclusions.

2 Related Work

In the past three years, researchers' recommendations for multi-behavior have focused on the optimization of recommendation effects, but few people are paying attention to privacy protection in the process of multi-behavior recommendation.

2.1 Multi-behavior Recommendation Algorithm

In recent years, graph neural networks have been widely used in multi-behavior recommendation algorithms. For instance, Jin B et al. proposed a user-item communication layer to capture the behavior semantics and explore the intensity of behavior [3]. In 2021, Xiao L et al. integrated multi-behavior mode into the meta-learning paradigm and automatically extracted the heterogeneity and interaction of behavior [4], but they did not consider the impact of time on user behavior. Xiao L et al. proposed a time-coding strategy to incorporate time perception into the context and developed a multi-behavior mutual attention encoder to learn different types of behavioral structure dependencies [5]. However, they overlooked the complexity of user-item interactions. In 2022, Wei W et al. proposed a new comparison meta learning model that maintains dedicated cross-type behavior for different users [6]. It effectively learns the characteristics of users and items by using deep learning frameworks. These papers mainly focused on exploring behavior information and ignored the personalized perspective of user behavior. Wu Y et al. addressed this issue by using user personalized sequence information and global map information in the multi-view comparison learning [16]. However, these models have not considered the personalized training of the model at the data level.

2.2 Privacy Protection Recommendation Algorithm

In this section, we provide an overview of recommendation methods for privacy protection in both centralized and decentralized settings. Privacy protection recommendation systems must prevent the leakage of user personal privacy data, while also defending against attacks from various sources. Different types of attacks, such as user attribute attacks [8], reasoning attacks, and attack attacks, require different defense mechanisms. To effectively address these attacks, privacy protection algorithms were developed. There are three main ways to protect user data: cryptography-based privacy protection algorithms, data disturbance-based privacy protection algorithms, and federated learning-based privacy protection methods. These methods aim to achieve data privacy protection through algorithmic approaches. In this article, we focus on the use of federated learning architecture to achieve privacy protection. The data are encrypted during the transmission process. The privacy of user data can be better protected.

Federated learning is a distributed machine learning framework that is designed for privacy protection. FL stores the user's original data locally and uses the intermediate parameters of the client and server to optimize the system, resulting in improved forecasting performance. In this article, we combine the recommendation algorithm with federated learning, which allows us to shift the centralized learning framework to the federated learning paradigm. As a result, the federated recommendation algorithm based on privacy protection has garnered significant attention from researchers.

The traditional federated recommendation algorithms encrypt data based on the federated architecture, but there is still a risk of privacy leakage. Therefore, researchers focus on developing federated privacy recommendation algorithms. Early federated recommendation frameworks required users to upload only gradient information, but this information can still lead to privacy breaches. In 2020, Di Chai et al. proposed a distributed matrix decomposition framework by using homomorphic encryption [17], but this did not guarantee the security of the data source. To address this, Guanyu Lin et al. proposed a solution based on explicit feedback [19], which involved creating a collection of noninteracted items to predict user preferences. This approach improved privacy protection and ensured the security of terminal data processing. In 2021, Chuhan Wu et al. proposed using pseudo interacting data and anonymous neighbor methods to enhance privacy protection performance [20]. V Asileios Perifanis et al. proposed the FedNCF model [21], which uses SecAvg to resolve privacy issues in small-scale datasets. Jingwei Yi et al. developed the Efficient-FedRec model [22], which uses an effective security aggregation agreement to protect user privacy in training. Wei Yuan et al. proposed the FRU model, which enables the deletion of user contribution data to improve privacy protection [23]. These approaches aim to improve the privacy protection mechanisms of federated recommendation algorithms while ensuring high recommendation performance.

The use of a centralized learning model is common in multi-behavior recommendations because the use of global user data for training and can lead to more accurate recommendations. However, this approach has a significant risk of privacy leakage. Additionally, the recommendations provided are often more general and less personalized due to the use of models for the entire user base. To address these issues, we propose a personalized federated recommendation framework. This approach uses personalized models to provide more accurate recommendations for user data while ensuring privacy protection. The framework includes a privacy protection module during the parameter upload process, further improving the privacy performance and recommendation quality of the model.

Table 1. Comparison of different methods in privacy protection

	NMTR	EHCF	RGCN	MB-GMN	S-MBRec	FedMB
User data storge	Center	Center	Center	Center	Center	Local
Rec process protection	✔	✔	✔	✔	✔	✔
Rec result protection	✘	✘	✘	✘	✘	✔

To better demonstrate the advantage of our approach, we summarize the comparison between FedMB and existing methods on exploiting privacy protection in Table 1. “Rec” means recommendation, “Center” and “Local” represent centralized and decentralized data storage, respectively.

2.3 Federated Learning

Federated learning is a machine learning technology that enables the training of machine learning models in a decentralized and distributed manner. The core idea of this learning approach is that the training data on each device do not need to be sent to a central server for coordination among different entities. This approach is different from the traditional machine learning training method and provides a higher level of privacy protection.

Federated Learning Algorithm. The federated learning algorithm follows a specific process, which includes collecting data at the initial stage, establishing a client model, and training the client data on the client model. Each client trains its own client model, and after the training is complete, each client uploads the model parameters or gradient to the server. The server conducts overall training based on these parameters and does not collect any private data from the client, thus maintaining the privacy of the data. Essentially, the server acts as a learning resource for the terminal equipment. Once the server processes the parameters uploaded by the client, it distributes them back to the client for further training. The process is summarized as follows:

Step1: The server provides the global model to the clients. This global model can be either an initial randomized model or a pretrained model.
Step2: The client uses its own data to conduct local training and to update the model.
Step3: The client uploads the model parameters or intermediate parameters to be updated to the central server.
Step4: The server aggregates the model parameters or intermediate parameters from the local client and performs multiple rounds of repeated iteration updates.

The above steps are repeated multiple times until the model converges, and then local models are used for reasoning and prediction. Throughout the process, users' privacy data, such as browsing history, likes, and collection history, are preserved locally, ensuring the safety of user privacy data. This approach greatly improves the privacy protection level of the learning model, while also providing accurate recommendations and maintaining the confidentiality of user data. Therefore, it is a more effective and secure method for machine learning in sensitive environments.

Federated Aggregation Algorithm. The most common approach to general aggregation is the federated average aggregation algorithm (Fedavg) [10]. In a global iteration, each participating client completes a small number of local iterations, and then uploads the parameters or gradient of the training instance to the server.

The server aggregates the uploaded parameter set or gradient set by using an aggregator, which is updated globally in the following ways:

$$w_{t+1} = \sum_{i=1}^{|c|} \frac{n_i}{n} w_t \tag{1}$$

where $|c|$ is the number of selected participants in a training round, w is a model parameter, n_i is the number of examples participating in this aggregation, n is the sum of the total number of participating instance training, w_t is the model parameter after the training of this participating instance, and t is the number of iterations.

3 Proposed Method

3.1 Problem Definition

We consider a scenario with multiple users (N > 2), each holding a private dataset. The goal is to build a federated learning system that can train models without compromising users' privacy. The datasets are generated locally by the users without any mutual transmission or interaction. We assume that the multi-behavior data generated by the users follow the principle of independence and distribution [21]. For a few users with less behavioral interaction, a pseudo interactive item is generated. Although this might have a negative impact on the recommendation results, this approach ensures the quality of the diagram learning, and the impact on the overall framework is small. This article is primarily focused on the stability of the overall framework.

The framework generates a list of previous recommendations for users using the local recommendation model. Training and recommendation are performed on the client side to ensure user privacy. For multi-behavior, the GCN [24] layer is used to extract user behavior characteristics.

3.2 FedMB Framework

The FedMB framework is illustrated in Fig. 2 and consists of four main parts: the client's training module, the server-side parameter selection module, the server-side parameter aggregation module, and the parameter encryption module. To enhance security during parameter transmission, noise data are added. To protect privacy, the model parameters after user-side model training are used. At the central server-side, the aggregated parameters are sent back to the client to complete each round of iteration training. The working principles of each module are explained in detail below.

Client Training. We adopt a self-supervised approach for multi-behavior recommendations, which is partitioned by user IDs in the dataset to allocate models for different users. The GCN layer is used to learn the embeddings of users and items, effectively extracting their personalized interaction characteristics (different users have different user-item interaction data, as shown in the Fig. 2). To differentiate between the importance of different behaviors, we propose a supervised task, and then use automatic learning to aggregate the embeddings of multi-behavior to distinguish between target behavior and auxiliary behavior. In each subgraph, the user and behavior embeddings are represented by R_k, where R_k denotes the k-th behavior graph, and the adjacent matrix A_k can be obtained from matrix R_k.

$$A_k = \begin{pmatrix} 0 & R_k \\ R_k^T & 0 \end{pmatrix} \tag{2}$$

Then, the GCN multilayer message communication formula is used to obtain the nodes of different behaviors embedded in the matrix. The formula is as follows:

$$X_k^{(l+1)} = \sigma\left(\widehat{A_k} X_k^{(l)} W_k\right) \tag{3}$$

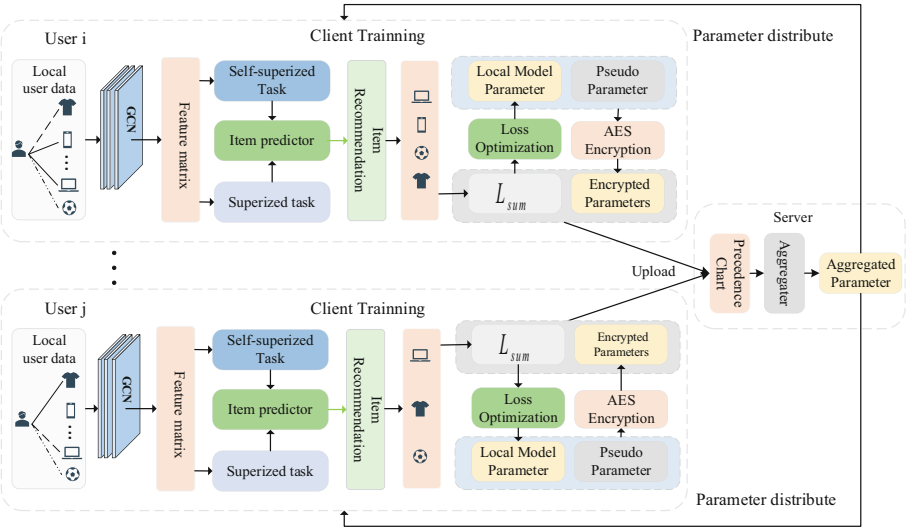

Fig. 2. The framework of FedMB

where $\widehat{A_k} = D_k^{-\frac{1}{2}}(A_k + I_k)D_k^{-\frac{1}{2}}$ is a self-connected normalized matrix. D_k is a $|V|*|V|$ dimension matrix under the k behavior. $|V|$ is the sum of the number of users and the number of items, $|V| = |U| + |I|$. I_k is the dimension matrix of $|V|*|V|$. $X_k^{(l)} \in R^{|V|*d}$ is the node embedded matrix in the node of the I_k behavior in the convolution layer. d is the dimension embedded. W_k and σ are model training parameters and non-linear activation functions. In order to ensure the embedding of the short connection node, the function of this article uses the function to merge all layers.

$$X_k = \int \left(X_k^{(l)}\right) \tag{4}$$

where $l = [0, 1 \ldots, L]$. X_k is composed of the user embedded matrix $X_{Uk} \in R^{|U|*d}$ and the item embedded matrix $X_{Ik} \in R^{|I|*d}$. $\int$ is the last layer of the connection operation.

First, a_{uk} is the semantic fusion coefficient of the user under the K behavior. Among them, we consider the proportion of the user's first behavior in all behaviors, and recognize the strength of different behaviors, as shown below:

$$a_{uk} = \frac{\exp(w_k * n_{uk})}{\exp(w_m * n_{um})} \tag{5}$$

where w_k said that under the behavior of the behavior k of all users, n_{uk} is the number of associations of user u under k.

X_{UK} and X_{IK} denote the embedded matrix of users and items under behavior k, while x_{uk} and x_{ik} denote the embedding of user u and item i under behavior k. Then the representation of all behaviors is merged. For user u, we merge all representations, as shown below.

$$e_u = \sigma\left\{W\left(\sum\nolimits_{m=1}^{K} a_{uk} * x_{uk}\right)\right\} \tag{6}$$

where W is the different behavior types. The characteristics of the item are static. For this reason, we combine the item's representation under different behaviors through series operations, as shown below.

$$e_i = g\{Cat(x_{ik})\} \tag{7}$$

where $k = [1, 2, \ldots, K]$, g is a multi-layer perception machine (MLP) [25], and Cat denotes the connection operation.

For supervision and training tasks, the loss function used is the BPR [26].

$$L_{st} = \sum_{(u,i,j)\in O} -log\left\{\sigma\left(e_u^T e_i - e_u^T e_j\right)\right\} \tag{8}$$

where e_i is the embedded item, e_u is the embedded item of the user, $O = \{(u, i, j)|(u, j) \in O_+, (u, j) \in O_-\}$ is the training task, O_+ is the observed interaction, and O_- is the unsteady interaction, which is used to generate positive and negative samples.

For unsupervised training tasks, we adopt a comparative method of learning and perform comparative learning between target behavior and auxiliary behavior subgraphs.

$$L_{sst_{k'}}^{user} = \sum_{u\in U} -log\frac{\sum_{u^+\in U} exp\left\{(x_{uK})^T x_u + \frac{k\prime}{\tau}\right\}}{\sum_{u^-\in U} exp\left\{(x_{uK})^T x_u - \frac{k\prime}{\tau}\right\}} \tag{9}$$

where $(x_{uK}, x_{u+k\prime})$ denotes the positive pair, and $(x_{uK}, x_{u-k\prime})$ denotes the negative pair. τ is a parameter, U is a user set, $k\prime$ is auxiliary behavior, and K is target behavior. Similarly, we obtain a comparison loss $L_{sst_{k\prime}}^{item}$ through the loss function under the combination user and the item, and then, we add multiple comparison losses to obtain the total unsupervised loss, as shown below.

$$L_{sst} = \sum_{k'=2}^{K}\left(L_{sst_{k'}}^{user} + L_{sst}^{item}\right) \tag{10}$$

Parameter Selection on the Server. The main purpose of parameter aggregation is to select and weight the model parameters uploaded by clients. To achieve this, each uploaded parameter set is assigned a score based on the training loss reported by the client. This score reflects the quality of the client's model training, with higher scores indicating better performance. During each round of global iteration in federated Learning, the server aggregates the parameters of each client and computes a score value for each parameter. The parameters with high scores are then selected for aggregation to improve the overall model's recommendation quality.

To optimize the parameter selection process, we propose a method that uses pairwise comparison of precedence chart to assign weights to each factor. The use of precedence chart for weight assignments ensures fairness, avoids extreme score values, and reduces algorithmic complexity and training overhead compared to traditional machine learning methods. Applying this method to the federated Learning framework can shorten the

global iteration time and can improve recommendation efficiency without compromising the results.

The weight assignment process involves evaluating the dataset, selecting the initial normal results, and assigning an initial weight score. Then, pairwise comparisons are performed to establish the corresponding mapping between the evaluation data and the weight value. Finally, the weight values are normalized and arranged to obtain different weights for different data.

Parameter Aggregation on the Server. The core engine of adaptive learning is personalized learning recommendation, which has greatly improved accuracy and diversity. Federated learning is a powerful data privacy protection machine learning solution that can safeguard data privacy and security while sharing data value. In this study, we combine federated learning with multi-behavior recommendation to create a federated personalized multi-behavior recommendation system and assess its feasibility and effectiveness.

Federated learning is comprised of two methods: global federated learning model training and training single models on clients. The global training method uses federated training plus local adaptation strategies and depends on the generalization performance of the global model for effective recommendations. The client training method focuses on personalized models that provide tailored solutions for each user. It modifies the aggregation process of the FL model to establish a personalized model and is advantageous for solving user preference drift. In this article, we use the client training method to train client models and to build a multi-behavior recommendation framework with strong generalization.

The client's training task is composed of self-supervised and unsupervised tasks. The total loss of the client's training is represented by the following formulas. L_{st} denotes unsupervised loss, while L_{sst} denotes the supervision loss.

$$L_{sum} = L_{st} + \lambda L_{sst} + \mu \|\Theta\|_2^2 \tag{11}$$

where Θ stands for training parameters, λ and μ indicate the control of self-supervision and $L2$ regularization proportional parameters, respectively.

$$S_n = PC(l_1, l_2 \ldots l_{n-1}, l_n) \tag{12}$$

On the server-side, the parameters are selected by the collected client model parameters and losses. Using precedence chart (PC) to give scores to different parameters, as shown in Formula (12), l_n denotes the n loss corresponding to the parameter, P_n is the n score, $S_n = (P_1, P_2 \ldots P_{n-1}, P_n)$, and the score set. We use the score set to give a native value to give power.

$$W_n = \frac{\gamma(P_1, P_2 \ldots P_{n-1}, P_n)}{\sum_{i=1}^{|n|} P_i} \tag{13}$$

where γ denotes the judgement function, which is used to remove extreme values. W_n denotes the weights allocated according to different scores. $W_n = (w_1, w_2 \ldots w_{n-1}, w_n)$,

W_n is the weight set corresponding to the n score of S_n. Perform parameter collection best according to different weights.

$$W_m = SelectN_\beta(w_1, w_2 \ldots w_{n-1}, w_n) \tag{14}$$

where $SelectN_\beta$ is the selection function, β is the proportion of the weight set to remove weights, $m = n * (1 - \beta)$, and $W_m = (w_1, w_2 \ldots w_{m-1}, w_m)$, which denotes the selected parameter weight collection. After obtaining the corresponding parameters, the parameters are averaged, and the aggregation formula is as follows.

$$p_{t+1} = \sum_{i=1}^{|m|} \frac{n_i}{n} p_t^i \tag{15}$$

where t is the current number of aggregation rounds, and the parameter p_{t+1} after the final aggregation is distributed down from the server-side, and participates in the next global iteration.

Parameter Encryption. Upon completing the training, the client uploads its parameters to the server, but this process is vulnerable to parameter interception and thus, can result in user privacy leakage.

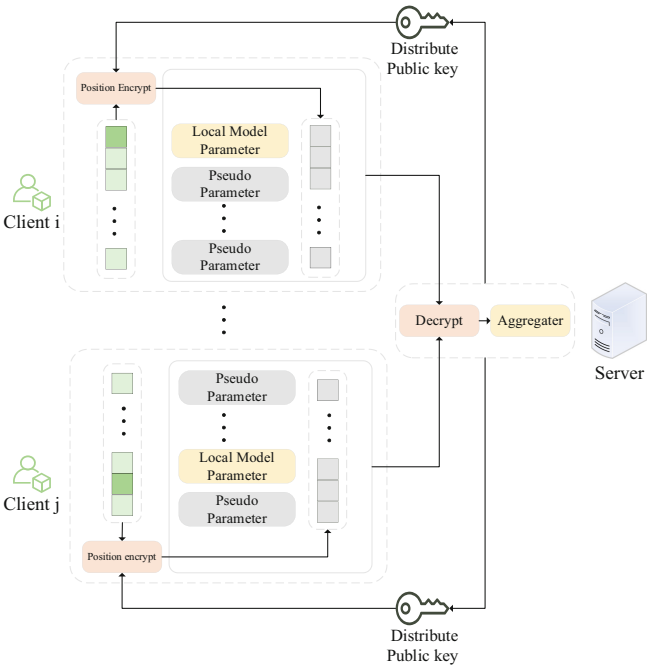

Fig. 3. Parameter encryption

To enhance the security, we increased the difficulty of identifying the real parameters, thereby reducing the risk of external malicious attacks. In accordance with the Fig. 3, the

structure of the parameter encryption module is illustrated. In this article, noise data are added to the parameters uploaded from the client to increase the difficulty of identifying the real parameter. The noise data are derived from the pseudo parameters generated by the client itself. The formula is as shown below:

$$MI_{t+1}^{i} = Ip_{t+1}^{i} + \sum_{j \in c: j < k} IR_j \tag{16}$$

where IR_j denotes randomly generated parameters, Ip_{t+1}^{i} are real user parameters, and MI_{t+1}^{i} is the mask parameters. k is the number of pseudo ginseng generated by the client.

Adding noise to the parameters makes it more challenging to recognize the real parameters on the server. To address this issue, position parameters are generated on the client and uploaded to the server to identify the actual parameters. For position parameters, AES encryption is used, which is efficient and fast [27], and the server-distributed key is employed for encryption and decryption on the server. Compared to direct encryption of parameters, this proposed encryption method significantly reduces the encryption time overhead. The figure below illustrates the main process.

4 Experiments

In this experiment, we evaluate the recommendation quality of FedMB by using three datasets and introducing the experimental settings in detail. The evaluations are conducted through an ablation study and hyperparameter study to analyse the model and compare the effects of different modules and parameters on results. Finally, we compare the experimental results with other models to provide a comprehensive evaluation of FedMB.

4.1 Datasets and Evaluation Settings

To evaluate the performance of FedMB, we verify the model effect in 3 real-world datasets. The details are described as follows:

Table 2. Beibei, Taobao, Yelp dataset statistics

Dataset	User	Item	Interactions	Behavior Type
Beibei	21716	7977	3.36×10^6	{View, Cart, Purchase}
Taobao	48749	98249	2.40×10^6	{Click, Add to cart, Purchase}
Yelp	19800	22734	1.40×10^6	{Like, Neutral, Tip, Dislike}

The sparsity of data can be significantly reduced by setting both the target behavior and auxiliary behavior. Although the target behavior is typically considered the supervisory signal for training the model, it is often sparse, and sparse supervision cannot guarantee the training quality of the model. Even in multi-behavior recommendation,

this phenomenon persists. To avoid this situation, we also regard auxiliary behavior as a supervisory signal for model training. When we cannot make recommendations based on user purchase behavior accurately, we can alternatively make recommendations based on user browsing or clicking behavior, which has been found to be effective.

To evaluate the model's recommendation effectiveness, we set up a corresponding test set for the client and utilize the user's target behavior as the test set. When the user has no target behavior interaction, the user's auxiliary behavior is used as the target behavior. In this paper, we utilize the NDCG and the Recall to better assess the model's recommendation effectiveness. NDCG accounts for the position of the recommendation ranking, while recall calculates the probability of the model recommending items that match the user's true interests. we report the Recall and NDCG values for k = 40 and k = 80, respectively, and the formulas for calculating NDCG and Recall are as follows:

$$DCG@k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{17}$$

$$NDCG@k = \frac{DCG@k}{IDCG@k} \tag{18}$$

where rel_i denotes the relevance score of the i^{th} recommended item, k is the length of the recommendation list, $DCG@K$ represents the discounted cumulative gain of the top k recommended items, and $IDCG@K$ represents the discounted cumulative gain of the top k relevant items.

$$Recall@k = \frac{TP@k}{TP@k + FN} \tag{19}$$

where the $TP@k$ metric represents the number of items that the user actually liked among the top k recommended items, while FN is defined as the difference between the total number of items that the user actually liked and the number of items that the user actually liked in the top k recommended items.

Experimental setup. The experiments in this paper are performed in an Ubuntu16.04 operating system. The hardware configuration used in the experiment is as follows: CPU: Intel Core i9-10900K; GPU: NVIDIA RTX 3080 TI. The programming language used is Python 3.7, and the deep learning development framework is PyTorch 1.7.

4.2 Experimental Settings

In this paper, we present a recommendation method that establishes end models and sets an aggregator on the server end. To optimize the aggregated parameter set, the federated weighted average algorithm is employed.

In terms of the setting of the federated framework, we divide users into different clients based on their IDs. By treating each ID as a different client, user data are read and the model is trained on the client side. As the number of clients is large, we set the number of client model trainings to 5, with a small number of iterations to avoid excessive time and space overhead. Due to the transitive nature of client parameters in the federated learning architecture, we focus more on the fitting of the overall global

model on the server-side. An aggregate aggregator is established on the server-side, which aggregates the parameters and distributes them to the clients. To ensure that as many clients as possible participate in the global training, we set a large number of random extraction times. For example, on the Beibei dataset, which has a total of 21,716 users, we extract 70 clients each time for 430 global iterations, with a total of 30,100 random extractions, covering as many user IDs as possible.

4.3 Hyperparameter Study

Formula (14) introduces β as the proportion of deleted parameters in the parameter set, with its size determining the number of aggregated parameters. Notably, the final aggregated parameter set decreases as the β value increases. Nonetheless, both excessively large and small β values compromise the quality of the final model's recommendation results. Furthermore, the size of the aggregated parameter set directly impacts the aggregation time, which subsequently influences the model's training duration. In this study, we adopt varying β sizes to measure the model's training time and efficacy.

The Effect of Parameter β on Recommendation. During the execution of the weighted aggregation algorithm, the iterative transfer parameter's quality is optimized.

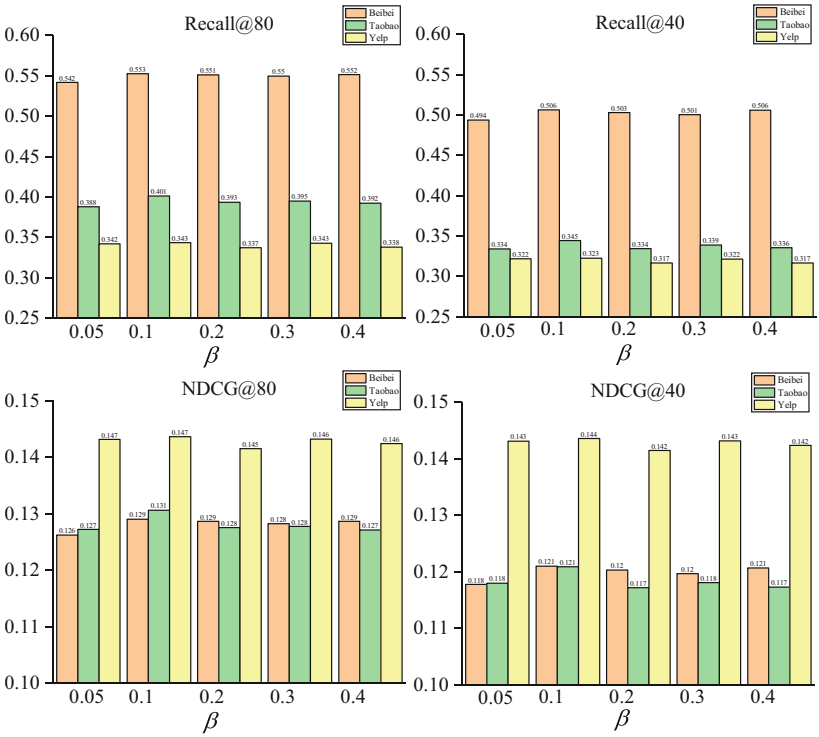

Fig. 4. Effects of different values of β on the results of FedMB

However, the final model results differ slightly under different optimal parameter β sizes. As illustrated in Fig. 4. From observation, a β value of 0.1 is optimal for the Beibei, Taobao, and Yelp datasets. Setting a larger β may remove too many well-trained parameters, resulting in a small parameter set that cannot achieve universal aggregation in training, hence reducing the quality of aggregated parameters and the recommendation effect. Conversely, a smaller β may select a smaller proportion of parameters, allowing client parameters with training disadvantages to participate in training, ultimately compromising the model training and results. Therefore, the experimental results show that setting β to less than 0.1 leads to a decrease in the results, validating the appropriateness of using a β value of 0.1.

The Effect of Parameter β on Training Time. While verifying the impact of parameter β on the results, we discovered its effect on the fitting time of the federated model training.

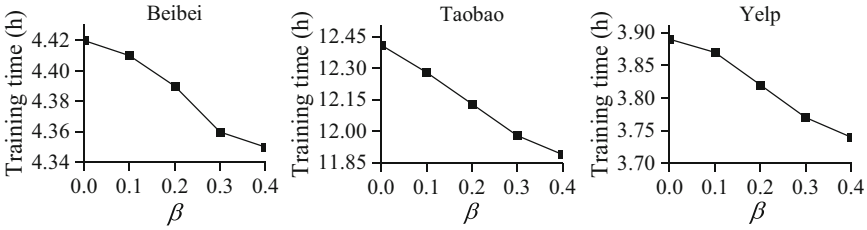

Fig. 5. Effect of parameter β on training time

Upon analyzing Fig. 5, we observed that the value of β affects the training time of the training client. Specifically, not using the precedence chart ($\beta = 0$) resulted in a longer training time compared to using the method. We also evaluated different values of β and set the total duration of the framework training under the training of 30,000 user instances to {0.1, 0.2, 0.3, 0.4}. We discovered that increasing the optimal parameter β led to a reduction in the overall framework's training time. The training time for the Taobao dataset decreased from 12.41 h to 11.89 h, the training time for the Beibei dataset reduced from 4.42 h to 4.35 h, and the training time for the yelp dataset reduced from 3.89 h to 3.74 h. Therefore, we concluded that when performing weighted optimal aggregation, a larger optimal parameter results in fewer server-processed parameters, leading to a shorter overall training time. Additionally, we observed that the training time for the Taobao dataset was longer compared to the training times for the Beibei and yelp datasets. The difference in training times was due to different datasets having varying orders of magnitude of interactive items. Since Taobao had more interactive items, the model required a larger interaction matrix, causing increased overhead and longer training times. Therefore, we set β to 0.1 to decrease the training time while ensuring recommendation quality.

4.4 Ablation Study

In federated aggregation, clients usually upload the gradient or parameters of the user model. However, this paper also includes the client training loss value to assess user training quality. The precedence chart plays a vital role in FedMB.

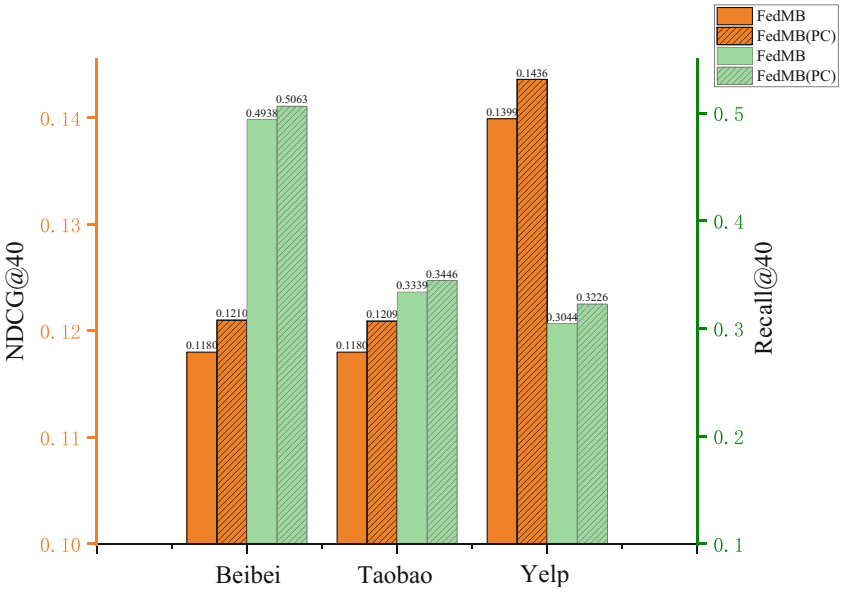

Fig. 6. Effect of precedence chart on the results of FedMB

In this paper, we conduct an experiment to demonstrate its effectiveness in improving the model's results under the influence of the precedence chart. We compare the results of our federated model, FedMB, with FedMB(PC), and observe an improvement in our model's performance. To investigate the impact of the precedence chart on the experimental results, we conducted an experiment and discussed its effect on the effectiveness of the federated recommendation model with and without PC. We refer to the "precedence chart" as PC for convenience. The results are shown in Fig. 6. The average quality of the recommended results on the Beibei dataset increased by 2.5%, while the improvement on the Taobao and Yelp datasets was 3% and 4.9%, respectively. These results indicate that using the precedence chart method to filter out the excellent training end model parameters and aggregate them has significant advantages over traditional federated aggregation methods. Furthermore, it improves the recommendation effect of the model. Therefore, we conclude that utilizing the precedence chart to optimize federated aggregation is crucial for enhancing the performance of the FedMB.

4.5 Comparative Experiments

In this paper, N GCN behavior user interaction views and N user behavior interaction views are set up, where N is the number of behaviors, N is set to 3. The weights are

learned by optimizing supervised and unsupervised losses using the Adam [28] optimizer with a learning rate of 0.0001. In each experiment, 30,000 user rounds are trained on the Beibei and Taobao, Yelp datasets, with the number of client training rounds set to 5 rounds. The number of users participating in each round is set to 70. The difference in model quality in the global model with different numbers of users can be attributed to the heterogeneity among participants, which leads to variability in the overall effect. In each joint training, a random sampling method is adopted to ensure that each client has the same probability of being sampled in each global iteration. To comprehensively train the client, a higher sampling number and a larger number of global iteration rounds must be selected. Based on hyperparameter research, setting β to 0.1 ensures optimal efficiency with relatively low time consumption. The training batch size on clients is set to 1024.

We conduct a comparison between FedMB and several classic multi-behavior recommendation models, including NMTR, EHCF, RGCN, MB-GMN, and S-MBRec.

NMTR [29]: This is a hierarchical multitask recommendation model that effectively manages complex interconnected behaviors.

EHCF [30]: This model uses heterogeneous collaborative filtering and transferable predictions for multi-behavior. It transfers information across behaviors to enhance its precision and personalization.

RGCN [31]: This model improves factorization models with multistep information propagation in relational graphs, leading to enhanced accuracy in tasks such as link prediction and entity classification.

MB-GMN [4]: This model efficiently predicts multiple source-target tasks by combining meta learning with graph neural networks to obtain cross-behavior predictors.

S-MBRec [2]: The model adopts a star comparison strategy to create a comparison view of target and auxiliary behavior. It uses supervised and unsupervised tasks to recommend models, improving the accuracy and effectiveness of the recommendations.

Our goal is to emphasize the unique features and advantages of FedMB in comparison to these traditional models. Through this comparison, we provide a more comprehensive understanding of the strengths and limitations of each model. We also highlight the potential applications of FedMB in various real-world scenarios. According to Table 2, the proposed federated multi-behavior recommendation model outperforms the latest S-MBRec model on the Beibei dataset, with an average index increase of 12.6% points. While there are decreases in Recall@10 and NDCG@10 due to the larger number of interactions per user in the Beibei dataset, its performance also decreases in shorter recommendation lists when unable to obtain preferences from other users. However, the other two datasets exhibit some mitigation of this effect due to their smaller numbers of interaction items per user. Despite this, for other metrics such as Recall@40 and NDCG@40, they are both twice as high as the latest methods, and Recall@80 and NDCG@80 also experience significant improvement. On the Taobao dataset, all performance indicators show improvement, especially in Recall@40 and NDCG@40, resulting in an average indicator increase of 13.1% points. Similarly, the Yelp dataset also shows an average increase of 14.5% points. These results demonstrate the excellent recommendation performance of the proposed model, especially when recommending 40 items (Table 3).

Table 3. FedMB in Beibei,Taobao, and Yelp results comparison

Datatset	Metric	NMTR	EHCF	RGCN	MB-GMN	S-MBRec	FedMB
Beibei	Recall@10	0.0460	0.0456	0.0483	0.0496	**0.0529**	0.0443
	Recall@40	0.1370	0.1270	0.1262	0.1497	0.1647	**0.5063**
	Recall@80	0.1989	0.1923	0.1912	0.2018	0.2740	**0.5525**
	NDCG@10	0.0124	0.0131	0.0122	0.0134	**0.0148**	0.0137
	NDCG@40	0.0192	0.0217	0.0226	0.0395	0.0429	**0.1210**
	NDCG@80	0.0422	0.0435	0.0440	0.0465	0.0615	**0.1290**
Taobao	Recall@10	0.0367	0.0292	0.0370	0.0423	0.0608	**0.1764**
	Recall@40	0.0485	0.0594	0.0703	0.0873	0.1027	**0.3446**
	Recall@80	0.0982	0.1032	0.1525	0.1553	0.1647	**0.4011**
	NDCG@10	0.0237	0.0285	0.0213	0.0325	0.0391	**0.0819**
	NDCG@40	0.0404	0.0373	0.0314	0.0396	0.0464	**0.1209**
	NDCG@80	0.0329	0.0390	0.0443	0.0475	0.0583	**0.1306**
Yelp	Recall@10	0.0195	0.0172	0.0210	0.0230	0.0259	**0.1986**
	Recall@40	0.0697	0.0704	0.0844	0.0899	0.1135	**0.3226**
	Recall@80	0.0903	0.0873	0.1105	0.1350	0.1548	**0.3433**
	NDCG@10	0.0191	0.0166	0.0199	0.0275	0.0287	**0.1145**
	NDCG@40	0.0305	0.0298	0.0263	0.0220	0.0337	**0.1436**
	NDCG@80	0.0355	0.0330	0.0395	0.0430	0.0438	**0.1471**

The proposed federated multi-behavior recommendation model is different from the baseline model in that it trains a personalized model for each user and makes recommendations based on their interactions with personalized items. Unlike the comparative methods, this model explores the interaction of users' personalized items and trains a model for each user, and it does not weaken with the overall user preference. Additionally, a security encryption module is added to the parameter upload process to ensure user privacy and security while improving recommendation performance. Consequently, the model effectively explores users' personalized preferences and makes recommendations without revealing their private data.

5 Conclusion

In this paper, we present FedMB, a multi-behavior recommendation model that utilizes federated learning to preserve data privacy during the recommendation process. FedMB employs a personalized federated learning framework to address the challenge of user personalization in multi-behavior recommendation. Through personalized user models, the system can provide high-quality recommendations while still protecting the privacy of users' data. The effectiveness and efficiency of FedMB are evaluated in experiments, and we also discuss the impact of using a precedence chart on the performance

of the federated model recommendation. The results show the feasibility of using federated learning in multi-behavior recommendation systems, with the proposed method performing well on three datasets.

In future research directions of multi-behavior recommendation under the federated learning paradigm, improving the security and efficiency of the system remains a top priority. Despite the higher privacy level provided by the decentralized data location and encrypted data transmission, there is still a risk of privacy leakage during parameter distribution. Additionally, enhancing the efficiency of client model training and compressing parameters to reduce communication overhead are crucial areas for further investigation.

References

1. Wang, L., Xiong, Y., Li, Y., Liu, et al.: A collaborative recommendation model based on enhanced graph convolutional neural network. J. Comput. Res. Dev. **58**(09), 1987–1996 (2021). (in Chinese)
2. Gu, S., Wang, X., Shi, C., et al.: Self-supervised graph neural networks for multi-behavior recommendation. In: International Joint Conference on Artificial Intelligence, Shenzhen (2022)
3. Jin, B., Gao, C., He, X., et al.: Multi-behavior recommendation with graph convolutional networks. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Xian, pp. 659–668 (2020)
4. Xia, L., Xu, Y., Huang, C., et al.: Graph meta network for multi-behavior recommendation. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Montreal, pp. 757–766 (2021)
5. Xia, L., Huang, C., Xu, Y., et al.: Knowledge-enhanced hierarchical graph transformer network for multi-behavior recommendation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 5, pp. 4486–4493 (2021)
6. Wei, W., Huang, C., Xia, L., et al.: Contrastive meta learning with behavior multiplicity for recommendation. In: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, New York, pp. 1120–1128 (2022)
7. Wu, J., Wang, X., Feng, F., et al.: Self-supervised graph learning for recommendation. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Montreal, pp. 726–735 (2021)
8. Zhang, H., Li, Y., Wu, J., et al.: A survey on privacy-preserving federated recommender systems. Acta Automatica Sinica **48**(09), 2142–2163. (in Chinese)
9. Voigt, P., Von dem Bussche, A.: The EU General Data Protection Regulation (GDPR). A Practical Guide, 1st edn. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57959-7
10. McMahan, B., Moore, E., Ramage, D., et al.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, Fort Lauderdale, pp. 1273–1282. PMLR (2017)
11. Shmueli, E., Tassa, T.: Secure multi-party protocols for item-based collaborative filtering. In: Proceedings of the Eleventh ACM Conference on Recommender Systems, Como, pp. 89–97 (2017)
12. Kim, S., Kim, J., Koo, D., et al.: Efficient privacy-preserving matrix factorization via fully homomorphic encryption. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, New York, pp. 617–628 (2016)

13. Berlioz, A., Friedman, A., Kaafar, M.A., et al.: Applying differential privacy to matrix factorization. In: Proceedings of the 9th ACM Conference on Recommender Systems, Vienna, pp. 107–114 (2015)
14. McSherry, F., Mironov, I.: Differentially private recommender systems: building privacy into the netflix prize contenders. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, pp. 627–636 (2009)
15. Lu, K.P., Chang, S.T.: Detecting change-points for shifts in mean and variance using fuzzy classification maximum likelihood change-point algorithms. J. Comput. Appl. Math. **308**, 447–463 (2016)
16. Wu, Y., Xie, R., Zhu, Y., et al.: Multi-view multi-behavior contrastive learning in recommendation. In: International Conference on Database Systems for Advanced Applications, Hyderabad, pp. 166–182 (2022)
17. Chai, D., Wang, L., Chen, K., et al.: Secure federated matrix factorization. IEEE Intell. Syst. **36**(5), 11–20 (2020)
18. Zhang, S., Yin, H., Chen, T., et al.: Pipattack: poisoning federated recommender systems for manipulating item promotion. In: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, New York, pp. 1415–1423 (2022)
19. Lin, G., Liang, F., Pan, W., et al.: Fedrec: federated recommendation with explicit feedback. IEEE Intell. Syst. **36**(5), 21–30 (2020)
20. Wu, C., Wu, F., Cao, Y., et al.: Fedgnn: federated graph neural network for privacy-preserving recommendation. In: Proceedings of the Thirty-Eighth International Conference on Machine Learning (2021)
21. Perifanis, V., Efraimidis, P.S.: Federated neural collaborative filtering. Knowl.-Based Syst. **242**, 108441 (2022)
22. Yi, J., Wu, F., Wu, C., et al.: Efficient-FedRec: efficient federated learning framework for privacy-preserving news recommendation. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Stroudsburg, pp. 2814–2824 (2021)
23. Yuan, W., Yin, H., Wu, F., et al.: Federated Unlearning for On-Device Recommendation. arXiv preprint arXiv:2210.10958 (2022)
24. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: Proceedings of the 5th International Conference on Learning Representations. Palais des Congrès Neptune (2017)
25. Pinkus, A.: Approximation theory of the MLP model in neural networks. Acta Numer. **8**, 143–195 (1999)
26. Rendle, S., Freudenthaler, C., Gantner, Z., et al.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, pp. 452–461 (2009)
27. Lee, B.H., Dewi, E.K., Wajdi, M.F.: Data security in cloud computing using AES under HEROKU cloud. In: 27th Wireless and Optical Communication Conference (WOCC), Hualien, pp. 1–5 (2018)
28. KingaD, A.: A method for stochastic optimization. In: Anon. International Conference on Learning Representations. SanDego (2015)
29. Gao, C., He, X., Gan, D., et al.: Neural multi-task recommendation from multi-behavior data. In: 2019 IEEE 35th International Conference on Data Engineering, Macau, pp. 1554–1557 (2019)
30. Chen, C., Zhang, M., Zhang, Y., et al.: Efficient heterogeneous collaborative filtering without negative sampling for recommendation. In: Proceedings of the AAAI Conference on Artificial Intelligence, New York, vol. 34, no. 01, pp. 19–26 (2020)
31. Schlichtkrull, M., Kipf, T.N., Bloem, P., et al.: Modeling relational data with graph convolutional networks. In: European Semantic Web Conference, Heraklion, pp. 593–607 (2018)