PriExRec: Defending Against Membership Inference Attacks in Federated Recommendation with Explicit Feedback

Rong Pu[1], Xiaochun Yang[1(✉)], Yinan Liu[1], Jian Li[1], Yaoyu Jin[1], Fanfei Song[1], and Bin Wang[1,2,3]

[1] School of Computer Science and Engineering, Northeastern University, Shenyang, China
{purong,jianli,yaoyujin}@stumail.neu.edu.cn,
{yangxc,binwang}@mail.neu.edu.cn, liuyinan@cse.neu.edu.cn,
fanfeisong@foxmail.com
[2] National Frontiers Science Center for Industrial Intelligence and Systems Optimization, China
[3] Key Laboratory of Data Analytics and Optimization for Smart Industry (Northeastern University), Ministry of Education, China

Abstract. Federated Recommender Systems (FedRecs) enhance privacy protection by keeping user data on local devices, thereby reducing data interactions between users and the server. However, the server can still potentially infer the user's relevant item set by analyzing the uploaded item gradients, which poses a threat to user privacy. In this paper, we propose a privacy-enhanced federated recommendation with explicit feedback named PriExRec to prevent user behavior data leakage in FedRecs. To reduce privacy risks while maintaining the utility of FedRecs, we first introduce an item mean imputation mechanism by combining Local Differential Privacy (LDP) with user rating habits. Empirical results reveal that the server can still successfully infer the user's relevant item set by accumulating analysis of user-uploaded gradients over multiple rounds. To address this, we introduce a virtual client and use a consistent hash function to perturb the order of uploaded item gradients, making it difficult for the server to infer the user's item set. Additionally, we adopt a federated feature compression mechanism, where both clients and servers use a compression matrix to handle high-dimensional features. Extensive experiments demonstrate the effectiveness of our proposed PriExRec in defending against interactive membership inference attacks (IMIA) while maintaining high recommendation quality.

Keywords: Federated recommendation · Privacy protection · Membership inference attacks.

1 Introduction

Federated Recommender Systems (FedRecs) are gaining widespread adoption due to their characteristic of not requiring users to transmit sensitive or private

data to a central server [22]. In this paradigm, the user's historical interaction data remains local device, while only model parameters are uploaded to the server. The server aggregates the parameters uploaded by clients (i.e., users) and returns the updated parameters to all clients, enabling a collaborative construction of a complete model [1]. Generally, existing FedRecs can be classified into explicit feedback and implicit feedback based on the form of user feedback. Explicit feedback includes user-provided ratings and reviews, which directly reflect user attitudes towards items, while implicit feedback is derived from users' historical behaviors (e.g., clicks, browsing). In FedRecs, users' explicit feedback is used to optimize the model but may also lead to privacy leakage risks. For instance, user rating data is vulnerable to interactive membership inference attacks (IMIA) [25] from an honest-but-curious server. IMIA aims to infer whether a user's interaction data is included in FedRecs, showing the privacy leakage risk of user interaction data.

Compared to implicit feedback, FedRecs with explicit feedback are more prone to membership inference attacks due to the model update mechanism. In FedRecs with implicit feedback, updated item embeddings contain both positive and negative samples, and the lack of clear meaning in negative samples increases the difficulty of membership inference attacks. In contrast, FedRecs with explicit feedback only update items that are rated, allowing the server to easily infer each client's relevant item set by simply checking which item embeddings were updated. This characteristic of FedRecs with explicit feedback places higher demands on user privacy protection, particularly in preventing interactive membership inference attacks. In this paper, we focus on FedRecs with explicit feedback.

Existing FedRecs with explicit feedback (e.g., FedRec [11], FedRec++ [10], and FR-FMSS [12]) attempt to defend against interactive membership inference attacks using techniques such as fake sampling. In FR-FMSS, clients randomly sample unrated items to generate fake gradients when uploading item gradients to the server. In FedRec, clients locally sample unrated items and fill in virtual ratings, which are used along with real ratings to update model parameters. FedRec++ further focuses on the impact of virtual ratings introduced in FedRec on model performance. However, existing FedRecs only consider user data privacy in a single model iteration, without adequately addressing privacy leakage risks throughout the entire model training process. The server only needs to record the list of items involved in gradient updates during each iteration, and by taking the intersection of all item lists after model training, it can infer the user's relevant item set with high probability.

To defend against IMIAs in federated recommender systems, this paper proposes a privacy-enhanced federated recommender model with explicit feedback named PriExRec. The core methodology of PriExRec includes three key strategies. *First*, we propose an improved item mean imputation scheme that utilizes item means generated with Local Differential Privacy (LDP) [5] along with users' own rating preferences (e.g., negative users tend to give low scores, positive users tend to give high scores), making the random sampling and impu-

tation of unrated items more reasonable in each iteration, thereby effectively preventing the server from inferring the user's relevant item set by inspecting the updated embeddings from clients. *Secondly*, to enhance model performance and defend against the server's attempts to infer specific user rating values, we design a federated feature compression mechanism. Both users and items are initially represented as high-dimensional features, which are processed using a low-dimensional compression matrix. During model updates, the client first compresses user embeddings into a low-dimensional space and uses these compressed features to update the model. Upon receiving the uploaded gradients, the server decompresses them to update item features and subsequently compresses these features before sending them back to the clients. This mechanism not only retains more fine-grained features and enhances model generalizability but also effectively defends against interactive membership inference attacks. *Lastly*, we introduce a virtual client that does not directly interact with the server. This virtual client broadcasts a random value to all users in each iteration, and users employ this value along with a consistent hash function to perturb the order of the uploaded item gradients. This perturbation ensures that the uploaded gradient order can be restored to the original sequence on the client side, preventing the server from accumulating user information over multiple iterations to infer relevant item sets, thereby further enhancing user privacy protection. The main contributions of this paper are as follows:

- To the best of our knowledge, we are the first to recognize that the accumulation of gradient parameters over multiple rounds can lead to user item set leakage and attempt to solve this privacy issue.
- By introducing a virtual client that does not directly interact with the server and implementing a feature compression mechanism, the proposed PriExRec effectively reduces the privacy leakage risk of user item sets and rating values due to the server's accumulation of gradients over multiple rounds.
- Extensive experiments on three real-world datasets verify that the proposed PriExRec not only ensures model performance but also effectively prevents membership inference attacks by the server.

2 Related Work

In this section, we mainly introduce the related work on federated recommendation and attacks against federated recommendation. Relevant foundational models, such as recommender systems, federated learning, local differential privacy, and recent advancements in consistent hashing, can be referred to [14, 19, 4].

2.1 Federated Recommendation

In recent years, federated recommender systems have received widespread attention for their ability to balance privacy protection with recommendation tasks.

By keeping users' raw rating data locally, federated recommender systems effectively prevent user information from being exposed to third parties. [18] proposed a sampling-based random perturbation scheme, which perturbs some elements of item feature gradients to meet local differential privacy (LDP) requirements, thereby reducing communication overhead to some extent. However, excessive noise injection leads to a significant decrease in model performance. [15] further proposed random perturbations on user-uploaded item gradients to ensure privacy protection that meets LDP standards, although this method improves privacy, it struggles to achieve desirable results in recommendation quality. [12] proposed sampling unrated items to generate fake gradients, which are uploaded together with real gradients to the server to create confusion. [2] emphasized the privacy protection of rating values but had limitations in hiding the rated items themselves. Additionally, FedGNN [20] proposed a federated learning-based privacy-preserving graph neural network recommender system, which protects user privacy through local GNN training and differential privacy techniques while effectively leveraging higher-order interaction information, achieving performance comparable to centralized methods. However, this approach assumes that items have prior knowledge of all neighboring users, which is not reasonable for protecting user rating information.

2.2 Attacks against Federated Recommendation

With the widespread application of FedRecs, its security issues have attracted the attention of researchers. Various attack methods and corresponding defense measures have been proposed in response to specific attacks. This section reviews these works.

Attacks. Studies have investigated target item promotion attacks in federated recommender systems [17, 27, 26], where malicious clients upload biased gradients to increase the exposure of specific target items. Sample selection-based methods [21] use user embeddings to select the most challenging positive and negative samples, aiming to maximally disrupt model performance. Methods in [17, 27] generate malicious gradients by approximating user features using public interaction data, with the goal of improving the recommendation opportunity for target items in federated recommender systems. In the context of news recommendation, untargeted attacks disrupt both news and user modeling by aligning malicious user updates in an opposite direction to normal users, thus reducing overall model performance [23]. Additionally, untargeted attacks like ClusterAttack [24] disrupt recommendation rankings by clustering item embeddings, leading to a degraded user experience.

Defenses. To address privacy and security concerns in federated recommendations, researchers have proposed various defense strategies against specific attacks. FSAD [9] proposed a gradient-feature-based detection method that uses a Bayesian classifier to identify malicious "sybil" attacks, though it struggles to cope with more sophisticated attack strategies, particularly those involving obfuscated malicious gradients. Differential privacy, with its mathematically provable privacy protection, has been widely applied in recent privacy solutions

[15]. To defend against interactive membership inference attacks, CIRDP [13] combines differential privacy and causal inference to protect privacy, but the introduced noise also reduces recommendation accuracy, making it challenging to balance privacy and performance. UC-FedRec [23] adopts a user-customized privacy protection level approach to balance recommendation effectiveness and privacy protection, but its protection remains unstable when facing attackers accumulating multi-round interaction information for global privacy leakage. These defense strategies demonstrate different ways of safeguarding privacy in federated recommender systems under specific attack contexts, but each has its own limitations. For example, some methods lead to decreased recommendation performance while enhancing privacy protection, while others struggle to handle global privacy leakage and stealthy attacks effectively.

3 Preliminaries

In this section, we first revisit the fundamental settings of FedRecs, as well as the relevant definitions of interactive membership inference attacks.

3.1 Federated Recommendation with Explicit Feedback

In this paper, we adopt the same model definitions and equations as those in [10] to ensure consistency in problem definition. Let $\mathcal{U}$ and $\mathcal{I}$ denote the set of users (i.e., clients) and the set of items. Each user u has a set of interaction records $\mathcal{R}_u = \{(u, i, r_{ui}); i \in \mathcal{I}_u\}$, where $\mathcal{I}_u$ is the set of items rated by the user u. In Probabilistic Matrix Factorization (PMF) [16], the predicted rating $\hat{r}_{ui}$ for user u on item i is represented as the inner product of the user's feature vector $U_{u\cdot} \in \mathbb{R}^{1\times d_1}$ and the item's feature vector $V_{i\cdot} \in \mathbb{R}^{1\times d_1}$, i.e., $\hat{r}_{ui} = U_{u\cdot}V_{i\cdot}^T$. In the federated learning framework, the server first initializes the shared item latent feature vectors $V_{i\cdot}$, $i \in \mathcal{I}$ and distributes them to all users, along with some hyper-parameters that are shared among all users. Meanwhile, each user also initializes their own user feature vectors $U_{u\cdot}$, $u \in \mathcal{U}$. Upon receiving the item feature vectors $V_{i\cdot}$, the client locally updates $V_{i\cdot}$ and $U_{u\cdot}$ based on the gradients $\nabla V_{i\cdot}$, $i \in \mathcal{I}_u$, and $\nabla U_{u\cdot}$ according to the loss function for several iterations. Since our research focuses on rating prediction with explicit feedback, we follow the gradients update method of FedRecs with explicit feedback:

$$\nabla V_{i\cdot} = y_{ui}(r_{ui} - U_{u\cdot}V_{i\cdot}^T) \cdot U_{u\cdot} \tag{1}$$

where $y_{ui} \in \{0, 1\}$ is an indicator vector, $y_{ui} = 1$ indicates that r_{ui} truly exists in the training data; in other words, $y_{ui} = 0$ indicates that the value of r_{ui} does not exist in the training data. Furthermore, we employ the Root Mean Square Error (RMSE) loss function, consistent with [10], to measure the discrepancy between the model's predicted ratings and the users' true ratings. Unlike centralized recommendation systems, for user u, the user's interaction data R_u and user latent feature vector U_u in federated recommendation systems always remain locally and are not shared with the server or other users.

3.2 Interaction-level Membership Inference Attack

In this paper, we follow the definition of interactive membership inference attacks (IMIA) as described in [25]. Specifically, we also assume that the central server is honest but curious, with prior knowledge including the federated protocol to be executed and some basic hyperparameters (e.g., learning rate, feature dimensions, etc.). The target of the server's attack is the user's private data (including private ratings and private parameters), but it does not disrupt the federated protocol. The server analyzes the parameters uploaded by the user to infer the user's interaction information:

$$\hat{\mathcal{R}}_u \leftarrow \nabla V_{i\cdot}, i \in \mathcal{I}_u \tag{2}$$

where $\hat{\mathcal{R}}_u$ represents the inferred set of true interaction records for user u, and ∇V_i. represents the gradient information uploaded by user u to the server. The central server aims to conduct the inference attack without affecting the normal learning and updating process of FedRecs.

4 PriExRec

In this section, we provide a detailed description of the federated recommendation scheme with explicit feedback PriExRec, to defend against interactive membership inference attacks by the server. We first define our studied problem and then describe the three specific defense modules included in the model. A discussion of the attack is provided in Sec. 5.2, while an overview of the overall PriExRec framework is illustrated in Fig. 1.

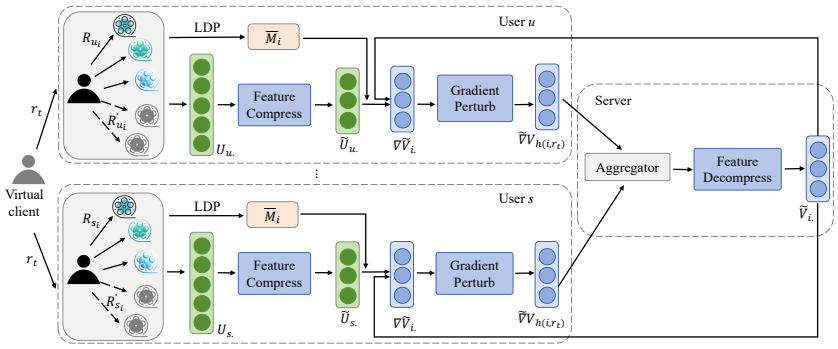


Fig. 1: The overview of PriExRec.

4.1 Problem Definition

In a federated recommendation model with explicit feedback, we denote the set of users (i.e., clients) and the set of items as $\mathcal{U}$ and $\mathcal{I}$, where $|\mathcal{U}| = n$ and $\mathcal{I} = m$.

Each user u has a set of interaction records $\mathcal{R}_u = \{(u, i, r_{ui}); i \in \mathcal{I}_u\}$, where $\mathcal{I}_u$ is the set of items rated by the user u. Our goal is to predict the ratings of all unrated items $\mathcal{I} \backslash \mathcal{I}_u$ for user u without revealing the user's private information (i.e., $\mathcal{R}_u$). Our work is based on the same problem definition as in Sec. 3.1, but has been extended to prevent privacy leakage.

4.2 Item Mean Imputation Scheme

To obtain the mean of items without revealing users' true rating information, and considering that user ratings are limited to discrete values, we introduce a multidimensional frequency estimation method based on LDP named RD+FD[OUE-r] [3]). This method effectively protects users' actual ratings while estimating the frequency of each rating value for each item, enabling secure mean calculation. Compared to other existing multidimensional frequency estimation methods, RD+FD[OUE-r] can result in less estimation error (i.e., the difference between the estimated frequency and the true frequency) [19]. Specifically, we treat the rating list $r_{u\cdot} \in \mathbb{R}^{1\times m}$ of user u as a numeric multidimensional array of size $1 \times m$ (to facilitate distinguishing the unrated items, we initially assign all unrated items an obvious rating value, such as 0). Given a privacy budget ϵ and a rating range $\mathfrak{R} = \{\mathfrak{r}_1, \ldots, \mathfrak{r}_k\}$, we apply the RD+FD[OUE-r] to the rating list $r_{u\cdot}$ of user u on the client side, and then send the perturbed rating list to the server. The server can then obtain the frequency estimate of any value $\mathfrak{r}_j \in \mathfrak{R}$ for item i as:

$$\hat{f}(\mathfrak{r}_j) = \frac{N_j \cdot m \cdot k - n \cdot [qk + (p-q)(m-1) + qk(m-1)]}{nk(p-q)} \tag{3}$$

where N_j is the number of times that the server records the item i with $\mathfrak{r}_j$, the perturbation probabilities are $p = \frac{1}{2}$ and $q = \frac{1}{e^\epsilon + 1}$. Note that the definition of local differential privacy is:

Definition 1. *A random mechanism $\mathcal{M}$ satisfies ϵ-local differential privacy if and only if, for any pair of possible inputs v and v', and for any possible output y of randomizer $\mathcal{M}$, we have*:

$$\Pr[\mathcal{M}(v) = t] \leq e^\epsilon \times \Pr[\mathcal{M}(v') = t] \tag{4}$$

where Pr[.] denotes the probability [5]. Besides, LDP has post-processing properties. That is, applying any function F to the output of a random mechanism $\mathcal{M}$ still satisfies ϵ-LDP for $F(\mathcal{M})$. The RD+FD[OUE-r] satisfies the ϵ-LDP in Definition 1, according to Eq. 3, we can prove that this frequency estimation is an unbiased estimate of the true frequency of $\mathfrak{r}_j$, i.e.,

$$E[\hat{f}(\mathfrak{r}_j)] = f(\mathfrak{r}_j) \tag{5}$$

Since $r_{ui} = 0$ in the rating list indicates that user u has not rated item i, the server only needs to compute the mean value $\bar{M}_i$ of non-zero ratings for item i,

$$\bar{M}_i = \frac{\sum_{j=1}^{k} \mathfrak{r}_j \hat{f}(\mathfrak{r}_j)}{\sum_{j=1}^{k} \hat{f}(\mathfrak{r}_j)} \quad (\mathfrak{r}_j \neq 0) \tag{6}$$

The server then sends the calculated list of item means $[\bar{M}_1, \ldots, \bar{M}_m]$ to all users, who independently sample and fill in their unrated items. According to the properties of LDP, we can use the item means calculated from Eq. 6 instead of the true item means for subsequent operations.

On the client side, we randomly sample some items $\mathcal{I}'_u$ and fill in ratings for them, where $|\mathcal{I}'_u| = \rho|\mathcal{I}_u|$ and ρ is the sampling parameter with $\rho \in \{1, 2, 3\}$. To make the filled ratings r'_{ui} more consistent with user u's own rating habits, we calculate the filled rating r'_{ui} as the sum of the mean $\bar{M}_i$ and the average difference between the user's existing ratings $r_{ui}(i \in \mathcal{I}_u)$ and the mean $\bar{M}_i$:

$$r'_{ui} = \bar{M}_i + \frac{\sum_{i \in \mathcal{I}_u}(r_{ui} - \bar{M}_i)}{|\mathcal{I}_u|}, \quad i \in \mathcal{I}'_u \tag{7}$$

Thus, we obtain the set of filled rating records for user u, denoted as $R'_u = \{(u, i, r'_{ui}); i \in I'_u\}$.

4.3 Federated Compression Mechanism

In this subsection, we further address the privacy protection of user rating values involved in model updates. Previous studies have shown that even if multiple rounds of gradient updates are performed locally on the user's device, the server can still approximately infer the user's rating values if it can access the item gradient information uploaded by the user over consecutive iterations. To address this issue, we propose a federated feature compression mechanism to enhance privacy protection.

Specifically, the client locally initializes the high-dimensional user feature vectors $U_{u\cdot} \in \mathbb{R}^{1 \times d_1}$ and generates a user compression matrix $C_u \in \mathbb{R}^{d_1 \times d_2}$. On the server side, high-dimensional item feature vectors $V_{i\cdot} \in \mathbb{R}^{1 \times d_1}$ and an item compression matrix $C_i \in \mathbb{R}^{d_1 \times d_2}$ are initialized. Both the client and the server then compress the high-dimensional features for users and items, respectively, resulting in:

$$\tilde{U}_{u\cdot} = U_{u\cdot}C_u, \quad \tilde{V}_{i\cdot} = V_{i\cdot}C_i^T \tag{8}$$

where $\tilde{U}_{u\cdot} \in \mathbb{R}^{1 \times d_2}$ and $\tilde{V}_{i\cdot} \in \mathbb{R}^{1 \times d_2}$ represent the compressed user and item features. This compression process retains most of the useful information while reducing the dimensionality of transmitted data, thus enhancing data privacy.

During model training, the client first receives the compressed low-dimensional item feature vector $\tilde{V}_{i\cdot}$ from the server and computes the predicted rating with the compressed user feature $\tilde{U}_{u\cdot}$, which is then used to construct the loss function. The loss function $\mathcal{L}$ is defined based on the compressed user and item feature vectors as follows:

$$\mathcal{L} = \sum_{u \in \mathcal{U}, i \in \mathcal{I}} (r_{ui} - \tilde{U}_{u\cdot}\tilde{V}_{i\cdot}^T)^2 + \lambda_u \|\tilde{U}_{u\cdot}\|^2 + \lambda_v \|\tilde{V}_{i\cdot}\|^2 \tag{9}$$

where λ_u and λ_v are the regulation coefficient of the compressed user feature vectors and the compressed item feature vectors. After sampling and filling the

unrated items for users as described in Sec. 4.2, to ensure that the model focuses more on the user's actual rated items and thus improves model performance, we set a decaying weight parameter $\alpha \in (0, 1)$ for the sampled unrated items. Then the client can perform local updates on the user features based on the loss function in Eq. 9 and calculates the low-dimensional item feature gradient:

$$\nabla V_i = (y_{ui} r_{ui} + \alpha(1 - y_{ui}) r'_{ui} - \tilde{U}_u \tilde{V}_i^T)\tilde{U}_u, \quad i \in \mathcal{I}_u \cup \mathcal{I}'_u \tag{10}$$

When the server receives the low-dimensional gradient uploaded by the client in the $(t + 1)$-th round of model iteration, it decompresses it to update the high-dimensional item features:

$$V_{i\cdot}^{(t+1)} = (\tilde{V}_{i\cdot} C_i^T) C_i \tag{11}$$

where $C_i^T \in \mathbb{R}^{d_2 \times d_1}$ is the transpose of the item compression matrix. This decompression ensures that the server updates the item features in the high-dimensional space, preserving fine-grained feature information. The decompression process allows the model to retain the details of item features in the high-dimensional space, enabling effective updates.

In federated recommendation systems, user features across different clients exhibit significant heterogeneity, and high-dimensional features are better at capturing these differences. The feature compression process filters and simplifies the information to retain as much useful information as possible while effectively preventing the server from accurately inferring users' original data from the high-dimensional features, thereby enhancing privacy protection.

4.4 Gradient Order Perturbation

Gradient Order Perturbation. To further enhance user privacy, we designed an item order perturbation and restoration mechanism based on a consistent hashing function (e.g. SHA-256) [6]. This mechanism utilizes random numbers generated by a virtual client to perturb gradients, making it difficult for the server to directly infer a user's related item set from the uploaded gradients.

We introduce a virtual client that generates a random value r_t for all users in t-th round of federated learning iterations and broadcasts it to all users:

$$r_t \sim \text{Uniform}(0, R) \tag{12}$$

where R is the range of the random number, ensuring that r_t is independently generated in each iteration. After receiving the random value r_t broadcast by the virtual client, each client perturbs the order of its locally computed compressed item gradients $\nabla \tilde{V}_{i\cdot}$ using the consistent hashing function h.

First, the client uses a consistent hashing function to generate an order mapping. The consistent hashing function $h(i, r_t)$ takes the item index i and random value r_t as inputs and generates a new perturbed item index $h(i, r_t)$:

$$h(i, r_t) = (i + r_t) \mod |\mathcal{I}| \tag{13}$$

where $|\mathcal{I}|$ is the total number of items, ensuring that the perturbed indices remain within the item index range. By adding the random value r_t, the client generates a new, unpredictable order of items.

Then, each client rearranges the compressed item gradient $\nabla V_{i\cdot}$ according to the generated order mapping to obtain the perturbed gradient:

$$\tilde{\nabla} V_{h(i,r_t)} = \nabla V_{i\cdot} \tag{14}$$

With this order perturbation, each client's upload order changes in every iteration, preventing the server from inferring user preferences and item sets by long-term tracking of fixed item features.

Item Order Increment Restoration. When the server receives the perturbed gradients uploaded by the client, it assumes that the original item order is maintained, and thus performs item feature updates based on this assumption. Therefore, in the next round of model updates, the client needs to restore the order of item features, focusing on the increment of item features rather than directly restoring the entire feature vector order.

Specifically, in the next model update round, the user first receives the compressed item feature vector $\tilde{V}_{i\cdot}^{(t+1)}$ returned by the server and calculates the increment in item features:

$$\Delta \tilde{V}_{i\cdot}^{(t)} = \tilde{V}_{i\cdot}^{(t+1)} - \tilde{V}_{i\cdot}^{(t)} \tag{15}$$

To restore the increment $\Delta \tilde{V}_{i\cdot}^{(t)}$ to the original order, the client uses the inverse mapping of the consistent hashing function from the previous update round:

$$h^{-1}(j, r_t) = (j - r_t) \mod |\mathcal{I}| \tag{16}$$

where j represents the perturbed item index. By computing the inverse mapping h^{-1}, the original item index can be obtained. The corresponding restored item feature increment is:

$$\Delta V_{i\cdot} = \Delta \tilde{V}_{h^{-1}(i,r_t)}^{(t)} \tag{17}$$

Next, the client adds the increment $\Delta V_{i\cdot}$ caculated by Eq. 17 back to the previous round's item features to obtain the correct item features for the current round:

$$V_{i\cdot}^{(t+1)} = V_{i\cdot}^{(t)} + \Delta V_{i\cdot} \tag{18}$$

Through this process, the client ensures that the item features participate in model training in the correct order, preventing model update bias due to order perturbation and ensuring the model's effectiveness and consistency.

5 Experiments

In this section, we discuss the experimental settings and present experiments conducted on three public datasets to verify the effectiveness of our proposed PriExRec in terms of both performance and privacy. We focus on the following four research questions:

– **RQ1:** How does our PriExRec perform in comparison to other models in the baselines in terms of model performance?
– **RQ2:** How much do different modules contribute to the model's performance?
– **RQ3:** What is the impact of different hyperparameters on the model's performance?
– **RQ4:** Compared to other models in the baselines, is our PriExRec more resilient to interaction-level membership inference attacks?

5.1 Experimental Settings

Datasets. We conducted experiments on three widely used public datasets in the recommender system domain, including Movielens 100k (ML100K)[1], Movielens 1M (ML1M)[2] [8], and a small dataset from Filmtrust[3] [7]. The statistics of the three datasets are shown in Table 1. We processed each dataset as follows: (i) we randomly split the dataset into five equally sized parts; (ii) we used four parts as training data and the remaining part as test data; (iii) we repeated the second step four times to obtain five different copies of training and test data. We conducted experiments and reported the average performance across these five data copies.

Evaluation Metrics. To assess the accuracy of rating predictions for each model, we employ two widely used evaluation metrics mean absolute error (MAE) and root mean square error (RMSE). These metrics help evaluate the performance of the models on the datasets.

Table 1: Dataset statistics.

Dataset	#Users	#Movies	#Ratings	Sparsity
ML100K	943	1,682	100,000	93.70%
ML1M	6,040	3,952	1,000,209	95.81%
Filmtrust	2,071	1,508	35,497	98.86%

Baselines. We compare our PriExRec with the following representative federated recommendation methods with explicit feedback: FedRec [11] is a federated recommendation model that utilizes each user's individual mean for data imputation. FedRec++ [10] improves upon FedRec by introducing noise-filtering clients, resulting in a performance-preserving federated recommendation model.

Parameter Configurations. In our work, we fix the dimension of the high-dimensional feature vectors $d_1 = 64$ and the dimension of the low-dimensional compressed feature vectors $d_2 = 15$. We search the best value of the learning rate

[1] https://grouplens.org/datasets/movielens/
[2] https://grouplens.org/datasets/movielens/
[3] https://guoguibing.github.io/librec/datasets.html

$\gamma \in \{0.1, 0.2, \ldots, 1.0\}$, and have $\gamma = 0.1$ on all three datasets. We searched the best value of the weight parameter of the sampled items $\alpha \in \{0.1, 0.2, \ldots, 1.0\}$, and have $\alpha = 0.6$ on all the three datasets. We searched the best value of the regularization terms λ_u, $\lambda_v \in \{0.005, 0.01, 0.02, 0.05\}$, and have λ_u, $\lambda_v = 0.02$ on ML100K, ML1M and have λ_u, $\lambda_v = 0.005$ on Filmtrust. We use different values of the sampling parameter $\rho \in \{1, 2, 3\}$. We use different values of the privacy budget $\epsilon \in \{0.001, 0.1, 1, 3, 5, 10\}$. For all three datasets, we set the training to stop after 100 epochs.

5.2 Results

Overall Performance. (RQ1) We report the performance of the model PriExRec compared to two other baseline models in Table 2, we can have the following observations: (1) PriExRec outperforms the others on all datasets, especially significantly better on the Filmtrust dataset compared to FedRec and FedRec++, where it improves the MAE by 8.22% and 4.88% and the RMSE by 6.38% and 3.42%, respectively. This indicates that PriExRec has strong generalization capabilities across datasets of different sizes, making it particularly suitable for small-scale and highly sparse datasets. (2) Different ρ values have little impact on the recommendation performance of PriExRec, demonstrating the model's robustness to parameter variations. (3) PriExRec significantly outperforms the comparison algorithms in terms of MAE and achieves comparable performance in terms of RMSE. This demonstrates that PriExRec excels in both prediction accuracy and the reduction of average error, delivering more precise recommendations to users, and thereby ensuring superior overall predictive performance.

Performance of Different Modules. (RQ2) Fig. 2 illustrates the MAE and RMSE performance of each model across different datasets under different values of ρ. In the legend, different colors represent different configurations of the models. Specifically, *PriExRec_ avg_ fill* indicates that the user rating mean was used to replace the proposed item mean, while *PriExRec_ no_ compress_ 64* indicates that the federated feature compression mechanism was not deployed, with the feature vector dimension of 64. The observations are as follows: (1) As ρ increases, the MAE and RMSE of each model show minimal variation, indicating that the models are robust to different noise levels. This stability is crucial for recommendation systems, as it ensures consistent performance even under privacy-preserving mechanisms, such as noise perturbation. (2) Comparisons between different modules indicate that the PriExRec model, which integrates multiple strategies, consistently outperforms individual modules. This suggests that the integration of various strategies significantly enhances model performance, giving PriExRec higher stability and prediction accuracy when handling different datasets and noise levels. Overall, this demonstrates its strong comprehensive capability and adaptability.

Performance of Different Parameters. (RQ3) We compare the impact of different hyperparameters on model performance in Fig. 3. From Fig. 3a, we observe that when the weight for the imputed ratings is fixed, the effect of different privacy budgets ϵ on model performance is not significant. In some

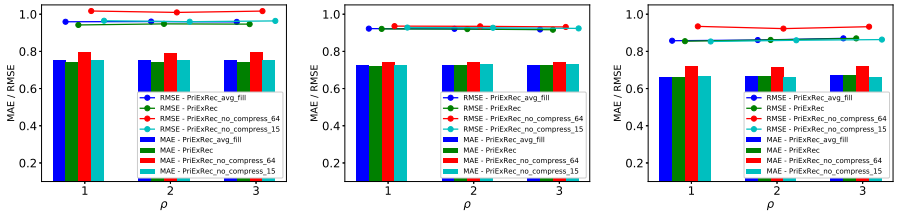
Table 2: Recommendation performance of FedRec, FedRec++ and our PriExRec model with different values of ρ. Note that we copy the results of FedRec and FedRec++ on ML100K and ML1M from [10] for comparison.

Datasets	Methods	MAE ↓	RMSE ↓	ρ
ML100K	FedRec	0.7441	0.9432	1
	FedRec++	0.7417	**0.9422**	
	PriExRec	**0.7388**	**0.9422**	
	FedRec	0.7445	0.9431	2
	FedRec++	0.7422	**0.9430**	
	PriExRec	**0.7415**	0.9480	
	FedRec	0.7447	0.9431	3
	FedRec++	**0.7416**	**0.9421**	
	PriExRec	**0.7416**	0.9466	
ML1M	FedRec	0.7217	0.9129	1
	FedRec++	**0.7198**	0.9133	
	PriExRec	0.7210	**0.9128**	
	FedRec	0.7239	0.9152	2
	FedRec++	**0.7195**	**0.9109**	
	PriExRec	0.7235	0.9199	
	FedRec	0.7263	0.9178	3
	FedRec++	**0.7195**	**0.9109**	
	PriExRec	0.7244	0.9165	
Filmtrust	FedRec	0.7040	0.9366	1
	FedRec++	0.6996	0.9348	
	PriExRec	**0.6615**	**0.8549**	
	FedRec	0.7020	0.9366	2
	FedRec++	0.6986	0.9306	
	PriExRec	**0.6637**	**0.8626**	
	FedRec	0.6981	0.9294	3
	FedRec++	0.6974	0.9259	
	PriExRec	**0.6700**	**0.8700**	

cases, smaller values of ϵ even lead to better performance. This indicates that we can assign a very small privacy budget to each user, effectively protecting user data privacy while maintaining good model performance. In combination with the results shown in Fig. 3b, we find that the model performance fluctuates considerably across different values of α, and when the imputation ratio ρ is fixed, an increase in α tends to lead to a decrease in performance. This is because larger α values introduce more noise, which significantly affects the model's prediction accuracy, resulting in a decline in overall performance. This further suggests that moderate noise addition is crucial for balancing privacy protection and model performance, as excessive noise can have adverse effects.

Effectiveness in Defending against Membership Inference Attacks. (RQ4) To evaluate the extent to which the proposed model, PriEcRec, defends against server-side interactive membership inference attacks, we conducted simulated experiments using gradient inversion attacks to measure the model's defense capabilities. Specifically, the server reconstructs users' feature vectors using the collected gradients and the global item factor matrix, employing pseudo-inverse or other linear algebra methods. These feature vectors are then multiplied by the item factor matrix to predict users' ratings, thereby revealing users' private information.

To assess the effectiveness of these methods in defending against attacks, we used the average attack error (Attack Error) as the evaluation criterion. A



(a) ML100K (b) ML1M (c) Filmtrust

Fig. 2: MAE (Bar) and RMSE (Line) for Different Modules in our PriExRec with Different ρ Values on three datasets.

(a) Different values of ϵ (b) Different values of α

Fig. 3: Impact of Different Hyperparameters on our PriExRec's Performance with the ML100K dataset.

higher value of this metric indicates that the server's inferred user ratings are less accurate, meaning better protection of user privacy. The experimental results in Table 3 indicate that PriExRec significantly outperforms other methods in defending against the server's gradient inversion attacks.

Table 3: Attack Error of Different Methods across Various Datasets and ρ Values.

Methods	**ML100K**			**ML1M**			**Filmtrust**		
	$\rho = 1$	$\rho = 2$	$\rho = 3$	$\rho = 1$	$\rho = 2$	$\rho = 3$	$\rho = 1$	$\rho = 2$	$\rho = 3$
FedRec_64	14.1184	14.0994	14.1088	14.9145	14.9194	14.9205	10.2114	10.2235	10.2116
FedRec++_64	14.0931	14.0954	14.1220	14.9220	14.9150	14.9307	10.1864	10.2066	10.2144
FedRec_15	14.1088	14.1132	14.0986	14.9135	14.9257	14.9245	10.2017	10.2149	10.2048
FedRec++_15	14.1145	14.1349	14.0841	14.9056	14.9203	14.8901	10.2235	10.2195	10.2182
PriExRec	**14.2395**	**14.2437**	**14.2432**	**14.9827**	**15.0054**	**14.9950**	**10.2243**	**10.2457**	**10.2312**

6 Conclusion

In this paper, we propose the PriEcRec model to defend against server-side membership inference attacks in federated learning while maintaining high recommendation performance. Experimental results show that PriEcRec signifi-

cantly increases attack error, making it harder for servers to infer user ratings accurately, thus enhancing privacy protection. Compared to other federated recommender systems offers significant defensive advantages across datasets and settings without compromising recommendation accuracy.

Acknowledgments. The work is partially supported by the National Key Research and Development Program of China (2024YFF0617702), the National Natural Science Foundation of China (Nos. U22A2025, 62402097, 62232007, U23A20309), the Joint Funds of Natural Science Foundation of Liaoning Province (2023-BSBA-132), and 111 Project (No. B16009).

References

1. Ammad-ud-din, M., Ivannikova, E., Khan, S.A., Oyomno, W., Fu, Q., Tan, K.E., Flanagan, A.: Federated collaborative filtering for privacy-preserving personalized recommendation system. CoRR **abs/1901.09888** (2019)
2. Anelli, V.W., Deldjoo, Y., Di Noia, T., Ferrara, A., Narducci, F.: How to put users in control of their data in federated top-n recommendation with learning to rank. In: Proceedings of the 36th Annual ACM Symposium on Applied Computing (2021)
3. Arcolezi, H.H., Couchot, J., al Bouna, B., Xiao, X.: Random sampling plus fake data: Multidimensional frequency estimates with local differential privacy. In: CIKM. pp. 47–57. ACM (2021)
4. Coluzzi, M., Brocco, A., Contu, P., Leidi, T.: A survey and comparison of consistent hashing algorithms. In: IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2023, Raleigh, NC, USA, April 23-25, 2023. pp. 346–348. IEEE (2023)
5. Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. Found. Trends Theor. Comput. Sci. **9**(3-4), 211–407 (2014)
6. Feng, Z., Yan, C., Jiang, R., Xu, X., Zhang, X., Zhou, X., Dou, W., Qi, L.: CHDAER: consistent hashing-based data allocation for efficient recommendation in edge environment. In: Serra, E., Spezzano, F. (eds.) Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM 2024, Boise, ID, USA, October 21-25, 2024. pp. 622–631. ACM (2024)
7. Guo, G., Zhang, J., Yorke-Smith, N.: A novel bayesian similarity measure for recommender systems. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI). pp. 2619–2625 (2013)
8. Harper, F.M., Konstan, J.A.: The movielens datasets. ACM Transactions on Interactive Intelligent Systems p. 1–19 (Jan 2016)
9. Jiang, Y., Zhou, Y., Wu, D., Li, C., Wang, Y.: On the detection of shilling attacks in federated collaborative filtering. In: International Symposium on Reliable Distributed Systems, SRDS 2020, Shanghai, China, September 21-24, 2020. pp. 185–194. IEEE (2020)
10. Liang, F., Pan, W., Ming, Z.: Fedrec++: Lossless federated recommendation with explicit feedback. In: AAAI. pp. 4224–4231. AAAI Press (2021)
11. Lin, G., Liang, F., Pan, W., Ming, Z.: Fedrec: Federated recommendation with explicit feedback. IEEE Intell. Syst. **36**(5), 21–30 (2021)
12. Lin, Z., Pan, W., Ming, Z.: Fr-fmss: Federated recommendation via fake marks and secret sharing. In: Fifteenth ACM Conference on Recommender Systems (2021)

13. Liu, X., Chen, Y., Pang, S.: Defending against membership inference attack for counterfactual federated recommendation with differentially private representation learning. IEEE Trans. Inf. Forensics Secur. **19**, 8037–8051 (2024)
14. Meng, X., Du, Y., Zhang, Y., Han, X.: A survey of context-aware recommender systems: From an evaluation perspective. IEEE Trans. Knowl. Data Eng. **35**(7), 6575–6594 (2023)
15. Minto, L., Haller, M., Livshits, B., Haddadi, H.: Stronger privacy for federated collaborative filtering with implicit feedback. In: RecSys '21: Fifteenth ACM Conference on Recommender Systems, Amsterdam, The Netherlands, 27 September 2021 - 1 October 2021. pp. 342–350. ACM (2021)
16. Mnih, A., Salakhutdinov, R.: Probabilistic matrix factorization. Neural Information Processing Systems (2007)
17. Rong, D., Ye, S., Zhao, R., Yuen, H.N., Chen, J., He, Q.: Fedrecattack: Model poisoning attack to federated recommendation. In: 38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022. pp. 2643–2655. IEEE (2022)
18. Shin, H., Kim, S., Shin, J., Xiao, X.: Privacy enhanced matrix factorization for recommendation with local differential privacy. IEEE Trans. Knowl. Data Eng. **30**(9), 1770–1782 (2018)
19. Wang, T., Blocki, J., Li, N., Jha, S.: Locally differentially private protocols for frequency estimation. USENIX Security Symposium,USENIX Security Symposium (2017)
20. Wu, C., Wu, F., Cao, Y., Huang, Y., Xie, X.: Fedgnn: Federated graph neural network for privacy-preserving recommendation. Cornell University - arXiv,Cornell University - arXiv (2021)
21. Wu, C., Wu, F., Qi, T., Huang, Y., Xie, X.: Fedattack: Effective and covert poisoning attack on federated recommendation via hard sampling. In: Zhang, A., Rangwala, H. (eds.) KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022. pp. 4164–4172. ACM (2022)
22. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Trans. Intell. Syst. Technol. **10**(2), 12:1–12:19 (2019)
23. Yi, J., Wu, F., Zhu, B., Yao, J., Tao, Z., Sun, G., Xie, X.: Ua-fedrec: Untargeted attack on federated news recommendation. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023. pp. 5428–5438. ACM (2023)
24. Yu, Y., Liu, Q., Wu, L., Yu, R., Yu, S.L., Zhang, Z.: Untargeted attack against federated recommendation systems via poisonous item embeddings and the defense. In: AAAI. pp. 4854–4863. AAAI Press (2023)
25. Yuan, W., Yang, C., Nguyen, Q.V.H., Cui, L., He, T., Yin, H.: Interaction-level membership inference attack against federated recommender systems. In: Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023. pp. 1053–1062. ACM (2023)
26. Yuan, W., Yuan, S., Yang, C., Hung, N.Q.V., Yin, H.: Manipulating visually aware federated recommender systems and its countermeasures. ACM Trans. Inf. Syst. **42**(3), 64:1–64:26 (2024)
27. Zhang, J., Li, H., Rong, D., Zhao, Y., Chen, K., Shou, L.: Preventing the popular item embedding based attack in federated recommendations. In: 40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024. pp. 2179–2191. IEEE (2024)