arXiv:2204.01499v2 [cs.CR] 13 Oct 2022

# FedRecAttack: Model Poisoning Attack to Federated Recommendation

Dazhong Rong†, Shuai Ye‡, Ruoyan Zhao‡, Hon Ning Yuen†, Jianhai Chen*† , Qinming He†
†College of Computer Science and Technology, Zhejiang University, Hangzhou, China
‡Polytechnic Institute, Zhejiang University, Hangzhou, China
{rdz98, zjuyeh, ryanzh, yuenhn, chenjh919, hqm}@zju.edu.cn

***Abstract*—Federated Recommendation (FR) has received considerable popularity and attention in the past few years. In FR, for each user, its feature vector and interaction data are kept locally on its own client thus are private to others. Without the access to above information, most existing poisoning attacks against recommender systems or federated learning lose validity. Benifiting from this characteristic, FR is commonly considered fairly secured. However, we argue that there is still possible and necessary security improvement could be made in FR. To prove our opinion, in this paper we present FedRecAttack, a model poisoning attack to FR aiming to raise the exposure ratio of target items. In most recommendation scenarios, apart from private user-item interactions (*e.g.,* clicks, watches and purchases), some interactions are public (*e.g.,* likes, follows and comments). Motivated by this point, in FedRecAttack we make use of the public interactions to approximate users' feature vectors, thereby attacker can generate poisoned gradients accordingly and control malicious users to upload the poisoned gradients in a well-designed way. To evaluate the effectiveness and side effects of FedRecAttack, we conduct extensive experiments on three real-world datasets of different sizes from two completely different scenarios. Experimental results demonstrate that our proposed FedRecAttack achieves the state-of-the-art effectiveness while its side effects are negligible. Moreover, even with small proportion ($3\%$) of malicious users and small proportion ($1\%$) of public interactions, FedRecAttack remains highly effective, which reveals that FR is more vulnerable to attack than people commonly considered.**

***Index Terms*—Recommender System, Federated Recommendation, Federated Learning, Poisoning Attack**

## I. Introduction

Research in the field of recommender systems mainly focused on boosting recommendation accuracy in the past few years. Specifically, the studies can be roughly divided into three categories. The first stream is to construct more sophisticated structures of recommender models (*e.g.,* NCF [1], NAIS [2], NGCF [3] and LightGCN [4]). The second stream is to integrate more auxiliary information (*e.g.,* VBPR [5], CKE [6], MMGCN [7] and KGIN [8]). The third stream is to adapt recommender models to scenarios of more specific areas (*e.g.,* e-commerce [9], news [10], [11] and micro-videos [12]). Note that combining knowledge graph to improve model expressivity is the current research trend [13].

Up to recent years, the performance of recommender systems was already quite advanced. As recommender systems were widely used in diverse scenarios, researchers' attention shifted to the security of recommender systems. Security issues are relevant to everyone, as recommender systems play an important role in people's ways of thinking, judging and acquitting information. An attacker could make his specific target items to be popular if he gains control of a recommender system. In severe cases, attacker can even influence public opinion on social media and change the public's views or attitudes on specific heat events, resulting in changes in people's actual behaviors. Cambridge Analytica is a data analysis company which manipulates political public opinion. They capture user-item interactions on different social media and use the data to generate user profiles, then accurately recommend particular advertisements to specific users. As a result, the company has a direct and decisive impact on the results of American Presidential Election [14].

References [15]–[17] present data poisoning attacks against matrix factorization based, deep learning based and graph based recommender systems respectively. All of the above attacks highly rely on attacker's prior knowledge of user-item interactions to approximate the parameters of recommender model, thus attacker can control malicious users to generate fake interactions accordingly, resulting in the exposure ratio of target items being raised.

Meanwhile, people were also paying increasing attention to privacy protection. In traditional centralized recommender systems, servers store all historical user-item interactions and private profiles of millions of users, which contain vast amount of sensitive information. Therefore, once there is leakage in data on servers, there will be extremely serious negative impact on millions of users, enterprises and the society. For example, *Wired*, *The New York Times* and *The Observer* reported that Facebook had been repeatedly suffering from server data breaches in recent years. In the most severe case, data of more than 500 million users was exposed, causing pensions of 5 billion dollars to Facebook [18]. In 2016, European Union promulgated the General Data Protection Regulation (GDPR), which prohibited commercial companies from collecting, processing or exchanging user data without the corresponding user's permission [19]. Afterwards, the United States, China, Egypt and Brazil also formulated and introduced similar regulations to protect users' privacy [20].

Federated recommendation (FR) is considered as a solution to recommend items under circumstances of protecting users' privacy. In FR, recommender systems are trained in

* Corresponding author

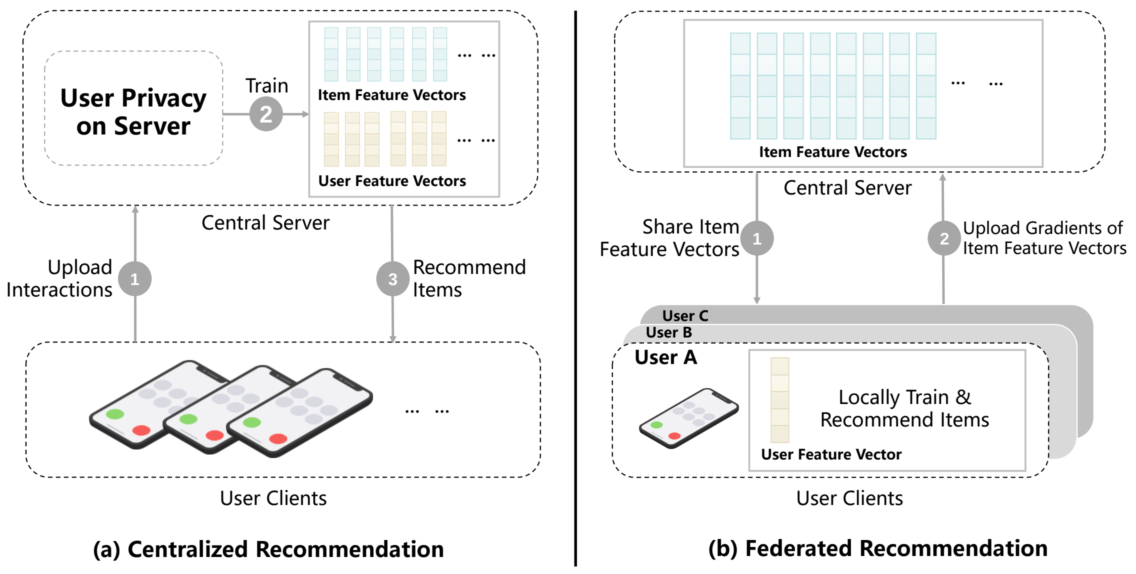

Fig. 1. Centralized Recommendation v.s. Federated Recommendation

federated scenarios. As [21] proposed the first framework of FR, other researches followed their work and raised extended versions [22]–[27] in a short time.

The strengths of FR include not only protecting users' privacy, but also being immune to existing attacks against recommender systems. As shown in Fig. 1, different from centralized recommendation, in FR for each user its interaction data and feature vector are kept locally on its own client thus are private to others (including central server). This means that it is impractical for attacker to access users' interaction data. Because of the dependency on users' interaction data, existing data poisoning attacks lose validity in FR. Similarly, existing model poisoning attacks against federated learning [28]–[30] are also inapplicable in FR. The reason is, all of these model poisoning attacks require full access to model parameters. However, each user's feature vector is also private in FR. Based on the above evidences, people consider FR fairly secured, therefore it could be stated that attacking federated recommender systems is a highly challenging task.

Nevertheless, we believe that there is still possible and necessary improvement to the security of FR. On numerous social platforms (*e.g.,* twitter, facebook and instagram), apart from private interactions (*e.g.,* clicks, watches and purchases), some interactions are public (*e.g.,* likes, follows and comments). Although the proportion of public interactions is usually small, it could still make significant contributions to attacks with the help of shared model parameters in FR. Inspired by the above hypothesis, we propose FedRecAttack. In FedRecAttack, attacker can approximate users' feature matrix with public interactions and the shared parameters. Subsequently, attacker can generate poisoned gradients accordingly, and control malicious users to upload them in a well-designed way. The **key challenges** of our work are summarized below.

**Conducting attacks with only small proportion of public interactions.** Existing data poisoning attacks against recommender systems highly rely on users' interaction data, which is inaccessible for attacker in FR. Even with the help of public interactions, these attacks remain ineffective, due to the low proportion. In FedRecAttack, we need to devise an ingenious way to make adequate use of public interactions.

**Conducting attacks without access to users' feature matrix.** In FR, model parameters can be separated into two parts, one is shared while the other is private. The shared model parameters are maintained on central server. In contrast, the private model parameters are kept locally on each user client. Existing model poisoning attacks agaist federated learning require full access to both parts, which is obviously impractical. In FedRecAttack, we need to validly approximate the private model parameters (*i.e.,* users' feature matrix) with the help of public interactions and the shared model parameters.

**Conducting attacks by stealth.** To the best of our knowledge, PipAttack [31] is the first proposed attack method against FR. However, there are two drawbacks of PipAttack. The first is it requires attacker's prior knowledge of side information about items' popularity, which is not always accessible in FR. The second is it leads to significant degradation of recommendation accuracy, indicating the attack is highly detectable. To ensure both the stealthiness and the effectiveness of FedRecAttack, we need to achieve attacker's goal without leading significant side effects.

In this paper, we attempt to explore the possibility of attacking federated recommender systems. We devise an ingenious way to approximate users' feature matrix and use it to compute poisoned gradients. Additionally, we design a sophisticated method to upload poisoned gradients under certain limitations. Our attack can poison the recommender model effectively, while the side effects of our attack are negligible, which demonstrates the vulnerability of FR.

The **key contributions** of our work can be summarized as following:

- We propose FedRecAttack, the first open source model poisoning attack method against FR.[1] As evaluated on three real-world datasets of different sizes from two completely different scenarios, our FedRecAttack reaches the highest effectiveness among all baseline attacks.
- We compare the effectiveness of FedRecAttack with that of state-of-the-art data poisoning and model poisoning attacks. The effectiveness of FedRecAttack sets the new state-of-the-art when the proportion of malicious users is small.
- We explore the side effects of FedRecAttack. The experimental results demonstrate that the side effects of FedRecAttack are the slightest in all model poisoning attacks, which means FedRecAttack is difficult to be detected.

## II. Related Work

### A. Recommender Systems

Collaborative filtering recommender systems use user-item interactions (*e.g.,* clicks or favors) to model the latent features of users and items. Training and optimizing the recommender model with historical user-item interactions, empowers the recommender system to predict rating score between uninteracted user-item pair, describing how much the user likes the item. Previous researches on Recommender Systems (RS) can be categorized into three groups: i) Matrix factorization based RS [32]–[35], which simply take the dot product of user's and item's feature vectors as the predicted rating score between the user and the item. ii) Deep learning based RS [1], [36]–[40], which use deep learning models to make the prediction of rating scores with high level of non-linearities. iii) Graph based RS [3], [4], which consider user-item interactions as the edges in a bipartite graph and try to exploit the high-order connectivity by performing embedding propagation.

All these researches designed elaborate model structures and boosted the recommendation performance.

### B. Attacks against Recommender Systems

Fueled by the technical maturity, RS have become more widely used in diverse scenarios continuously, along with the number of researchers focusing on security issues of RS growing. Existing studies show that current RS are vulnerable under certain circumstances as researchers proposed many effective attack methods against RS. One line of the attacks aim to cause loss of the recommendation accuracy and undermine the validity of target model. Another line of the attacks aim to make specific items to be recommended to as many users as possible.

In this paper, we focus on the latter attacks as they are more stealthy and more harmful. Data poisoning attack is the main form of the attacks, which proceeds by injecting fake users and controlling them to interact with some carefully selected items, leading to training data being poisoned. The recommender model trained on the poisoned data will predict abnormally high rating scores of the target items to many users. References [15]–[17], [41] proposed data poisoning attacks against the above three classes of RS. Nevertheless, all of these attacks have limited effects and highly rely on attacker's prior knowledge (*i.e.,* historical interactions). More specifically, in [15], [17] attacker was assumed to have access to full interactions, and in [16], [41] attacker was assumed to have access to partial interactions (more than 20%). These assumptions are often impractical in real-world scenarios.

[1]Our implementation is available at https://github.com/rdz98/FedRecAttack

### C. Federated Recommendation

Federated Recommendation (FR) is considered to be immune to such attacks. Fig. 1 shows the differences between the architectures of traditional centralized recommendation and FR. In FR, for each user, its interaction data is kept locally on its own client. As mentioned above, attacker's prior knowledge about historical interactions is critical to the attacks, which means that all these attacks are invalid in FR.

Due to the advantages of protecting users' privacy and immunity from attacks, FR has become a hot research topic. As [21] proposed the first framework of FR with implicit feedback, subsequent studies have sprung up. In [22], the deficiency that explicit feedback is inapplicable in FR was remedied by randomly sampling the items which user did not interact with. Based on [22], [23] proposed a lossless federated recommender system, which used denoising clients to collect the gradients with noise from ordinary clients. Reference [24] applied graph neural network to FR, and achieved fairly good recommendation accuracy without losing the protection of users' privacy.

### D. Attacks against Federated Learning

Since the advent of Federated Learning (FL), the security issues of FL have been of concern to researchers, and a large number of attacks against FL have been proposed. Model poisoning attack is the main form of these attacks, which proceeds by controlling malicious users to upload poisoned gradients. References [28]–[30] presented model poisoning attacks against FL in classification tasks. FR is a special case of FL. However existing attacks against FL are invalid in FR because of the following reasons: i) Most existing attacks are designed in classification tasks, but in FR the learning objective is completely different from that in classification tasks. ii) Existing attacks rely on attacker's full access to model parameters, but in FR each user's feature vector is kept hidden to others.

### E. Attacks against Federated Recommendation

The attack methods against FR are still under-explored. Reference [31] proposed PipAttack, the first framework of model poisoning attack against FR. Its experimental results show that PipAttack is effective in FR. However, there are two disadvantages of PipAttack: i) PipAttack causes significant degradation of recommendation accuracy, which increases the

TABLE I
LIST OF IMPORTANT NOTATIONS

| | |
|---|---|
| $\boldsymbol{\mathcal{D}}$ | all interactions |
| $\boldsymbol{\mathcal{D}}'$ | public interactions |
| $\boldsymbol{\mathcal{V}}_i^+$ | items user $u_i$ has interacted with |
| $\boldsymbol{\mathcal{V}}_i^-$ | items user $u_i$ has not interacted with |
| $\boldsymbol{\mathcal{V}}_i^{-\prime}$ | negative items for $u_i$ |
| $\boldsymbol{\mathcal{V}}_i^{-\prime\prime}$ | items user $u_i$ has not publicly interact with |
| $\boldsymbol{\mathcal{V}}_i^{rec}$ | items in $\boldsymbol{\mathcal{V}}_i^-$ with top-$K$ predicted scores for user $u_i$ |
| $\boldsymbol{\mathcal{V}}_i^{rec\prime}$ | items in $\boldsymbol{\mathcal{V}}_i^{-\prime\prime}$ with top-$K$ predicted scores for user $u_i$ |
| $\boldsymbol{\mathcal{V}}^{tar}$ | target items |
| $\boldsymbol{\mathcal{U}}^t$ | selected benign users in the $t$-th iteration |
| $\boldsymbol{\mathcal{U}}_m^t$ | selected malicious users in the $t$-th iteration |
| $\nabla \mathbf{V}_i^t$ | the gradients of $\mathbf{V}$ uploaded by user $u_i$ in the $t$-th iteration |
| $\nabla \tilde{\mathbf{V}}_i^t$ | $\nabla \mathbf{V}_i^t$ when user $u_i$ is malicious |
| $\nabla \tilde{\mathbf{V}}^t$ | the sum of poisoned gradients of $\mathbf{V}$ in the $t$-th iteration |
| $\nabla \mathbf{v}_{ij}^t$ | the $j$-th row of $\nabla \mathbf{V}_i^t$ |
| $\nabla \tilde{\mathbf{v}}_{ij}^t$ | the $j$-th row of $\nabla \tilde{\mathbf{V}}_i^t$ |
| $\nabla \tilde{\mathbf{v}}_j^t$ | the $j$-th row of $\nabla \tilde{\mathbf{V}}^t$ |
| $k$ | dimension of feature vectors |
| $\mu$ | noise scale |
| $\eta$ | learning rate |
| $\xi$ | the proportion of public interactions |
| $\rho$ | the proportion of malicious users |
| $\kappa$ | the maximum number of non-zero rows in uploaded gradients |
| $C$ | the maximum $\ell_2$-norm of rows in uploaded gradients |

probability of being detected. ii) PipAttack demands a large proportion (10%) of malicious users to achieve its effectiveness, raising the attack costs to a fairly unaffordable extent.

Although both of the above characteristics imply that it is still difficult to cause serious damages to current federated recommender systems, we consider that there is still necessary improvement to the security of FR. To prove that it is possible to carry out attacks in FR with negligible side effects and small proportion of malicious users, we propose FedRecAttack.

## III. Problem Formulation

In this section, we first formally define the base recommender and the framework of FR. Then we introduce the fundamental settings of our attack.

### A. Base Recommender

We define a target recommender system to contain a set of users $\boldsymbol{\mathcal{U}} = \{u_1, u_2, \dots, u_n\}$ and a set of items $\boldsymbol{\mathcal{V}} = \{v_1, v_2, \dots, v_m\}$, where $n$ and $m$ are the numbers of users and items respectively. Training dataset $\boldsymbol{\mathcal{D}} \subseteq \boldsymbol{\mathcal{U}} \times \boldsymbol{\mathcal{V}}$ is composed of implicit feedback tuples which represent user-item interactions (*e.g.,* clicked, watched). Tuple $(u_i, v_j) \in \boldsymbol{\mathcal{D}}$ indicates the interaction between user $u_i$ and item $v_j$. For each user $u_i$, we also define $\boldsymbol{\mathcal{V}}_i^+ = \{v \in \boldsymbol{\mathcal{V}} \colon (u_i, v) \in \boldsymbol{\mathcal{D}}\}$ and $\boldsymbol{\mathcal{V}}_i^- = \{v \in \boldsymbol{\mathcal{V}} \colon (u_i, v) \notin \boldsymbol{\mathcal{D}}\}$ as the set of the items user $u_i$ has interacted with and that of those user $u_i$ has not, for convenience.

Let $\mathbf{U}\colon |\boldsymbol{\mathcal{U}}| \times k$ and $\mathbf{V}\colon |\boldsymbol{\mathcal{V}}| \times k$ denote the feature matrices with $k$ columns of users and items respectively. The feature vectors $\mathbf{u}_i$ and $\mathbf{v}_j$ are the $i$-th row in $\mathbf{U}$ and the $j$-th row in $\mathbf{V}$, describing the latent features of user $u_i$ and item $v_j$ respectively. The predicted rating score $\hat{x}_{ij}$, which represents how much user $u_i$ likes item $v_j$, is computed as following:

$$\hat{x}_{ij} = \Upsilon(\mathbf{u}_i, \mathbf{v}_j), \tag{1}$$

where $\Upsilon$ is the interaction function.

In matrix factorization based recommender models (*e.g.,* MF [42], FISM [33]), $\Upsilon$ is fixed. In deep learning based recommender models (*e.g.,* NCF [1], ONCF [37]), $\Upsilon$ is learnable. Various models are adopted in different scenarios of FR. It is worth mentioning that our attack is applicable to almost all recommender models as long as they are collaborative filtering.

To be specific, in this paper we adopt Matrix Factorization (MF), one of the most classic and widely used recommender models, as our base recommender. In MF, $\Upsilon$ is fixed to be dot product (*i.e.,* $\hat{x}_{ij} = \mathbf{u}_i \odot \mathbf{v}_j$, where $\odot$ denotes dot product).

We adopt Bayesian Personalized Ranking (BPR) [43] loss to train our base recommender. The BPR loss for each user is defined as following:

$$\mathcal{L}_i^{rec} = - \sum_{v_j \in \boldsymbol{\mathcal{V}}_i^+ \wedge v_k \in \boldsymbol{\mathcal{V}}_i^-} \ln \sigma(\hat{x}_{ijk}), \tag{2}$$

where $\sigma$ is the logistic sigmoid and $\hat{x}_{ijk} = \hat{x}_{ij} - \hat{x}_{ik}$. The model is optimized by minimizing the BPR loss for all users:

$$\mathcal{L}^{rec} = \sum_{u_i \in \boldsymbol{\mathcal{U}}} \mathcal{L}_i^{rec}. \tag{3}$$

In real-world scenarios, various loss functions are applied due to the feedback from users could be implicit or explicit. Once again note that our attack is applicable while other popular loss functions (*e.g.,* cross-entropy loss, hinge loss) are adopted.

### B. Framework of Federated Recommendation

We divide the parameters of our base recommender into users' feature matrix $\mathbf{U}$, items' feature matrix $\mathbf{V}$ and other learnable model parameters (denoted by $\mathbf{\Theta}$). If the interaction function $\Upsilon$ is fixed, $\mathbf{\Theta}$ is an empty set. If $\Upsilon$ is learnable through a deep neural network, $\mathbf{\Theta}$ is the set of the parameters in the neural network.

In FR there is a central server and a large amount of user clients. The shared parameters $\mathbf{V}$ and $\mathbf{\Theta}$ are maintained by the central server, while for each user $u_i$ its interaction data $\boldsymbol{\mathcal{V}}_i^+$ and its private parameter $\mathbf{u}_i$ (*i.e.,* the $i$-th row in $\mathbf{U}$) are kept locally on its own client, which is the main difference between federated and centralized recommendation.

In the beginning, the central server initializes the shared parameters $\mathbf{V}$ and $\mathbf{\Theta}$, meanwhile each user client $u_i$ initializes its private parameters $\mathbf{u}_i$. Moreover, to reduce the complexity of computation, each user client $u_i$ randomly samples a subset of negative items (denoted by $\boldsymbol{\mathcal{V}}_i^{-\prime}$) from $\boldsymbol{\mathcal{V}}_i^-$, and uses $\boldsymbol{\mathcal{V}}_i^{-\prime}$ instead of $\boldsymbol{\mathcal{V}}_i^-$. We can represent $\boldsymbol{\mathcal{V}}_i^+ = \{v_{i1}^+, v_{i2}^+, \dots, v_{ip}^+\}$ and

$\boldsymbol{\mathcal{V}}_i^{-'} = \{v_{i1}^-, v_{i2}^-, \ldots, v_{ip}^-\}$, where $p = |\boldsymbol{\mathcal{V}}_i^{-'}| = |\boldsymbol{\mathcal{V}}_i^+|$. The loss function for user $u_i$ can be simplified from Eq. (2) to:

$$\mathcal{L}_i^{rec} = -\sum_{(v_j, v_k) \in \boldsymbol{\mathcal{V}}_i} \ln \sigma(\hat{x}_{ijk}), \tag{4}$$

where $\boldsymbol{\mathcal{V}}_i = \{(v_{i1}^+, v_{i1}^-), (v_{i2}^+, v_{i2}^-), \ldots (v_{ip}^+, v_{ip}^-)\}$.

Subsequently, the training stage starts. In each training iteration, the central server randomly selects a batch of user clients $\boldsymbol{\mathcal{U}'} \subseteq \boldsymbol{\mathcal{U}}$ and sends the shared model parameters $\mathbf{V}$ and $\boldsymbol{\Theta}$ to them. For each selected user client $u_i \in \boldsymbol{\mathcal{U}'}$, with its loss $\mathcal{L}_i^{rec}$ computed, it derives the gradients $\nabla \mathbf{V}_i$, $\nabla \boldsymbol{\Theta}_i$ and $\nabla \mathbf{u}_i$ of parameters $\mathbf{V}$, $\boldsymbol{\Theta}$ and $\mathbf{u}_i$ respectively. To strongly protect users' sensitive data, as in [44], each selected user client $u_i$ adds some noise to both $\nabla \mathbf{V}_i$ and $\nabla \boldsymbol{\Theta}_i$:

$$\begin{aligned} \nabla \mathbf{V}_i &\leftarrow \nabla \mathbf{V}_i + \mathcal{N}(\mathbf{0}, \mu^2 C^2 \mathbf{I}), \\ \nabla \boldsymbol{\Theta}_i &\leftarrow \nabla \boldsymbol{\Theta}_i + \mathcal{N}(\mathbf{0}, \mu^2 C^2 \mathbf{I}), \end{aligned} \tag{5}$$

where $\mathcal{N}$, $\mu$ and $C$ denote normal distribution, the noise scale and the $\ell_2$-norm bound of gradients, respectively. Afterwards, each selected user client $u_i$ uploads $\nabla \mathbf{V}_i$ and $\nabla \boldsymbol{\Theta}_i$ to the central server and updates its private model parameter $\mathbf{u}_i$ locally as:

$$\mathbf{u}_i \leftarrow \mathbf{u}_i - \eta \cdot \nabla \mathbf{u}_i, \tag{6}$$

where $\eta$ is the learning rate. After collecting the gradients from all user clients in $\boldsymbol{\mathcal{U}'}$, the central server updates $\mathbf{V}$ and $\boldsymbol{\Theta}$ batchly:

$$\begin{aligned} \mathbf{V} &\leftarrow \mathbf{V} - \eta \cdot \sum_{u_i \in \boldsymbol{\mathcal{U}'}} \nabla \mathbf{V}_i, \\ \boldsymbol{\Theta} &\leftarrow \boldsymbol{\Theta} - \eta \cdot \sum_{u_i \in \boldsymbol{\mathcal{U}'}} \nabla \boldsymbol{\Theta}_i. \end{aligned} \tag{7}$$

Throughout the training stage, all users' privacy is well protected. As for a user client $u_i$, through its uploaded gradients, the central server can only extract the knowledge of whether an item $v$ is in $(\boldsymbol{\mathcal{V}}_i^+ \cup \boldsymbol{\mathcal{V}}_i^{-'})$, but not the knowledge of whether an item $v$ is in $\boldsymbol{\mathcal{V}}_i^+$.

### C. Fundamental Settings of Attack

**Attacker's Goal** For each user $u_i$, we suppose the recommender system recommends $K$ items in $\boldsymbol{\mathcal{V}}_i^-$ with the top-$K$ predicted scores. Let $\boldsymbol{\mathcal{V}}_i^{rec}$ denote the set of the $K$ items recommended for $u_i$, and let $\boldsymbol{\mathcal{V}}^{tar}$ denote the set of the target items. The exposure ratio at $K$ [31] of the target items is defined as:

$$\text{ER@K} = \frac{1}{|\boldsymbol{\mathcal{U}}|} \sum_{u_i \in \boldsymbol{\mathcal{U}}} \frac{|\boldsymbol{\mathcal{V}}^{tar} \wedge \boldsymbol{\mathcal{V}}_i^{rec}|}{|\boldsymbol{\mathcal{V}}^{tar} \wedge \boldsymbol{\mathcal{V}}_i^-|}. \tag{8}$$

We adopt ER@K as the metric to evaluate the effectiveness of our attack. Attacker's goal is to make the target items appear in the top-$K$ recommendation lists of as many users as possible. In other words, attacker aims to raise ER@K as high as possible.

**Attacker's Prior Knowledge** In FR, each user' feature vectors and historical interactions are private to others (so as to attacker). However, as mentioned in the previous section, in many cases a small part (no more than 5%) of the whole interactions is publicly available. Our work attempts to exploit these few public interactions for attacker to estimate the overall distribution of all users' actual feature vectors (*i.e.,* $\mathbf{U}$). Hence, based on the estimation of $\mathbf{U}$, attacker can compute the poisoned gradients of $\mathbf{V}$ and control the malicious user clients to upload the poisoned gradients to the central server. Following the common settings of FR, we assume that attacker knows the model structure and some hyper parameters (*e.g.,* learning rate $\eta$) used in the federated system. Due to the fact that central server shares the model parameter $\mathbf{V}$ and $\boldsymbol{\Theta}$ at the beginning of each iteration, once any malicious user client is selected to participate in training, attacker knows the current value of $\mathbf{V}$ and $\boldsymbol{\Theta}$. In addition, we assume attacker knows the public part of interactions. Let $\xi$ denote the propotion of the public interactions. In our experimental section, we explore the effectiveness of our attack in the case of $\xi = 1\%, 2\%, 3\%, 5\%, 10\%$ respectively. Note that all other information (*e.g.,* any benign user's feature vector) is unknown to attacker.

**Limitations on Attacker** Considering the circumstances in real scenarios, our attack is conducted under certain restrictions, including:

- **The proportion of malicious users, denoted as $\rho$.** Obviously, the more number of malicious users, the more effective the attack will be. If there is no restriction on $\rho$, attacker can inject a huge amount of malicious users and simply control them to interact with target items, resulting in rising the popularity of target items easily. However, too many malicious users will lead to an unaffordable cost of the attack. To make our work more challenging and practical, we restrict $\rho \leq 10\%$. In our experimental section, we explore our attack effectiveness in the case of $\rho = 1\%, 2\%, 3\%, 5\%, 10\%$ respectively.
- **The maximum number of non-zero rows in $\nabla \mathbf{V}_i$, denoted as $\kappa$.** A non-zero row means a row containing at least one non-zero element. The more number of non-zero rows in the uploaded poisoned gradients, the more powerful the gradients will be. If the number is unlimited, the attack is easy to be effective, because each malicious user client $u_i$ could upload $\nabla \mathbf{V}_i$ composed entirely of non-zero rows, which would affect all items' feature vectors. However, an excessive number of non-zero rows in $\nabla \mathbf{V}_i$ will lead to the attack being detected. To limit malicious users to behave like benign users, in our experiments we set $\kappa = 20, 40, 60, 80, 100$ respectively.
- **The maximum $\ell_2$-norm of rows in $\nabla \mathbf{V}_i$, denoted as $C$.** If there is no limit on the sizes of rows in $\nabla \mathbf{V}_i$, malicious users can easily poison items' feature vectors by uploading extremely large gradients, which will also lead to the attack being detected. Therefore, we limit the $\ell_2$-norm of each row in $\nabla \mathbf{V}_i$ not to be higher than $C$.

## IV. Our Attack: FedRecAttack

In this section, we present our FedRecAttack, a method of model poisoning attack in FR scenarios. In FedRecAttack, at-

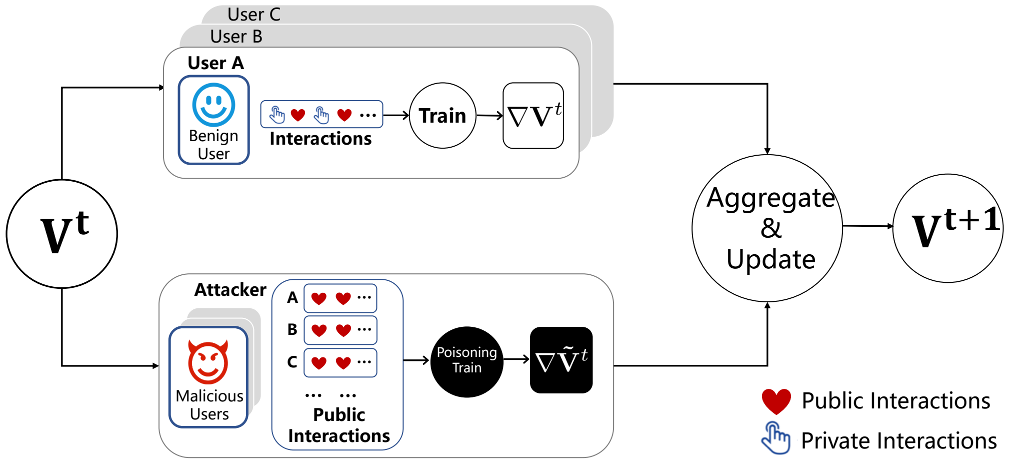

Fig. 2. Diagram of FedRecAttack

tacker first computes the poisoning gradients of items' feature matrix $\mathbf{V}$ with the access to the shared model parameters and the prior knowledge of the public interactions. Then attacker controls malicious clients to upload the poisoning gradients in a well-designed way.

Note that when the recommender is deep learning based, poisoning the learnable interaction function $\Upsilon$ is possibly a simpler and more effective attack method. However this method is not generic when the base recommender is matrix factorization based, because in this case the interaction function $\Upsilon$ is fixed. Therefore, to ensure the generality of our attack, in FedRecAttack we consider to poison items' feature matrix $\mathbf{V}$ only.

## A. *Solving the Optimization Problem*

**Formulate the Attack as an Optimization Problem** Let $T$ denote the total number of training iterations. Let $\boldsymbol{\mathcal{U}}_m^t$ and $\nabla\tilde{\mathbf{V}}^t$ denote the set of selected malicious users and the sum of thier uploaded gradients in the $t$-th iteration, respectively. Let $\mathbf{U}^t$, $\mathbf{V}^t$ and $\mathbf{\Theta}^t$ denote the parameters $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{\Theta}$ at the end of the $t$-th iteration, respectively. Then we can formally define our attack as an optimization problem:

$$\max_{\{\nabla\tilde{\mathbf{V}}^1,\nabla\tilde{\mathbf{V}}^2,\cdots,\nabla\tilde{\mathbf{V}}^T\}} \quad \mathrm{ER@K}(\mathbf{U}^t,\mathbf{V}^t,\mathbf{\Theta}^t).$$
$$\underset{t\in\{1,2,\cdots,T\},u_i\in\boldsymbol{\mathcal{U}}_m^t}{s.t.} \quad \begin{cases} \|\nabla\mathbf{v}_{ij}^t\|_2 \le C, & (\forall v_j\in\boldsymbol{\mathcal{V}}) \\ \sum_{v_j\in\boldsymbol{\mathcal{V}}} \delta(\nabla\mathbf{v}_{ij}^t) \le \kappa, \end{cases} \tag{9}$$

where $\nabla\mathbf{v}_{ij}^t$ denotes the gradients of item $v_j$ uploaded by user client $u_i$ in the $t$-th iteration, and $\delta$ is a function as:

$$\delta(\mathbf{v}) = \begin{cases} 1, & \|(v)\|_2 > 0 \\ 0, & \|(v)\|_2 = 0. \end{cases} \tag{10}$$

**Simplify the Optimization Problem** Let $\boldsymbol{\mathcal{U}}^t$ and $\nabla\mathbf{V}_i^t$ denote the set of selected benign users and the uploaded gradients from user client $u_i$ in the $t$-th iteration respectively. Following Eq. (7), we have:

$$\mathbf{V}^{t+1} = \mathbf{V}^t - \eta\cdot(\nabla\tilde{\mathbf{V}}^{t+1} + \sum_{u_i\in\boldsymbol{\mathcal{U}}^t} \nabla\mathbf{V}_i^{t+1}). \tag{11}$$

From the above equation, we can conclude that:

1) $\mathbf{V}^{t+1}$ depends on both $\nabla\tilde{\mathbf{V}}^{t+1}$ and $\nabla\mathbf{V}_i^{t+1}$.
2) $\nabla\mathbf{V}_i^{t+1}$ denotes the gradients of $\mathbf{V}^t$, so $\nabla\mathbf{V}_i^{t+1}$ also depends on $\nabla\tilde{\mathbf{V}}^t$ and $\nabla\mathbf{V}_i^t$.
3) By combining (1) and (2) recursively, $\mathbf{V}^{t+1}$ depends on $\{\nabla\tilde{\mathbf{V}}^{t+1},\nabla\tilde{\mathbf{V}}^t,\cdots,\nabla\tilde{\mathbf{V}}^1\}$.

In summary, the poisoned gradients in the $t$-th iteration (*i.e.,* $\nabla\tilde{\mathbf{V}}^t$) have influences on all subsequent iterations. Because attacker has no access to benign users' data, attacker is unable to compute $\nabla\mathbf{V}_i^{t+1}$. Therefore, it is impossible for attacker to predict the influences of $\nabla\tilde{\mathbf{V}}^t$ on subsequent learning. Due to the above reason, finding the global optimal solution of our optimization problem is difficult. We have to use a greedy strategy to simplify the problem. We try to find the current optimal solution instead of the global one, without the consideration of gradients uploaded by benign users. More specifically, in the $t$-th iteration, we need to solve the optimization problem as following:

$$\max_{\nabla\tilde{\mathbf{V}}^t} \quad \mathrm{ER@K}(\mathbf{U}^{t-1},\mathbf{V}^{t-1}-\eta\nabla\tilde{\mathbf{V}}^t,\mathbf{\Theta}^{t-1}).$$
$$\underset{u_i\in\boldsymbol{\mathcal{U}}_m^t}{s.t.} \quad \begin{cases} \|\nabla\mathbf{v}_{ij}^t\|_2 \le C, & (\forall v_j\in\boldsymbol{\mathcal{V}}) \\ \sum_{v_j\in\boldsymbol{\mathcal{V}}} \delta(\nabla\mathbf{v}_{ij}) \le \kappa. \end{cases} \tag{12}$$

**Approximate the Exposure Ratio** The exposure ratio (*i.e.,* ER@K) is a discontinuous and non-differentiable function of predicted scores, thus optimizing it by gradient descent is impossible. It is still difficult to solve the simplified optimization problem. To address the challenge, we design a continuous loss function to indicate ER@K of target items. For each benign user $u_i$, we define its loss function as:

$$\mathcal{L}_i^{atk} = \sum_{v_t \in \boldsymbol{\mathcal{V}}^{tar} \wedge (u_i, v_t) \notin \boldsymbol{\mathcal{D}}} g\Big( \min_{v_j \in \boldsymbol{\mathcal{V}}_i^{rec} \wedge v_j \notin \boldsymbol{\mathcal{V}}^{tar}} \{\hat{x}_{ij}\} - \hat{x}_{it} \Big), \tag{13}$$

where the function $g$ is defined as:

$$g(x) = \begin{cases} x, & x \geq 0 \\ e^x - 1, & x < 0. \end{cases} \tag{14}$$

Let $\boldsymbol{\mathcal{D}}' \subseteq \boldsymbol{\mathcal{D}}$ ($|\boldsymbol{\mathcal{D}}'| \leq \xi |\boldsymbol{\mathcal{D}}|$) denote the set of public interactions which is accessible to attacker. Then we can define $\boldsymbol{\mathcal{V}}_i^{-\prime\prime} = \{v \in \boldsymbol{\mathcal{V}} \colon (u_i, v) \notin \boldsymbol{\mathcal{D}}'\}$, and use $\boldsymbol{\mathcal{V}}_i^{rec'}$ to denote the set of items with top-$K$ predicted scores for user $u_i$ in $\boldsymbol{\mathcal{V}}_i^{-\prime\prime}$. Due to the limitations on attacker's prior knowledge, $\boldsymbol{\mathcal{D}}$ and $\boldsymbol{\mathcal{V}}_i^{rec}$ are hidden from attacker, thus we have to approximate them with $\boldsymbol{\mathcal{D}}'$ and $\boldsymbol{\mathcal{V}}_i^{rec'}$ respectively. The loss function for each user is now transformed to be:

$$\mathcal{L}_i^{atk} = \sum_{v_t \in \boldsymbol{\mathcal{V}}^{tar} \wedge (u_i, v_t) \notin \boldsymbol{\mathcal{D}}'} g\Big( \min_{v_j \in \boldsymbol{\mathcal{V}}_i^{rec'} \wedge v_j \notin \boldsymbol{\mathcal{V}}^{tar}} \{\hat{x}_{ij}\} - \hat{x}_{it} \Big), \tag{15}$$

The loss function for all benign users is defined as:

$$\mathcal{L}^{atk} = \sum_{u_i \in \boldsymbol{\mathcal{U}}} \mathcal{L}_i^{atk}. \tag{16}$$

Now we can raise ER@K of target items by minimizing $\mathcal{L}^{atk}$.

**Approximate Users' Feature Matrix** Because $\mathcal{L}^{atk}$ depends on the private parameter $\mathbf{U}$, attacker is not able to compute $\mathcal{L}^{atk}$. To address this challenge, we propose an effective method to approximate $\mathbf{U}$ with the shared model parameters (*i.e.,* $\mathbf{V}$ and $\boldsymbol{\Theta}$) and the public interactions (*i.e.,* $\boldsymbol{\mathcal{D}}'$), which are accessible to attacker. Remember that the goal of training a recommender model is to optimize the model parameters $\{\mathbf{U}, \mathbf{V}, \boldsymbol{\Theta}\}$ to $\{\mathbf{U}^*, \mathbf{V}^*, \boldsymbol{\Theta}^*\}$, where:

$$\{\mathbf{U}^*, \mathbf{V}^*, \boldsymbol{\Theta}^*\} = \mathop{\arg\min}_{\{\mathbf{U}, \mathbf{V}, \boldsymbol{\Theta}\}} \{\mathcal{L}^{rec}(\mathbf{U}, \mathbf{V}, \boldsymbol{\Theta}; \boldsymbol{\mathcal{D}})\}. \tag{17}$$

We can further infer that:

$$\mathbf{U}^* = \mathop{\arg\min}_{\mathbf{U}} \{\mathcal{L}^{rec}(\mathbf{U}, \mathbf{V}^*, \boldsymbol{\Theta}^*; \boldsymbol{\mathcal{D}})\}. \tag{18}$$

Once again, considering the limitations on attacker's prior knowledge, we replace $\boldsymbol{\mathcal{D}}$ with $\boldsymbol{\mathcal{D}}'$, and approximate $\mathbf{U}^t$ as:

$$\mathbf{U}^t \approx \mathop{\arg\min}_{\mathbf{U}} \{\mathcal{L}^{rec}(\mathbf{U}, \mathbf{V}^t, \boldsymbol{\Theta}^t; \boldsymbol{\mathcal{D}}')\}, \tag{19}$$

Now we can approximate $\mathbf{U}^t$ by using stochastic gradient descent to minimize $\mathcal{L}^{rec}(\mathbf{U}, \mathbf{V}^t, \boldsymbol{\Theta}^t; \boldsymbol{\mathcal{D}}')$.

**Compute Poisoned Gradients** Finally, we clearly formulate our attack. In the $t$-th training iteration, we need to calculate the best poisoning gradients $\nabla\tilde{\mathbf{V}}^t$ to minimize $\mathcal{L}^{atk}(\mathbf{U}^{t-1}, \mathbf{V}^{t-1} - \eta\nabla\tilde{\mathbf{V}}^t, \boldsymbol{\Theta}^{t-1}; \boldsymbol{\mathcal{D}}')$. We compute $\nabla\tilde{\mathbf{V}}^t$ as following:

$$\nabla\tilde{\mathbf{V}}^t = \zeta \cdot \frac{\partial}{\partial \mathbf{V}} \mathcal{L}^{atk}(\mathbf{U}^{t-1}, \mathbf{V}^{t-1}, \boldsymbol{\Theta}^{t-1}; \boldsymbol{\mathcal{D}}'), \tag{20}$$

where $\zeta$ is the step size.

**Algorithm 1:** Steps of FedRecAttack in the $t$-th training iteration.

```
/* Assuming $p$ malicious users $u_{m_1}$, $u_{m_2}$,
   $\cdots$, $u_{m_p}$ are selected to participate in
   the $t$-th training interaction              */
Input  : the set of public interactions $\boldsymbol{\mathcal{D}}'$
         shared model parameters $\mathbf{V}^t$ and $\boldsymbol{\Theta}^t$
Output: poisoned gradients $\nabla\tilde{\mathbf{V}}^t_{m_1}$, $\nabla\tilde{\mathbf{V}}^t_{m_2}$, $\ldots$,
         $\nabla\tilde{\mathbf{V}}^t_{m_p}$ for corresponding malicious users to
         upload in the $t$-th iteration
1  Approximate $\mathbf{U}^t$ by Eq. (19);
2  Generate poisoned gradients $\nabla\tilde{\mathbf{V}}^t$ by Eq. (20);
3  Initialize empty set $\mathbf{G}$ of poisoned gradients.;
4  foreach $u_i$ in $\{u_{m_1}, u_{m_2}, \ldots, u_{m_p}\}$ do
5      if the first time for $u_i$ participating in training then
6          Initialize $\boldsymbol{\mathcal{V}}_i$ by Eq. (21)
7      end
8      Compute $\nabla\tilde{\mathbf{V}}^t_i$ by Eq. 23;
9      Control malicious user $u_i$ to upload $\nabla\tilde{\mathbf{V}}^t_i$;
10     Update $\nabla\tilde{\mathbf{V}}^t$ by Eq. 24;
11     $\mathbf{G} \leftarrow \mathbf{G} \cup \nabla\tilde{\mathbf{V}}^t_i$
12 end
13 return $\mathbf{G}$
```

## *B. Upload the Poisoned Gradients*

Due to the restrictions on Eq. (12), attacker can not upload the poisoned gradients $\nabla\tilde{\mathbf{V}}^t$ directly. Hence, we design an elaborate method to upload the poisoned gradients. More specifically, in the $t$-th training iteration, each selected malicious user $u_i \in \boldsymbol{\mathcal{U}}_m^t$ follows the three steps below.

**Compute $\boldsymbol{\mathcal{V}}_i$** If it is the first time user $u_i$ to participate in training, we compute $\boldsymbol{\mathcal{V}}_i$ as following:

$$\boldsymbol{\mathcal{V}}_i = \boldsymbol{\mathcal{V}}^{tar} \cup R(\nabla\tilde{\mathbf{V}}^t, \kappa - |\boldsymbol{\mathcal{V}}^{tar}|), \tag{21}$$

where $R(\nabla\tilde{\mathbf{V}}^t, \kappa - |\boldsymbol{\mathcal{V}}^{tar}|)$ is a function to stochasticly select $(\kappa - |\boldsymbol{\mathcal{V}}^{tar}|)$ items without replacement. The probability of item $v_j$ being selected is:

$$p(v_j) = \begin{cases} 0, & v_j \in \boldsymbol{\mathcal{V}}^{tar} \\ \dfrac{||\nabla\tilde{\mathbf{v}}_j^t||_2}{\sum\limits_{v_k \in \boldsymbol{\mathcal{V}} \setminus \boldsymbol{\mathcal{V}}^{tar}} ||\nabla\tilde{\mathbf{v}}_k^t||_2}, & v_j \notin \boldsymbol{\mathcal{V}}^{tar}, \end{cases} \tag{22}$$

where $\nabla\tilde{\mathbf{v}}_j^t$ indicates the $j$-th row in $\nabla\tilde{\mathbf{V}}^t$. The larger $\nabla\tilde{\mathbf{v}}_j^t$, the higher the probability of item $v_j$ being selected. Besides,we keep $\boldsymbol{\mathcal{V}}_i$ unchanged if it is not the first time user $u_i$ to participate in training.

**Compute $\nabla\tilde{\mathbf{V}}_i^t$** Let $\nabla\tilde{\mathbf{V}}_i^t$ denote the uploaded poisoned gradients from malicious user $u_i$, and $\nabla\tilde{\mathbf{v}}_{ij}^t$ denote the $j$-th row in $\nabla\tilde{\mathbf{V}}_i^t$. We compute $\nabla\tilde{\mathbf{v}}_{ij}^t$ as following:

$$\nabla\tilde{\mathbf{v}}_{ij}^t = \begin{cases} \mathbf{0}, & v_j \notin \boldsymbol{\mathcal{V}}_i \\ \nabla\tilde{\mathbf{v}}_j^t, & v_j \in \boldsymbol{\mathcal{V}}_i \wedge ||\nabla\tilde{\mathbf{v}}_j^t||_2 \le C \\ C \cdot \dfrac{\nabla\tilde{\mathbf{v}}_j^t}{||\nabla\tilde{\mathbf{v}}_j^t||_2}, & v_j \in \boldsymbol{\mathcal{V}}_i \wedge ||\nabla\tilde{\mathbf{v}}_j^t||_2 > C. \end{cases} \tag{23}$$

**Update $\nabla\tilde{\mathbf{V}}^t$** After the computation of $\nabla\tilde{\mathbf{V}}_i^t$, we update $\nabla\tilde{\mathbf{V}}^t$ as following:

$$\nabla\tilde{\mathbf{V}}^t \leftarrow \nabla\tilde{\mathbf{V}}^t - \nabla\tilde{\mathbf{V}}_i^t. \tag{24}$$

The complete procedure of FedRecAttack is illustrated in Algorithm 1.

## V. Experiments

In this section, firstly, we introduce our experimental settings. Secondly, we demonstrate the impact of the limitations on attacker in FedRecAttack. Thirdly, we compare the effectiveness of FedRecAttack with that of other attack methods. Forthly, we analyze the side effects of FedRecAttack on recommendation accuracy. And lastly, we conduct an ablation test to demonstrate the necessity of attacker's prior knowledge (*i.e.,* the publicly accessible interactions).

### A. Experimental Settings

**Datasets** We use three real-world datasets in two completely different scenarios (movie recommendation and game recommendation) for our experiments. The three datasets are: **MovieLens-100K**, **MovieLens-1M** [45] and **Steam-200K** [46]. Their sizes are shown in Table II. MovieLens is a dataset of users' ratings for movies. As a version of small size, MovieLens-100K involves $943$ users, $1,682$ movie items and $100,000$ user-item interactions. MovieLens-1M is an extended version involving $6,040$ users, $3,706$ items and $1,000,209$ interactions. Steam-200K is a dataset of users' behaviors (own and play) on Steam, the largest and the most famous platform of games. It involves $3,753$ users, $5,134$ game items and $114,713$ interactions. In data preprocessing, we transform all kinds of interactions into implicit feedback, and drop the duplicate interactions. We use the leave-one-out method to divide the training set and test set. For each user $u_i \in \boldsymbol{\mathcal{U}}$, we randomly select $\xi$ of items in $\boldsymbol{\mathcal{V}}_i^+$, and expose the interactions between user $u_i$ and these selected items to attacker.

TABLE II
Sizes of Datasets

| Dataset | #users | #items | #interactions | Avg. | Sparsity |
|---|---|---|---|---|---|
| MovieLens-100K | 943 | 1,682 | 100,000 | 106 | 93.70% |
| MovieLens-1M | 6,040 | 3,706 | 1,000,209 | 166 | 95.53% |
| Steam-200K | 3,753 | 5,134 | 114,713 | 31 | 99.40% |

**Baseline Attacks** In our experiments, we compare FedRecAttack with several baseline attack methods, including:

- **Random Attack** [47]: For each malicious user client, attacker randomly selects $(\lfloor\frac{\kappa}{2}\rfloor - |\boldsymbol{\mathcal{V}}^{tar}|)$ items in addition to $\boldsymbol{\mathcal{V}}^{tar}$, and generates fake interactions between the malicious user and the items.
- **Bandwagon Attack** [48]: Similar to random attack but different, attacker randomly selects items based on items' popularity. More specifically, we define popular items as the set of the top $10\%$ of items which have the most interactions. For each malicious client, in addition to $\boldsymbol{\mathcal{V}}^{tar}$, attacker randomly selects $(\lfloor\frac{\kappa}{2}\rfloor - |\boldsymbol{\mathcal{V}}^{tar}|) \times 10\%$ items from popular items, and $(\lfloor\frac{\kappa}{2}\rfloor - |\boldsymbol{\mathcal{V}}^{tar}|) \times 90\%$ items among the left unselected items.
- **Popular Attack** [47]: In addition to $\boldsymbol{\mathcal{V}}^{tar}$, attacker selects the top $(\lfloor\frac{\kappa}{2}\rfloor - |\boldsymbol{\mathcal{V}}^{tar}|)$ items which have the most interactions. And attacker generates fake interactions between all malicious users and the items.

Besides, we use **None** to indicate the situations when no attacks are conducted. All the baseline attacks increase ER@K of target items by raising their popularity and making their feature vectors similar to those of the popular items.

**Evaluation Metrics** To evaluate attack effectiveness, we adopt three generally-used metrics: ER@5, ER@10 and NDCG@10. The computation of ER@5 and ER@10 follows Eq. (8). For a target item, its rank in user's recommendation list will affect the probability of the item being interacted with by the user. Hence, we are interested not only in ER@K of target items, but also in the ranks of target items in users' recommendation lists. To reflect the ranks, as in [49], we also adopt NDCG@10 as one of our metrics.

**Parameter Setting** Unless otherwise specified, we take the default values of the hyper parameters as: $k = 32, \eta = 0.01, \xi = 1\%, \rho = 5\%, \kappa = 60, C = 1, \zeta = 1$. For ensuring recommender model convergence, we set the total number of training epochs to 200. The attack effectiveness is evaluated at the end of the last training epoch. We conduct our experiments on a Ubuntu Server with 8 NVIDIA Tesla T4 GPUs, 64-bit 18-core Intel(R) Xeon(R) CPU Gold 6240 @ 2.60GHz and 256 GBs of RAM.

### B. Impact of Limitations on Attacker

To explore the impact of limitations on attacker, we set up three auxiliary comparative experiments on MovieLens-100K with different values of $\xi$, $\rho$ and $\kappa$ respectively.

**The proportion of public interactions ($\xi$)** Table III shows the effectiveness of FedRecAttack with different values of $\xi$ in $\{1\%, 2\%, 3\%, 5\%, 10\%\}$. As can be inferred from the table, increasing the proportion of public interactions can improve attack effectiveness, but the benifits diminish. When $\xi$ is only $1\%$, our attack is quite effective. Even if $\xi$ is increased to $10\%$ from $1\%$ (which also makes the attack more demanding), the improvement of attack effectiveness is less than $5\%$. Considering both the attack effectiveness and the difficulty of conducting the attack, it is reasonable for us to set the default value of $\xi$ to be $1\%$. From the general observation, we could conclude that FedRecAttack is effective with very small proportion of public interactions.

TABLE III
IMPACT OF $\xi$ ON EFFECTIVENESS OF FEDRECATTACK.

| Metric | Proportion of Public Interactions | | | | |
|---|---|---|---|---|---|
| | $\xi = 1\%$ | $\xi = 2\%$ | $\xi = 3\%$ | $\xi = 5\%$ | $\xi = 10\%$ |
| ER@5 | 0.9400 | 0.9818 | 0.9882 | 0.9936 | 0.9914 |
| ER@10 | 0.9475 | 0.9893 | 0.9914 | 0.9946 | 0.9925 |
| NDCG@10 | 0.9411 | 0.9789 | 0.9866 | 0.9886 | 0.9890 |

**The proportion of malicious users ($\rho$)** To explore the impact of the proportion of malicious users on the effectiveness of FedRecAttack, we set different values of $\rho$ in $\{1\%, 2\%, 3\%, 5\%, 10\%\}$. It is evident from Table IV that the proportion of malicious users is a key factor to the effectiveness of FedRecAttack. The effectiveness is improved significantly as the proportion increases. Particularly, when $\rho = 5\%$, ER@5 comes to a value of $0.9400$, which is just a little worse than the maximum. Amongst other model poisoning attacks, $\rho = 5\%$ is adequately small. Remember that the attack cost is positively correlated with the proportion of malicious users. Taking both aspects of effectiveness and cost of attack into consideration, we set the default value of $\rho$ to be $5\%$. It could be concluded that FedRecAttack reaches quite good results with fairly small proportion of malicious clients.

TABLE IV
IMPACT OF $\rho$ ON EFFECTIVENESS OF FEDRECATTACK.

| Metric | Proportion of Malicious Users | | | | |
|---|---|---|---|---|---|
| | $\rho = 1\%$ | $\rho = 2\%$ | $\rho = 3\%$ | $\rho = 5\%$ | $\rho = 10\%$ |
| ER@5 | 0.0011 | 0.0043 | 0.6902 | 0.9400 | 0.9475 |
| ER@10 | 0.0011 | 0.0075 | 0.7395 | 0.9475 | 0.9518 |
| NDCG@10 | 0.0011 | 0.0042 | 0.6615 | 0.9411 | 0.9423 |

**The maximum number of non-zero rows in uploaded poisoned gradients ($\kappa$)** We examine the results of different values of $\kappa$ in $\{20, 40, 60, 80, 100\}$. As shown in Table V, $\kappa$ has little impact on attack effectiveness. The validity of FedRecAttack can be proved by the fact that, for all values of $\kappa$, the attack effectiveness remains in high level (all metrics reach $94\%$). For central server, more number of non-zero rows in user's uploaded gradients implies more number of items which the user has interacted with. If the number of non-zero rows in uploaded poisoned gradients is abnormally high, the attack is highly detectable. On all three datasets, for most users, the number of interactions is around $30$, which indicates the number of non-zero rows in uploaded gradients is around $60$ (including both positive and negative item gradients). Based on the above evidence, it is suitable to set $\kappa = 60$ as default.

TABLE V
IMPACT OF $\kappa$ ON EFFECTIVENESS OF FEDRECATTACK.

| Metric | Maximum Number of Non-zero Rows | | | | |
|---|---|---|---|---|---|
| | $\kappa = 20$ | $\kappa = 40$ | $\kappa = 60$ | $\kappa = 80$ | $\kappa = 100$ |
| ER@5 | 0.9475 | 0.9464 | 0.9400 | 0.9507 | 0.9453 |
| ER@10 | 0.9539 | 0.9518 | 0.9475 | 0.9593 | 0.9518 |
| NDCG@10 | 0.9453 | 0.9442 | 0.9411 | 0.9480 | 0.9456 |

## C. *Effectiveness of Attacks*

We compare the effectiveness of FedRecAttack with that of other baseline attacks in cases of different values of $\rho$ $\{3\%, 5\%, 10\%\}$ on three real-world datasets respectively. Note that for FedRecAttack we set $\xi = 1\%$. As shown in Table VII, FedRecAttack achieves the best attack effectiveness on all three datasets with different proportions of malicious users. In cases of small proportion of malicious users, other methods have merely slight effects, while FedRecAttack remains good performance. From these evidences, we can draw the conclusion that FedRecAttack has the least dependency on the proportion of malicious users. Moreover, from the table we can find that, the more dense the dataset is, the more difficult the attack is. When the proportion of malicious users is kept constant, the attack effectiveness is higher on the more sparse dataset.

Additionally, we compare the effectiveness of FedRecAttack with that of two state-of-the-art data poisoning attacks on MovieLens-100K. The two attacks are as following:

1) P1 [15], [41]: data poisoning attack to matrix factorization based recommender systems.
2) P2 [16]: data poisoning attack to deep learning based recommender systems.

For FedRecAttack, we keep the limitation on attacker's prior knowledge and set $\xi = 1\%$. For P1 and P2, because both of them highly rely on attacker's prior knowledge of user-item interactions and can not work with the limitation, we conduct the experiments with the same settings as in [16] (assuming attacker has access to all user-item interactions). We set different values of $\rho$ in $\{0.5\%, 1\%, 3\%, 5\%\}$ and adopt ER@10 to evaluate the attack effectiveness. As shown in Table VI, P1 and P2 have better performance when the proportion of malicious users is extremely small. We argue that they benifit from attacker's prior knowledge of all interactions. However, ER@10 of these methods is hardly able to reach a satisfactory level, while that of FedRecAttack rapidly rises to a significant level as $\rho$ increases.

TABLE VI
ER@10 OF FEDRECATTACK AND OTHER DATA POISONING ATTACKS ON MOVIELENS-100K.

| Attack Method | Proportion of Malicious Users | | | |
|---|---|---|---|---|
| | $\rho = 0.5\%$ | $\rho = 1\%$ | $\rho = 3\%$ | $\rho = 5\%$ |
| None | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| P1 | 0.0001 | 0.0002 | 0.0014 | 0.0033 |
| P2 | 0.0007 | 0.0019 | 0.0111 | 0.0206 |
| **FedRecAttack** | **0.0000** | **0.0011** | **0.7449** | **0.9475** |

TABLE VII
EFFECTIVENESS OF DIFFERENT ATTACKS WITH DIFFERENT PROPORTIONS OF MALICIOUS USERS.

| Dataset | Attack Method | Proportion of Malicious Users | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\rho = 3\%$ | | | $\rho = 5\%$ | | | $\rho = 10\%$ | | |
| | | ER@5 | ER@10 | NDCG@10 | ER@5 | ER@10 | NDCG@10 | ER@5 | ER@10 | NDCG@10 |
| MovieLens-100K | None | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| | Random | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0011 | 0.0011 | 0.0004 |
| | Bandwagon | 0.0011 | 0.0011 | 0.0011 | 0.0000 | 0.0021 | 0.0006 | 0.0000 | 0.0000 | 0.0000 |
| | Popular | 0.0011 | 0.0011 | 0.0005 | 0.0011 | 0.0011 | 0.0011 | 0.0032 | 0.0075 | 0.0035 |
| | **FedRecAttack** | **0.6988** | **0.7449** | **0.6702** | **0.9400** | **0.9475** | **0.9411** | **0.9507** | **0.9528** | **0.9455** |
| MovieLens-1M | None | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| | Random | 0.0000 | 0.0000 | 0.0000 | 0.0002 | 0.0002 | 0.0001 | 0.0002 | 0.0005 | 0.0002 |
| | Bandwagon | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0010 | 0.0012 | 0.0008 |
| | Popular | 0.0035 | 0.0056 | 0.0030 | 0.0393 | 0.0503 | 0.0349 | 0.1358 | 0.1598 | 0.1255 |
| | **FedRecAttack** | **0.9722** | **0.9752** | **0.9684** | **0.9659** | **0.9704** | **0.9610** | **0.9689** | **0.9742** | **0.9646** |
| Steam-200K | None | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| | Random | 0.0027 | 0.0037 | 0.0022 | 0.0024 | 0.0029 | 0.0025 | 0.0029 | 0.0032 | 0.0027 |
| | Bandwagon | 0.0133 | 0.0157 | 0.0121 | 0.0702 | 0.0952 | 0.0669 | 0.8829 | 0.8944 | 0.8774 |
| | Popular | 0.2067 | 0.3129 | 0.1994 | 0.7165 | 0.7639 | 0.6908 | 0.8349 | 0.8480 | 0.8246 |
| | **FedRecAttack** | **0.9843** | **0.9848** | **0.9833** | **0.9835** | **0.9848** | **0.9831** | **0.9864** | **0.9869** | **0.9852** |

Moreover, we compare FedRecAttack with following state-of-the-art model poisoning attacks on MovieLens-1M:

1) P3 [28]: poison the federated learning model by directing it to misclassify the target input.
2) P4 [50]: a genaral method for attacking distributed models with defense mechanisms.
3) EB [31]: explicitly boost the predicted scores between malicious users and target items.
4) PipAttack [31]: poison the federated recommender model with the information about items popularity.

Since the above model poisoning attacks require more malicious users to be effective, we set $\rho = \{10\%, 20\%, 30\%, 40\%\}$ respectively. For FedRecAttack, we set $\xi = 1\%$. And for the others, we adopt the same settings as in [31] (assuming attacker has side information about items' popularity). As presented in Table VIII, it is clear that ER@5 of P3, P4 and EB is numerically unstable with different proportions of malicious users, while both PipAttack and FedRecAttack maintain comparatively high effectiveness. And when the proportion of malicious users is small (*i.e.,* $\rho = 10\%$), FedRecAttack has the best performance in all model poisoning attacks, which demonstrates the strengths of FedRecAttack.

### D. Stealthiness of Attacks

Our work focuses not only on the effectiveness of attacks, but also on the stealthiness of attacks. Methods to detect attacks in FR can be divided into two categories: i) One line of the methods detects whether there are any abnormalities in the uploaded gradients. ii) The other line of the methods observes whethere there is a suspicious degradation in recommendation accuracy during the training stage. Unfortunately, the former line of methods does not perform well in FR because of the following two reasons: i) The feature vectors of different users

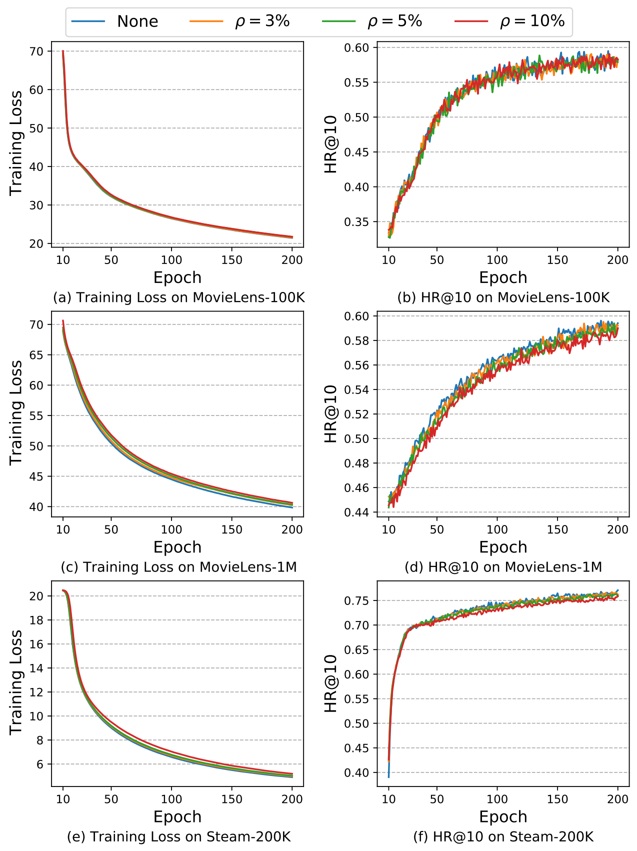

Fig. 3. The side effects of FedRecAttack with different proportions of malicious users on MovieLens-100K, MovieLens-1M and Steam-200K.

vary widely, hence the uploaded gradients vary widely. ii) In Eq. (5), as guided by the policy of differential privacy, user clients will add noise to their uploaded gradients. Both of the

TABLE VIII
HR@10 AND ER@5 OF FEDRECATTACK AND OTHER MODEL POISONING ATTACKS ON MOVIELENS-1M.

| **Attack Method** | **Proportion of Malicious Uers** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $\rho = 10\%$ | | $\rho = 20\%$ | | $\rho = 30\%$ | | $\rho = 40\%$ | |
| | HR@10 | ER@5 | HR@10 | ER@5 | HR@10 | ER@5 | HR@10 | ER@5 |
| None | 0.5940 | 0.0000 | 0.5940 | 0.0000 | 0.5940 | 0.0000 | 0.5940 | 0.0000 |
| P3 | 0.4434 | 0.0000 | 0.4430 | 0.0000 | 0.4435 | 0.0154 | 0.4454 | 0.0298 |
| P4 | 0.4392 | 0.0000 | 0.4386 | 0.9625 | 0.4320 | 0.9016 | 0.4425 | 1.0000 |
| EB | 0.4432 | 0.0000 | 0.4449 | 1.0000 | 0.4363 | 0.9998 | 0.4432 | 1.0000 |
| PipAttack | 0.4384 | 0.9513 | 0.4412 | 1.0000 | 0.4401 | 1.0000 | 0.4349 | 1.0000 |
| **FedRecAttack** | **0.5901** | **0.9689** | **0.5800** | **0.9735** | **0.5829** | **0.9733** | **0.5800** | **0.9786** |

two reasons increase the difficulty of distinguishing poisoned gradients from clean gradients. In contrast, the latter line of methods is always much more practical in FR. Different from data poisoning attacks, most model poisoning attacks work without attacker's prior knowledge of user-item interactions. As a result, in model poisoning attacks, the poisoned gradients generated by attacker are not precise enough to raise ER@K without any side effects on recommendation accuracy. If there is a significant drop in recommendation accuracy, the attack would be detected easily.

Therefore, we emphasize the side effects of FedRecAttack on recommendation accuracy to ensure its stealthiness. We adopt HR@10, which is also used in [1], as our metric to evaluate recommendation accuracy. Fig. 3 shows training loss and HR@10 in each training iteration under FedRecAttack and none attack. In FedRecAttack, we set $\rho = \{3\%, 5\%, 10\%\}$ respectively. As the figure shown, the side effects of FedRecAttack are negligible, which means FedRecAttack is stealthy.

Furthermore, we compare HR@10 under FedRecAttack with that under other model poisoning attacks on MovieLens-1M. Since other model poisoning attacks require more malicious users to be effective, we set $\rho = \{10\%, 20\%, 30\%, 40\%\}$ respectively. As shown in Table VIII, all other methods caused significant drops of HR@10 (more than 25%), which indicates these attacks are easy to notice. In contrast, FedRecAttack has the slightest side effects (HR@10 has only dropped less than 2.5%), and its effectiveness always stays in high level (only a little worse than the best). From these observations, we can conclude that FedRecAttack strikes a perfect balance between attack effectiveness and side effects.

We consider that, the function $g$ we used in Eq. (13) is the key that the side effects of FedRecAttack are slight. As $x$ decreases, $g'(x)$ converges to 0. Due to this characteristic of $g$, in FedRecAttack, the predicted scores of target items will not be raised indefinitely, but will only be raised to exactly a little higher than that of the last item in user's recommendation list.

## E. Ablation Study

In order to have a better understanding on the contributions of attacker's prior knowledge in FedRecAttack, we conduct an ablation experiment. We compare the effectiveness of FedRecAttack with public interactions ($\xi = 1\%$) and without any ($\xi = 0\%$) on all three datasets. Although $\xi$ is very small, attacker's prior knowledge of public interactions is essential in FedRecAttack. As shown in Table IX, FedRecAttack is highly effective when $\xi = 1\%$, while loses validity completely when $\xi = 0\%$. The reason of the above observation is that effective poisoned gradients can not be generated, as attacker can not rationally approximate users' feature matrix without attacker's prior knowledge. From these evidences, we can testify our hypothesis of the validity of approximating users' feature matrix.

TABLE IX
EFFECTIVENESS OF FEDRECATTACK WITH & WITHOUT ATTACKER'S PRIOR KNOWLEDGE OF PUBLIC INTERACTIONS.

| **Dataset** | **Metric** | **Proportion of Public Interactions** | |
|---|---|---|---|
| | | $\xi = 1\%$ | $\xi = 0\%$ |
| MovieLens-100K | ER@5 | 0.9400 | 0.0000 |
| | ER@10 | 0.9475 | 0.0000 |
| | NDCG@10 | 0.9411 | 0.0000 |
| MovieLens-1M | ER@5 | 0.9659 | 0.0000 |
| | ER@10 | 0.9704 | 0.0000 |
| | NDCG@10 | 0.9610 | 0.0000 |
| Steam-200K | ER@5 | 0.9835 | 0.0000 |
| | ER@10 | 0.9848 | 0.0000 |
| | NDCG@10 | 0.9831 | 0.0000 |

## VI. CONCLUSION AND FUTURE WORK

In this paper, we present FedRecAttack, a model poisoning attack against FR, which aims to make target items recommended to as many users as possible. We evaluate the performance of FedRecAttack on three real-world datasets of different sizes. The experimental results show that: i) FedRecAttack achieves the state-of-the-art effectiveness. ii) FedRecAttack has the slightest side effects among the existing model poisoning attacks. iii) FedRecAttack still performs well when the proportion of malicious users and the proportion of public interactions are small. From these observations, we can draw the conclusion that FedRecAttack can effectively poison the target recommender model at low cost with high stealthiness, demonstrating the vulnerability of FR.

The mainstream of existing methods to detect model poisoning attacks is training a support vector machine or a deep neural network to distinguish poisoned gradients from clean gradients [51]. The mainstream of existing methods to defend against model poisoning attacks is adopting byzantine-robust aggregations (*e.g.,* k-rum, trimmed mean, median) [52]. Although these methods have been studied in the field of federated learning, they do not fit FR perfectly. In FR, because of the large differences in feature vectors of different users, the gradients uploaded by different users vary widely, which significantly increases the difficulty of detection or defense. Interesting future work includes approaching methods of detecting or defending against model poisoning attacks with the consideration of the specificity of FR.

## ACKNOWLEDGEMENT

This research is supported by the National Key R&D Program of China (2021YFB2700500, 2021YFB2700501) and Scientific Research Fund of Zhejiang University (XY2021049). The authors thank the anonymous reviewers for giving us a lot of valuable comments.

## REFERENCES

[1] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[2] X. He, Z. He, J. Song, Z. Liu, Y. Jiang, and T. Chua, "NAIS: neural attentive item similarity model for recommendation," *TKDE*, vol. 30, no. 12, pp. 2354–2366, 2018.

[3] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 2019, pp. 165–174.

[4] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 639–648.

[5] R. He and J. McAuley, "Vbpr: visual bayesian personalized ranking from implicit feedback," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[6] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 353–362.

[7] Y. Wei, X. Wang, L. Nie, X. He, R. Hong, and T.-S. Chua, "Mmgcn: Multi-modal graph convolution network for personalized recommendation of micro-video," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 1437–1445.

[8] X. Wang, T. Huang, D. Wang, Y. Yuan, Z. Liu, X. He, and T. Chua, "Learning intents behind interactions with knowledge graph for recommendation," in *WWW*, 2021, pp. 878–887.

[9] J. Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao, and D. L. Lee, "Billion-scale commodity embedding for e-commerce recommendation in alibaba," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 839–848.

[10] H. Wang, F. Zhang, X. Xie, and M. Guo, "Dkn: Deep knowledge-aware network for news recommendation," in *Proceedings of the 2018 world wide web conference*, 2018, pp. 1835–1844.

[11] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li, "Drn: A deep reinforcement learning framework for news recommendation," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 167–176.

[12] Y. Wei, Z. Cheng, X. Yu, Z. Zhao, L. Zhu, and L. Nie, "Personalized hashtag recommendation for micro-videos," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 1446–1454.

[13] Z. Liu, P. Qian, X. Wang, Y. Zhuang, L. Qiu, and X. Wang, "Combining graph neural networks with expert knowledge for smart contract vulnerability detection," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[14] R. J. González, "Hacking the citizenry?: Personality profiling,'big data'and the election of donald trump," *Anthropology Today*, vol. 33, no. 3, pp. 9–12, 2017.

[15] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," *Advances in neural information processing systems*, vol. 29, pp. 1885–1893, 2016.

[16] H. Huang, J. Mu, N. Z. Gong, Q. Li, B. Liu, and M. Xu, "Data poisoning attacks to deep learning based recommender systems," in *NDSS*. The Internet Society, 2021.

[17] M. Fang, G. Yang, N. Z. Gong, and J. Liu, "Poisoning attacks to graph-based recommender systems," in *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018, pp. 381–392.

[18] P. S. Chauhan and N. Kshetri, "2021 state of the practice in data privacy and security," *Computer*, vol. 54, no. 08, pp. 125–132, 2021.

[19] G. D. P. Regulation, "General data protection regulation (gdpr)," *Intersoft Consulting, Accessed in October*, vol. 24, no. 1, 2018.

[20] N. Kalyanpur and A. L. Newman, "The mnc-coalition paradox: Issue salience, foreign firms and the general data protection regulation," *JCMS: Journal of Common Market Studies*, vol. 57, no. 3, pp. 448–467, 2019.

[21] M. Ammad-Ud-Din, E. Ivannikova, S. A. Khan, W. Oyomno, Q. Fu, K. E. Tan, and A. Flanagan, "Federated collaborative filtering for privacy-preserving personalized recommendation system," *arXiv preprint arXiv:1901.09888*, 2019.

[22] G. Lin, F. Liang, W. Pan, and Z. Ming, "Fedrec: Federated recommendation with explicit feedback," *IEEE Intelligent Systems*, 2020.

[23] F. Liang, W. Pan, and Z. Ming, "Fedrec++: Lossless federated recommendation with explicit feedback," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 4224–4231.

[24] C. Wu, F. Wu, Y. Cao, Y. Huang, and X. Xie, "Fedgnn: Federated graph neural network for privacy-preserving recommendation," *arXiv preprint arXiv:2102.04925*, 2021.

[25] K. Muhammad, Q. Wang, D. O'Reilly-Morgan, E. Tragos, B. Smyth, N. Hurley, J. Geraci, and A. Lawlor, "Fedfast: Going beyond average for faster training of federated recommender systems," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1234–1242.

[26] J. Wu, Q. Liu, Z. Huang, Y. Ning, H. Wang, E. Chen, J. Yi, and B. Zhou, "Hierarchical personalized federated learning for user modeling," in *Proceedings of the Web Conference 2021*, 2021, pp. 957–968.

[27] L. Minto, M. Haller, B. Livshits, and H. Haddadi, "Stronger privacy for federated collaborative filtering with implicit feedback," in *Fifteenth ACM Conference on Recommender Systems*, 2021, pp. 342–350.

[28] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning*. PMLR, 2019, pp. 634–643.

[29] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2938–2948.

[30] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 1605–1622.

[31] S. Zhang, H. Yin, T. Chen, Z. Huang, Q. V. H. Nguyen, and L. Cui, "Pipattack: Poisoning federated recommender systems for manipulating item promotion," in *WSDM*. ACM, 2022, pp. 1415–1423.

[32] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," *Advances in neural information processing systems*, vol. 20, 2007.

[33] S. Kabbur, X. Ning, and G. Karypis, "Fism: factored item similarity models for top-n recommender systems," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 659–667.

[34] R. Kumar, B. Verma, and S. S. Rastogi, "Social popularity based svd++ recommender system," *International Journal of Computer Applications*, vol. 87, no. 14, 2014.

[35] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016, pp. 549–558.

[36] T. Bai, J.-R. Wen, J. Zhang, and W. X. Zhao, "A neural collaborative filtering model with interaction-based neighborhood," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 1979–1982.

[37] X. He, X. Du, X. Wang, F. Tian, J. Tang, and T. Chua, "Outer product-based neural collaborative filtering," in *IJCAI*. ijcai.org, 2018, pp. 2227–2233.

[38] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th international conference on World Wide Web*, 2015, pp. 111–112.

[39] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," in *Proceedings of the ninth ACM international conference on web search and data mining*, 2016, pp. 153–162.

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

[41] M. Fang, N. Z. Gong, and J. Liu, "Influence function based data poisoning attacks to top-n recommender systems," in *Proceedings of The Web Conference 2020*, 2020, pp. 3019–3025.

[42] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[43] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: bayesian personalized ranking from implicit feedback," in *UAI*. AUAI Press, 2009, pp. 452–461.

[44] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

[45] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[46] G. Cheuque, J. Guzmán, and D. Parra, "Recommender systems for online video game platforms: The case of steam," in *Companion Proceedings of The 2019 World Wide Web Conference*, 2019, pp. 763–771.

[47] I. Gunes, C. Kaleli, A. Bilge, and H. Polat, "Shilling attacks against recommender systems: a comprehensive survey," *Artificial Intelligence Review*, vol. 42, no. 4, pp. 767–799, 2014.

[48] S. Kapoor, V. Kapoor, and R. Kumar, "A review of attacks and its detection attributes on collaborative recommender systems." *International Journal of Advanced Research in Computer Science*, vol. 8, no. 7, 2017.

[49] W. Krichene and S. Rendle, "On sampled metrics for item recommendation," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1748–1757.

[50] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[51] W. Zhou, J. Wen, Q. Xiong, M. Gao, and J. Zeng, "Svm-tia a shilling attack detection method based on svm and target item analysis in recommender systems," *Neurocomputing*, vol. 210, pp. 197–205, 2016.

[52] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.