Poisoning Deep Learning Based Recommender Model in Federated Learning Scenarios

Dazhong Rong, **Qinming He**, **Jianhai Chen***
College of Computer Science and Technology, Zhejiang University
{rdz98, hqm, chenjh919}@zju.edu.cn

Abstract

Various attack methods against recommender systems have been proposed in the past years, and the security issues of recommender systems have drawn considerable attention. Traditional attacks attempt to make target items recommended to as many users as possible by poisoning the training data. Benifiting from the feature of protecting users' private data, federated recommendation can effectively defend such attacks. Therefore, quite a few works have devoted themselves to developing federated recommender systems. For proving current federated recommendation is still vulnerable, in this work we probe to design attack approaches targeting deep learning based recommender models in federated learning scenarios. Specifically, our attacks generate poisoned gradients for manipulated malicious users to upload based on two strategies (*i.e.,* random approximation and hard user mining). Extensive experiments show that our well-designed attacks can effectively poison the target models, and the attack effectiveness sets the state-of-the-art.

1 Introduction

Recommender Systems (RS) have become one of the major channels for people to acquire information, and can directly influence people's perceptions while recommending items. However, there are many vulnerabilities in RS which can be exploited to manipulate recommendation of items. Attacks against RS aim to get target items recommended to as many users as possible by raising the predicted scores of target items. One line of the attacks inject fake interactions to poison training data, termed as data poisoning attack [Kapoor *et al.*, 2017]. As attacker can only inject a limited number of fake interactions, these attacks are less effective. Some works attempt to generate more crafted fake interactions by considering real interactions to improve attack effectiveness [Li *et al.*, 2016; Huang *et al.*, 2021]. However, in Federated Learning (FL) scenarios, due to the fact that attacker has no access to real interactions, these works are not applicable. Another line of the attacks directly poison recommender model by generating poisoned gradients, termed as model poisoning attack. These attacks are specially targeted at FL scenarios and usually more effective than data poisoning attacks. However, these attacks rely on attacker's prioir knowledge (*e.g.,* items' popularity [Zhang *et al.*, 2022] or public interactions [Rong *et al.*, 2022]). Under the circumstances that user privacy is strictly protected in FL scenarios, they also lose validity. We summerized the key challenges here as following:

- Existing attacks with attacker's prior knowledge are not applicable in FL scenarios.
- Existing attacks without attacker's prior knowledge have poor effectiveness.

We aim to poison deep learning based recommender model in FL scenarios without the prior knowledge. To address the challenges above, we first approximate benign users' embedding vectors, and then generate poisoned gradients based on the approximated vectors rather than side information.

In this paper, we present two attack methods using different ways to approximate benign users' embedding vectors. In the first attack method, inspired by recent work on self-supervised learning for recommendation [Wu *et al.*, 2021; Yao *et al.*, 2021] (See Section 3.1 for more details), we approximate benign users' embedding vectors with gaussian distribution. In the second attack method, inspired by OHEM [Shrivastava *et al.*, 2016] (See Section 3.2 for more details), we first initialize the approximated benign users' embedding vectors with gaussian distribution, and then optimize the vectors by gradient descent to mine hard users. The key contributions of our work[1] are:

- To the best of our knowledge, we are the first to present attacks against RS in FL scenarios without attacker's prior knowledge.
- We proposed two strategies (*i.e.,* random approximation and hard user mining) to approximate benign users' embedding vectors in our attacks. Hard user mining can improve the attack effectiveness when the proportion of malicious users is extremely small.
- Extensive experiments on two real world datasets demonstrate that our attacks are effective and outperform all baseline attacks.

*Corresponding author

[1]https://github.com/rdz98/PoisonFedDLRS

2 Preliminaries

In this section, we briefly introduce our base recommender model and the framework of federated recommendation.

2.1 Base Recommender Model

Neural Collaborative Filtering (NCF) [He *et al.*, 2017] is one of the most widely used deep learning based recommender models, and has state-of-the-art performance of recommendation. Without loss of generality, we adopt NCF as our base recommender model.

Respectively, Let N and M denote the number of users and items in the target recommender system. Let $\boldsymbol{\mathcal{U}}$ and $\boldsymbol{\mathcal{I}}$ denote the set of N users and the set of M items. Each user $\boldsymbol{u} \in \boldsymbol{\mathcal{U}}$ has an embedding vector $\boldsymbol{p}_u$ which describes its latent feature. Similarly, each item $\boldsymbol{i} \in \boldsymbol{\mathcal{I}}$ has an embedding vector $\boldsymbol{q}_i$. We use $\hat{Y}_{ui}$ to denote the predicted score between user $\boldsymbol{u}$ and item $\boldsymbol{i}$, which indicates the preference of user $\boldsymbol{u}$ for item $\boldsymbol{i}$. In NCF, $\hat{Y}_{ui}$ is predicted as following:

$$\hat{Y}_{ui} = \boldsymbol{\Upsilon}(\boldsymbol{p}_u, \boldsymbol{q}_i), \tag{1}$$

where $\boldsymbol{\Upsilon}$ is the user-item interaction function. To model feedback signals with a high-level of non-linearities, NCF utilizes a Multi-Layer Perceptron (MLP) to learn the function $\boldsymbol{\Upsilon}$, as following:

$$\boldsymbol{\Upsilon}(\boldsymbol{p}_u, \boldsymbol{q}_i) = \mathbf{a_{out}}(\boldsymbol{h}^T \boldsymbol{\Phi}(\boldsymbol{p}_u \oplus \boldsymbol{q}_i)), \tag{2}$$

where $\mathbf{a_{out}}$, $\boldsymbol{h}$, $\boldsymbol{\Phi}$ and $\oplus$ denote the activation function, the weight vector, the MLP function and the vector concatenation respectively. We use the sigmoid function as $\mathbf{a_{out}}$ to restrict the model output to be in $(0, 1)$. More specifically, let $\mathbf{a_k}$, $\boldsymbol{W}_k$ and $\boldsymbol{b}_k$ denote the activation function, the weight matrix and the bias vector in the k-th layer of the perceptron, respectively. In NCF with L hidden layers, the MLP function $\boldsymbol{\Phi}$ is as following:

$$\boldsymbol{\Phi}(\boldsymbol{x}) = \boldsymbol{\phi_L}(\dots(\boldsymbol{\phi_2}(\boldsymbol{\phi_1}(\boldsymbol{x}))\dots), \tag{3}$$

where $\boldsymbol{\phi_k}(\boldsymbol{x}) = \mathbf{a_k}(\boldsymbol{W}_k^T \boldsymbol{x} + \boldsymbol{b}_k)$. To achieve better performance of recommendation, we use Rectifier (ReLU) as the activation function $\mathbf{a_k}$ in all perceptron layers.

It is worth mentioning that, in federated learning scenarios, our attacks also apply to many other recommender models (*e.g.,* NNCF [Bai *et al.*, 2017] and ONCF [He *et al.*, 2018]) as long as they learn the user-item interaction function $\boldsymbol{\Upsilon}$ through deep neural networks.

2.2 Framework of Federated Recommendation

Our base recommender model is distributedly trained under the framework of federated recommendation. More specifically, in FL scenarios, there is a central server and a large amount of individual user clients. As each user corresponds to one of the user clients, we use user to represent its client for convenience. Let $\boldsymbol{P}, \boldsymbol{Q}$ respectively denote the embedding matrices of users and items, where $\boldsymbol{p}_u$ is row of $\boldsymbol{P}$, and $\boldsymbol{q}_i$ is row of $\boldsymbol{Q}$. Let $\boldsymbol{\mathcal{D}}_u$ denote the training dataset of user $\boldsymbol{u}$, which consists of item-score pairs $(\boldsymbol{i}, Y_{ui})$. If user $\boldsymbol{u}$ has interacted with item $\boldsymbol{i}$, $Y_{ui} = 1$ (*i.e.,* a positive instance). Otherwise, $Y_{ui} = 0$ (*i.e.,* a negative instance). Note that, for each user $\boldsymbol{u}$, since there are a large amount of uninteracted items, the negative instances in $\boldsymbol{\mathcal{D}}_u$ are sampled with a ratio of $r : 1$ as the number of negative instances against that of positive. Let $\boldsymbol{\Theta}$ denote all trainable model parameters except for users' embedding matrix $\boldsymbol{P}$ (In NCF, $\boldsymbol{\Theta} = \{\boldsymbol{Q}, \boldsymbol{h}, \boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{W}_2, \boldsymbol{b}_2, \dots, \boldsymbol{W}_L, \boldsymbol{b}_L\}$). For each user $\boldsymbol{u}$, $\boldsymbol{p}_u$ and $\boldsymbol{\mathcal{D}}_u$ are stored locally on its own device. Meanwhile, $\boldsymbol{\Theta}$ are stored on the central server.

To train our base recommender model, we adopt the Binary Cross-Entropy (BCE) loss to quantify the difference between the model predicted scores and the ground-truth scores on training dataset. Let $\boldsymbol{\mathcal{L}}_u$ denote the loss function for user u. We can consider $\boldsymbol{\mathcal{L}}_u$ as a function of $\boldsymbol{p}_u$ and $\boldsymbol{\Theta}$ as following:

$$\boldsymbol{\mathcal{L}}_u = -\sum_{(\boldsymbol{i}, Y_{ui}) \in \boldsymbol{\mathcal{D}}_u} Y_{ui} \log \hat{Y}_{ui} + (1 - Y_{ui}) \log(1 - \hat{Y}_{ui}). \tag{4}$$

Once again note that, without loss of generality, our attacks are valid while other popular loss functions (*e.g.,* BPR loss [Rendle *et al.*, 2009]) are adopted.

In every training round, the central server randomly selects a subset of users to participate in training. Let $\boldsymbol{\mathcal{U}}^t$ denote the selected user subset in the t-th round. Let $\boldsymbol{p}_u^t$ and $\boldsymbol{\Theta}^t$ denote $\boldsymbol{p}_u$ and $\boldsymbol{\Theta}$ in the t-th round, respectively. In the t-th round, the central server first sends a copy of $\boldsymbol{\Theta}^t$ to each selected users in $\boldsymbol{\Theta}^t$. Then, for each selected user $\boldsymbol{u}$, with $\boldsymbol{\mathcal{L}}_u$ computed, it derives the gradients of $\boldsymbol{p}_u^t$ and $\boldsymbol{\Theta}^t$ denoted by $\nabla \boldsymbol{p}_u^t$ and $\nabla \boldsymbol{\Theta}_u^t$. Moreover, each selected user $\boldsymbol{u}$ updates $\boldsymbol{p}_u$ locally with learning rate η as following:

$$\boldsymbol{p}_u^{t+1} = \boldsymbol{p}_u^t - \eta \nabla \boldsymbol{p}_u^t, \tag{5}$$

and uploads $\nabla \boldsymbol{\Theta}_u^t$ to the central server. At last, after the central server collects all uploaded gradients from selected users, it updates $\boldsymbol{\Theta}$ by aggregating the uploaded gradients as following:

$$\boldsymbol{\Theta}^{t+1} = \boldsymbol{\Theta}^t - \eta \sum_{\boldsymbol{u} \in \boldsymbol{\mathcal{U}}^t} \nabla \boldsymbol{\Theta}_u^t. \tag{6}$$

There are repetitive rounds of training until the recommender model converges.

3 Our Attacks

In order to avoid ambiguity, we use $\boldsymbol{\mathcal{U}}$ and $\tilde{\boldsymbol{\mathcal{U}}}$ to denote the set of benign users and the set of malicious users injected by attacker, respectively. We assume that for each user the recommender model recommends K items with the highest predicted scores among the items that the user did not interact with. Let $\boldsymbol{\mathcal{I}}_u$ denote the top-K recommended items for user $\boldsymbol{u}$. The Exposure Ratio at rank K (ER@K) of item $\boldsymbol{i}$ is defined as: $|\{u \in \boldsymbol{\mathcal{U}} | i \in \boldsymbol{\mathcal{I}}_u\}| / |\{u \in \boldsymbol{\mathcal{U}} | (\boldsymbol{i}, 1) \notin \boldsymbol{\mathcal{D}}_u\}|$.

Let $\tilde{\boldsymbol{\mathcal{I}}}$ denote the set of target items. Let $\boldsymbol{\varepsilon_i}$ denote ER@K of item $\boldsymbol{i}$. The goal of our attacks is to raise ER@K of target items (*i.e.,* maximize $\sum_{i \in \tilde{\boldsymbol{\mathcal{I}}}} \boldsymbol{\varepsilon_i}$). To achieve the goal, attacker manipulates malicious users to upload well-crafted poisoned gradients to central server. In this section, we propose two attack methods A-ra and A-hum that use different strategies (*i.e.,* random approximation and hard user mining) to generate the poisoned gradients. In the following, we will introduce them respectively.

3.1 Attack with Random Approximation (A-ra)

The attacks against RS in FL scenarios can be considered as a special class of backdoor attacks. Gu *et al.* [2017] backdoored neural networks in the outsourced training scenario by mixing clean inputs and backdoored inputs. Inspired by the previous work, we backdoor the target recommender model similarly by uploading a mix of clean gradients and poisoned gradients. Let $\tilde{\mathcal{L}}$ denote the loss function which can indicate the goal of our attacks. In the t-th round of training, each selected benign user $\boldsymbol{u}$ uploads gradients of $\mathcal{L}_u$ (*i.e.,* clean gradients). To achieve the goal of our attacks, we manipulate the selected malicious users to upload gradients of $\tilde{\mathcal{L}}$ (*i.e.,* poisoned gradients). The actual loss function of the recommender model under our attacks can be represented as $\sum_{\boldsymbol{u}\in\boldsymbol{\mathcal{U}}} \mathcal{L}_u + \alpha\tilde{\mathcal{L}}$, where α is a positive coefficient that trades off between the model validity and the attack effectiveness. Note that attacker can adjust α by changing the learning rate for malicious users.

Unfortunately, we can not bring ER@K of target items into $\tilde{\mathcal{L}}$, because:

- ER@K is a highly non-differentiable discontinuous function of $\boldsymbol{\Theta}$, hence leads to the difficulty in the computation of effective poisoned gradients.
- In our attacks, each benign user's embedding vector and training dataset are unknown to attacker. However, ER@K depends on these parameters.

To address the problems, we adopt two steps to design a proper loss function $\tilde{\mathcal{L}}$ for our attacks.

Step 1: Using predicted scores to indicate ER@K. For each item $\boldsymbol{i}$, there is a positive correlation between its ER@K (*i.e.,* $\boldsymbol{\varepsilon_i}$) and its average predicted score to all benign users (*i.e.,* $\frac{1}{|\boldsymbol{\mathcal{U}}|}\sum_{\boldsymbol{u}\in\boldsymbol{\mathcal{U}}}\hat{Y}_{ui}$). Instead of maximizing $\sum_{i\in\tilde{\boldsymbol{\mathcal{I}}}}\boldsymbol{\varepsilon_i}$, we can change the goal of our attacks to maximizing $\frac{1}{|\boldsymbol{\mathcal{U}}|}\sum_{i\in\tilde{\boldsymbol{\mathcal{I}}}}\sum_{u\in\boldsymbol{\mathcal{U}}}\hat{Y}_{ui}$. Different from $\boldsymbol{\varepsilon_i}$, $\hat{Y}_{ui}$ is a continuous and differentiable function of $\boldsymbol{\Theta}$. However, if user u have interacted with the target item $\boldsymbol{i}$, $\hat{Y}_{ui}$ does not affect $\boldsymbol{\varepsilon_i}$. Moreover, if the target item $\boldsymbol{i}$ has already been in user $\boldsymbol{u}$'s top-K recommendation list, increasing $\hat{Y}_{ui}$ is meaningless. Due to the above two reasons, our new goal leads to a little loss of precision.

Step 2: Approximating users' embedding vectors with normal distribution. The scores predicted by target recommender model depend on $\boldsymbol{P}$ and $\boldsymbol{\Theta}$. In the t-th round of training, the central server sends a copy of $\boldsymbol{\Theta}^t$ to each user in $\boldsymbol{\mathcal{U}}^t$. Attacker can easily access $\boldsymbol{\Theta}^t$ through the malicious users in $\boldsymbol{\mathcal{U}}^t$. However, $\boldsymbol{P}$ is always inaccessible for attacker, because $\boldsymbol{P}$ is distributedly stored on benign users' clients as their embedding vectors. To carry out our attacks, we have to approximate $\boldsymbol{P}$. Some recent studies on self-supervised learning for recommendation [Wu *et al.*, 2021; Yao *et al.*, 2021] introduce contrastive learning to make the unit embedding vectors of different users or different items far apart, so as to improve the accuracy and robustness for recommendation. Inspired by these studies, we realize that the directions of users' embedding vectors are expected to be uniformly distributed. With the additional consideration of the effect of L2-regularization, we roughly assume that users' embedding vectors follow a gaussian distribution. Therefore, we approximate benign user's embedding vector as a vector-valued random variable $\hat{\mathbf{p}} \sim N(0, \sigma^2)$, where σ is a hyperparameter. Let $\hat{\mathbf{p}}_j$ denotes our j-th approximated embedding vector. We further change the goal of our attack to maximizing $\frac{1}{n}\sum_{i\in\tilde{\boldsymbol{\mathcal{I}}}}\sum_{j=1}^{n}\boldsymbol{\Upsilon}(\hat{\mathbf{p}}_j, \boldsymbol{q}_i)$, where n is a hyperparameter which indicates the stability of our attack. The larger n is, the slighter the effect of randomness to our attack is.

Considering the stealthiness of our attacks, $\tilde{\mathcal{L}}$ is designed as following:

$$\tilde{\mathcal{L}}(\boldsymbol{\Theta}) = \frac{1}{n}\sum_{i\in\tilde{\boldsymbol{\mathcal{I}}}}\sum_{j=1}^{n} -\log\boldsymbol{\Upsilon}(\hat{\mathbf{p}}_j, \boldsymbol{q}_i). \tag{7}$$

In order to make $\tilde{\mathcal{L}}$ have a similar structure to $\mathcal{L}$, we maximize $\boldsymbol{\Upsilon}(\hat{\mathbf{p}}_j, \boldsymbol{q}_i)$ by minimizing the BCE loss between $\boldsymbol{\Upsilon}(\hat{\mathbf{p}}_j, \boldsymbol{q}_i)$ and the maximum score (*i.e.,* 1).

To summarize the flow of our attacks in the t-th round of training, the selected malicious users first derive the gradients $\widetilde{\nabla\boldsymbol{\Theta}}^t$ of $\boldsymbol{\Theta}^t$ with $\tilde{\mathcal{L}}$ computed, and then upload $\widetilde{\nabla\boldsymbol{\Theta}}^t$ to the central server.

3.2 Attack with Hard User Mining (A-hum)

In A-ra, the predicted scores of target items to most benign users will increase significantly under the effect of poisoned gradients. However, for the benign user $\boldsymbol{u}$ who takes the target item $\boldsymbol{i}$ as a negative instance (*i.e.,* $(i, 0) \in \boldsymbol{\mathcal{D}}_u$), its uploaded gradients have opposite effect to poisoned gradients on the predicted score $\hat{Y}_{ui}$. For convenience, we call the set of such benign users as hard users of item $\boldsymbol{i}$. It is difficult to productively increase the predicted scores of target items to their corresponding hard users. As a result, our attack effectiveness is limited.

We argue that Step 2 in A-ra is still improvable. Shrivastava *et al.* [2016] introduced an Online Hard Example Mining (OHEM) algorithm which automatically select hard examples to train region-based ConvNet detectors in a more effective and efficient way. Inspired by OHEM, to address the above challenge and further improve our attack effectiveness, we propose A-hum. In A-hum, we use gradient descent to mine hard users, and generate poisoned gradients in allusion to them. More specifically, the flow of A-hum in the t-th round of training is as following:

1. For each target item $\boldsymbol{i}$, we first initialize the approximated embedding vectors of its hard user as $\hat{\mathbf{p}} \sim N(0, \sigma^2)$, which is the same as that in Step 2 in A-ra. Let $\hat{\mathbf{p}}_j^i$ denote the j-th approximated embedding vector for the target item $\boldsymbol{i}$.
2. For each target item $\boldsymbol{i}$ and its hard user $\boldsymbol{u}$, due to the fact that $-\log(1 - \boldsymbol{\Upsilon}(\boldsymbol{p}_u, \boldsymbol{q}_i))$ is one term of $\mathcal{L}_u$, its value is expected to be small. Therefore, we define the loss function of hard user mining for the target item $\boldsymbol{i}$ as following:

$$\boldsymbol{l}_i(\boldsymbol{p}) = -\log(1 - \boldsymbol{\Upsilon}(\boldsymbol{p}, \boldsymbol{q}_i)). \tag{8}$$

And then for each $\hat{\mathbf{p}}_j^i$, we optimize it to minimize $\boldsymbol{l}_i(\hat{\mathbf{p}}_j^i)$ by gradient descent as following:

$$\hat{\mathbf{p}}_j^i \leftarrow \hat{\mathbf{p}}_j^i - \xi \frac{\partial \boldsymbol{l}_i}{\partial \hat{\mathbf{p}}_j^i}(\hat{\mathbf{p}}_j^i), \tag{9}$$

where ξ is the learning rate for hard user mining. This optimization process is repeated for β iterations, where β is a hyperparameter.

3. Finally, we design $\tilde{\boldsymbol{\mathcal{L}}}$ as following:

$$\tilde{\boldsymbol{\mathcal{L}}}(\boldsymbol{\Theta}) = \frac{1}{n} \sum_{i \in \tilde{\boldsymbol{\mathcal{I}}}} \sum_{j=1}^{n} -\log \boldsymbol{\Upsilon}(\hat{\mathbf{p}}_j^i, \boldsymbol{q}_i), \tag{10}$$

which specifically aims to solve the difficulty brought by the hard users. As in A-ra, the selected malicious users first derive the gradients $\widetilde{\nabla \boldsymbol{\Theta}^t}$ of $\boldsymbol{\Theta}^t$ with $\tilde{\boldsymbol{\mathcal{L}}}$ computed, and then upload $\widetilde{\nabla \boldsymbol{\Theta}^t}$ to the central server.

4 Related Work

In Federated Recommendation (FR), each user's privacy (historical interactions) is kept locally on the user client itself instead of being collected and uploaded to the central server, which means that the user's privacy will not be exposed to anyone else including the central server. Because users' privacy is well protected, FR becomes a hot topic of research, and a large number of studies (*e.g.,* [Ammad-Ud-Din *et al.*, 2019; Lin *et al.*, 2020; Liang *et al.*, 2021]) spring up on this research area. Fueled by the maturity of FR techniques, the recommender systems based on FR are becoming continuously more widely used in diverse scenarios, as long as the security issues of FR are emerging. Although some research has been carried out on FR, few studies have investigated the security of FR. In the following we will discuss the studies related to the attacks against FR from three perspectives respectively.

Attacks against centralized recommendation. Data poisoning attack is the main stream of attacks against centralized recommendation, originating from adversarial machine learning. In such attacks, attacker injects a small amount of fake users and generates well-crafted interactions by manipulating the fake users to interact with certain specific items. The recommender model trained on the poisoned data (fake users' interactions) will predict abnormally high scores of target items to many users. Kapoor *et al.* [2017] introduced two naive data poisoning attacks (*i.e.,* random attack and bandwagon attack). In random attack, fake users interact with target items and randomly selected items under attacker's manipulation. In bandwagon attack, fake users interact with target items and certain items with high popularity under attacker's manipulation. Note that bandwagon attack relies on side information about items' popularity. Both random attacks and bandwagon attacks don't take into consideration the algorithm adopted by the target recommender model. As a result, these two attacks have limited effectiveness. To improve on this shortcoming, Li *et al.* [2016], Fang *et al.* [2018] and Huang *et al.* [2021] developed data poisoning attacks for matrix factorization based, graph based and deep learning based recommender systems respectively. Note that all these three attacks rely on attacker's prior knowledge of historical interactions. Even though these attacks consider the recommendation algorithm and take use of attacker's prior knowledge, their effectiveness is still limited as attacker can only poison a minor part of the training data. Unlike centralized recommendation, in FR, attacker can directly poison the gradients uploaded to the central server through malicious clients. Hence, theoretically, attacks against FR can achieve better effectiveness than attacks against centralized recommendation. More importantly, most of data poisoning attacks require access to all or major part of user-item interactions. These attacks are quite impractical in FR scenarios, because each user's historical interactions are kept locally and attacker has no access to them.

Attacks against federated learning. McMahan *et al.* [2017] proposed a framework of Federated Learning (FL), which trains deep neural network distributedly with thousands participants. In every training round of FL, the central server randomly selects a subset of user clients, then share the master model with these clients. Each of these clients trains the master model locally with the training data kept on the client itself, then uploads the gradients of model parameters to the central server. At the end of the round, the central server updates the master model by averaging the uploaded gradients. The issues of security arise along with the rapid development of FL. A large number of attacks against FL are proposed. Bagdasaryan *et al.* [2020] introduced model poisoning attacks to which FL is generically vulnerable. In model poisoning attack, attacker directly influences master model by controlling malicious clients to upload poisoned gradients. Subsequently, Fang *et al.* [2020] performed the first systematic study on attacking Byzantine-robust FL and proposed local model poisoning attacks against Byzantine-robust FL. However, these attacks against FL rely on the complete access to the model, which is not realistic in FR. FR is a special case of FL, because in FR users' embedding vectors are kept locally and no one has the complete access to the model. Therefore, these attacks against FL do not work in FR.

Attacks aginst federated recommendation. Previous studies on the attacks against centralized recommendation and FL are not effective in FR. Moreover, there are few studies on the attacks especially designed against FR. Zhang *et al.* [2022] and Rong *et al.* [2022] presented model poisoning attacks to FR named PipAttack and FedRecAttack respectively. However, both two attacks require attacker to have prior knowledge (*i.e.,* the side information that reflects each item's popularity or some public user-item interactions), which is not generic in all FR scenarios. From the above discussion we can conclude that, although there is increasing concern about the security issues of FR, the attacks against FR are still under-explored.

5 Experiments

In this section, we conduct extensive experiments to address the following research questions (RQs):

- **RQ1**: How is the effectiveness of our attacks compared

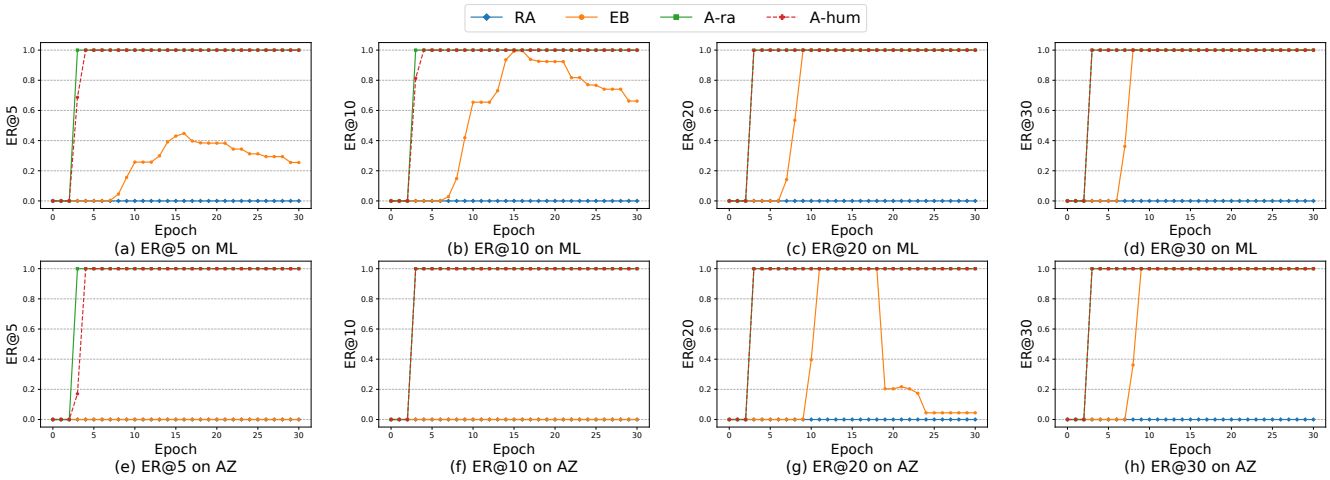

Figure 1: Attack Effectiveness under Different Attacks on ML ((a)-(d)) and AZ ((e)-(h)).

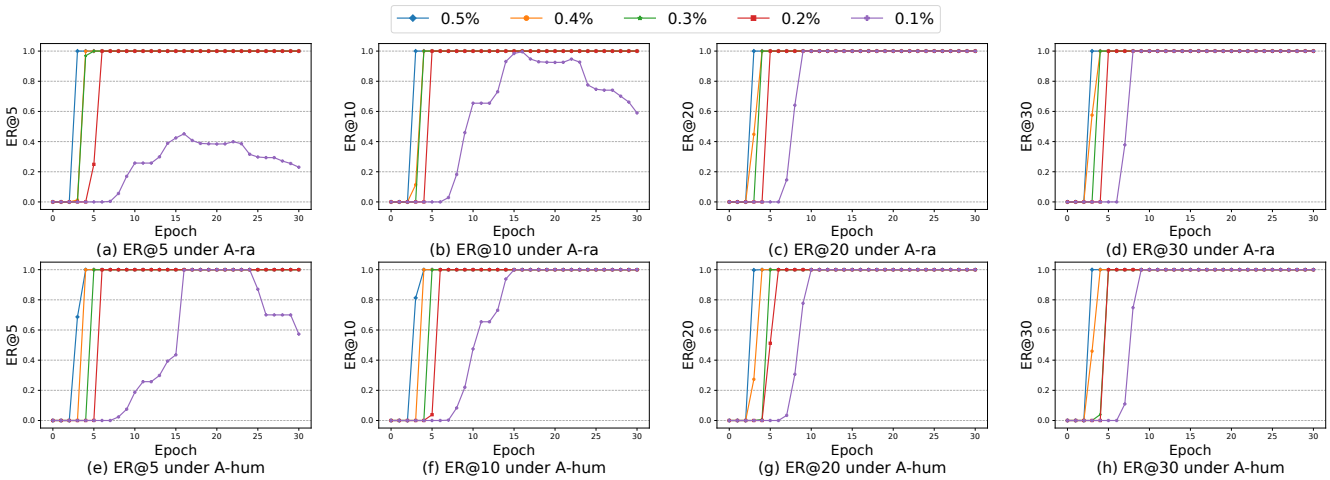

Figure 2: Attack Effectiveness under A-ra ((a)-(d)) and A-hum ((e)-(h)) with Different Malicious User Proportions.

to that of existing attacks?

- **RQ2**: Are our proposed attacks still effective when the proportion of malicious users is extremely small?
- **RQ3**: Does hard user mining really improve the attack effectiveness?

5.1 Experimental Settings

Dataset. We experiment with two popular and publicly accessible datasets: **MovieLens (ML)** and **Amazon Digital Music (AZ)**. For ML, we use the version containing 1,000,208 ratings involving 6,040 users and 3,706 movies, where each user has at least 20 ratings. For AZ, we use the version containing 169,781 reviews involving 16,566 users and 11,797 products of digital music, where each user and product has at least 5 reviews. In both datasets, we convert the user-item interactions (*i.e.,* ratings and reviews) into implicit data following [He *et al.*, 2017], and divide each user's interactions into training set and test set in the ratio of $4:1$. Note that we intentionally used two datasets of completely different sizes in order to validate the robustness of our attacks.

Evaluation Protocols. We assume that malicious users are injected by attacker before the training starts. As malicious users upload poisoned gradients in each epoch of training, we evaluate effectiveness of our attacks after each epoch. We adopt average Exposure Ratio at K (**ER@K**) of target items as our metric, where ER@K is defined in Section 3. High metric values indicate strong attack effectiveness. Note that to ensure fairness, we select T most unpopular items (*i.e.,* the items with the least number of interactions) as our target items on both datasets. The average ER@K of these items is the most difficult to be boosted. In our experiments, to observe the attack effects from various scales, we set $K = 5, 10, 20, 30$ respectively.

Baseline Attacks. Following the common setting for federated recommendation, we assume that attacker does not have any prior knowledge at all. Bandwagon attack [Kapoor *et al.*, 2017; Gunes *et al.*, 2014], PipAttack [Zhang *et al.*,

2022] and FedRecAttack [Rong *et al.*, 2022] rely on side infomation of items' popularity or public interactions, hence they are not applicable under such circumstances. The data poisoning attacks in [Li *et al.*, 2016; Huang *et al.*, 2021] rely on historical interactions of benign users, they are either not applicable. We choose the attacks which are still practical under such circumstances as our baseline attacks: **i) Random Attack (RA)** [Kapoor *et al.*, 2017]. It injects fake users as malicious users, and manipulates them to interact with both target items and randomly selected items. **ii) Explicit Boosting (EB)** [Zhang *et al.*, 2022]. It is one component of PipAttack which does not rely on attacker's prior knowledge. It explicitly boosts the predicted scores of target items for malicious users. More specifically, different from benign users, malicious users take the explicit item promotion objective as their loss function. Other than that, malicious users behave same as benign users.

Parameter Settings. We adopt NCF with 2 hidden layers as our base recommender model. The dimensions of both hidden layer are set to 8. The dimensions of users' embedding vectors and items' embedding vectors are also set to 8. The learning rate η for both benign users and malicious users is set to 0.001. The base recommender model is federated trained for 30 epochs on both ML and AZ to ensure convergence for recommendation. Moreover, we set r, T, n, σ, ξ and β to 4, 1, 10, 0.01, 0.001 and 30, respectively. Let ρ denote the proportion of malicious users. Unless otherwise mentioned, ρ is set to 0.5% on ML and 0.1% on AZ.

5.2 Comparison of Attack Effectiveness (RQ1)

We compare the effectiveness of baseline attacks and ours on ML and AZ. We set different proportion of malicious users on two datasets because attacks are more likely to achieve stronger effectiveness on AZ than on ML. More specifically, we set $\rho = 0.5\%$ for ML and set $\rho = 0.1\%$ for AZ. Figure 1 shows ER@5, ER@10, ER@20 and ER@30 under baseline attacks and our attacks on ML and AZ. As the figure shown: **(i)** Due to the proportion of malicious users is too small for the naive data poisoning attack RA, it has no effects. **(ii)** Though EB creates slight effects, its effectiveness is unstable. It could be easily seen that the results of EB continuously descrease once the training process passed half. **(iii)** Our proposed A-ra and A-hum outperforms baseline attacks consistently in terms of all metrics. The metric values of both A-ra and A-hum increase rapidly to 1 in the first few epochs, maintaining in high level throughout the whole rest of the training epochs.

5.3 Impact of Malicious User Proportion (RQ2)

To further explore the effectiveness of our attacks under certain circumstances, we evaluate our attacks with smaller proportion of malicious users on ML. Figure 2 shows attack effectiveness of A-ra and A-hum with $\rho = 0.5\%, 0.4\%, 0.3\%, 0.2\%, 0.1\%$ on ML. As the figure shown, the effectiveness of both attacks does not drop until the proportion of malicious users is decreased to 0.1%. Note that our attacks demand the lowest proportion of malicious users compared to any existing attacks against recommender systems (including the attacks with attacker's prior knowledge),

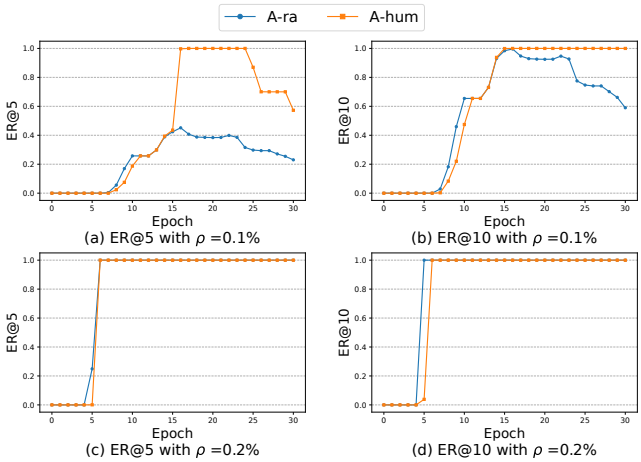

Figure 3: Attack Effectiveness under A-ra and A-hum with 0.1% malicious users and 0.2% malicious users.

not to mention that most attacks are only effective when the proportion of malicious users is more than 5%.

5.4 Ablation Test (RQ3)

By analyzing the effectiveness of our attacks on ML with 0.1% malicious users and 0.2% malicious users in Figure 3, it can be easily deduced that, although the strategy of hard user mining leads to a slower rise in attack effectiveness, it does improve overall attack effectiveness throughout the whole training epochs. More specifically, we compare ER@5 and ER@10 under A-ra and A-hum on ML as following: **(i)** As for A-hum, the values of ER@5 and ER@10 can reach maximum of 1, and respectively result in 0.5728 and 1 at the point that training process ends. **(ii)** As for A-ra, maximum values of 0.4514 and 0.9960 in same metrics are reached, decreased to 0.2302 and 0.5900 at the end. We argue that A-hum benifits from the strategy of hard user mining indeed. The trouble brought by hard users is more influential and challenging in cases of the proportion of malicious users being extremely small, and obviously A-hum is more effective in confronting the challenge.

6 Conclusion and Future Work

In this paper, we aim to attack deep learning based RS in FL scenarios without attacker's prior knowledge. We proposed two attack methods A-ra and A-hum, that use different ways to approximate benign users' embedding vectors. Experimental results demonstrate that our attacks set the state-of-the-art, and prove that there is necessary improvement should be made in FR. As part of future work, we will further explore the methods to detect the attacks in FR.

Acknowledgments

This research is supported by the National Key R&D Program of China (2021YFB2700500, 2021YFB2700502). Specially, the authors thank Jiasheng Chen (jasonalpaca@foxmail.com) as an independent researcher for his important contribution to the codes of this work.

References

[Ammad-Ud-Din *et al.*, 2019] Muhammad Ammad-Ud-Din, Elena Ivannikova, Suleiman A Khan, Were Oyomno, Qiang Fu, Kuan Eeik Tan, and Adrian Flanagan. Federated collaborative filtering for privacy-preserving personalized recommendation system. *arXiv preprint arXiv:1901.09888*, 2019.

[Bagdasaryan *et al.*, 2020] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

[Bai *et al.*, 2017] Ting Bai, Ji-Rong Wen, Jun Zhang, and Wayne Xin Zhao. A neural collaborative filtering model with interaction-based neighborhood. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1979–1982, 2017.

[Fang *et al.*, 2018] Minghong Fang, Guolei Yang, Neil Zhenqiang Gong, and Jia Liu. Poisoning attacks to graph-based recommender systems. In *Proceedings of the 34th Annual Computer Security Applications Conference*, pages 381–392, 2018.

[Fang *et al.*, 2020] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to byzantine-robust federated learning. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 1605–1622, 2020.

[Gu *et al.*, 2017] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[Gunes *et al.*, 2014] Ihsan Gunes, Cihan Kaleli, Alper Bilge, and Huseyin Polat. Shilling attacks against recommender systems: a comprehensive survey. *Artificial Intelligence Review*, 42(4):767–799, 2014.

[He *et al.*, 2017] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[He *et al.*, 2018] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. Outer product-based neural collaborative filtering. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 2227–2233. ijcai.org, 2018.

[Huang *et al.*, 2021] Hai Huang, Jiaming Mu, Neil Zhenqiang Gong, Qi Li, Bin Liu, and Mingwei Xu. Data poisoning attacks to deep learning based recommender systems. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. The Internet Society, 2021.

[Kapoor *et al.*, 2017] Saakshi Kapoor, Vishal Kapoor, and Rohit Kumar. A review of attacks and its detection attributes on collaborative recommender systems. *International Journal of Advanced Research in Computer Science*, 8(7), 2017.

[Li *et al.*, 2016] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. *Advances in neural information processing systems*, 29:1885–1893, 2016.

[Liang *et al.*, 2021] Feng Liang, Weike Pan, and Zhong Ming. Fedrec++: Lossless federated recommendation with explicit feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4224–4231, 2021.

[Lin *et al.*, 2020] Guanyu Lin, Feng Liang, Weike Pan, and Zhong Ming. Fedrec: Federated recommendation with explicit feedback. *IEEE Intelligent Systems*, 2020.

[McMahan *et al.*, 2017] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In Jeff A. Bilmes and Andrew Y. Ng, editors, *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, pages 452–461. AUAI Press, 2009.

[Rong *et al.*, 2022] Dazhong Rong, Shuai Ye, Ruoyan Zhao, Hon Ning Yuen, Jianhai Chen, and Qinming He. Fedrecattack: Model poisoning attack to federated recommendation. *arXiv preprint arXiv:2204.01499*, 2022.

[Shrivastava *et al.*, 2016] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 761–769, 2016.

[Wu *et al.*, 2021] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. Self-supervised graph learning for recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 726–735, 2021.

[Yao *et al.*, 2021] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, Felix Yu, Ting Chen, Aditya Menon, Lichan Hong, Ed H Chi, Steve Tjoa, Jieqi Kang, et al. Self-supervised learning for large-scale item recommendations. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 4321–4330, 2021.

[Zhang *et al.*, 2022] Shijie Zhang, Hongzhi Yin, Tong Chen, Zi Huang, Quoc Viet Hung Nguyen, and Lizhen Cui. Pipattack: Poisoning federated recommender systems for manipulating item promotion. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1415–1423, 2022.